S-RAMP Version 1.0. Part 1: Foundation

Committee Specification Draft 01 / Public Review Draft 01

08 April 2013

Specification URIs

This version:

http://docs.oasis-open.org/s-ramp/s-ramp/v1.0/csprd01/part1-foundation/s-ramp-v1.0-csprd01-part1-foundation.doc (Authoritative)
http://docs.oasis-open.org/s-ramp/s-ramp/v1.0/csprd01/part1-foundation/s-ramp-v1.0-csprd01-part1-foundation.html
http://docs.oasis-open.org/s-ramp/s-ramp/v1.0/csprd01/part1-foundation/s-ramp-v1.0-csprd01-part1-foundation.pdf

Previous version:

N/A

Latest version:

http://docs.oasis-open.org/s-ramp/s-ramp/v1.0/s-ramp-v1.0-part1-foundation.doc (Authoritative)
http://docs.oasis-open.org/s-ramp/s-ramp/v1.0/s-ramp-v1.0-part1-foundation.html
http://docs.oasis-open.org/s-ramp/s-ramp/v1.0/s-ramp-v1.0-part1-foundation.pdf

Technical Committee:

OASIS SOA Repository Artifact Model and Protocol (S-RAMP) TC

Chair:

Vincent Brunssen (brunssen@us.ibm.com), IBM

Editors:

Kurt Stam (kstam@redhat.com), Red Hat
Eric Wittmann (eric.wittmann@redhat.com), Red Hat

Additional artifacts:

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

- XML schemas: http://docs.oasis-open.org/s-ramp/s-ramp/v1.0/csprd01/schemas/
- *S-RAMP Version 1.0. Part 1: Foundation*. (this document) http://docs.oasis-open.org/s-ramp/s-ramp/v1.0/csprd01/part1-foundation/s-ramp-v1.0-csprd01-part1-foundation.html
- *S-RAMP Version 1.0. Part 2: Atom Binding*. http://docs.oasis-open.org/s-ramp/s-ramp/v1.0/csprd01/part2-atom-binding/s-ramp-v1.0-csprd01-part2-atom-binding.html

Related work:

This specification is related to:

- Service Oriented Architecture Ontology (http://www.opengroup.org/projects/soa-ontology/)
- XML Schema Part 1: Structures Second Edition (http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/)
- Web Services Description Language (http://www.w3.org/TR/2001/NOTE-wsdl-20010315)

Declared XML namespace:

- http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0

Abstract:

Vendors offer tools to facilitate various activities across the life cycle of a SOA artifact, such as design, assembly, quality assurance, deployment and runtime operation of SOA based applications and business processes. The lack of a standardized information model and interaction protocol for artifacts and their metadata residing in a SOA repository means that tools must be customized for use with each different vendor's SOA repository product. This reduces choice, flexibility and adds costs for customers when choosing tools. This specification defines a SOA artifact data model together with bindings that describe the syntax for interacting with a SOA repository.

Status:

This document was last revised or approved by the OASIS SOA Repository Artifact Model and Protocol (S-RAMP) TC on the above date. The level of approval is also listed above. Check the "Latest version" location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at http://www.oasis-open.org/committees/s-ramp/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/s-ramp/ipr.php).

Citation format

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

[S-RAMP-v1.0-Foundation]

S-RAMP Version 1.0. Part 1: Foundation. 08 April 2013. OASIS Committee Specification Draft 01 / Public Review Draft 01. http://docs.oasis-open.org/s-ramp/s-ramp/v1.0/csprd01/part1-foundation/s-ramp-v1.0-csprd01-part1-foundation.html.

Notices

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

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

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

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

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

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

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

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

Table of Contents

1 Introduction

The “SOA - Repository Artifact Model and Protocol” (S-RAMP) specification defines a common data model for SOA repositories to facilitate the use of common tooling and sharing of data. It provides a rich representation data model that supports query. It includes binding(s) that document the syntax for interaction with a compliant repository for create, read, update, delete and query operations within the context of each binding. Initially, only one binding will be defined, but others can be added.

The specification is organized into multiple documents. This document, the SOA - Repository Artifact Model and Protocol Foundation (hereafter referred to as Foundation) describes the overall goals of S-RAMP and defines the Artifact Type Model and associated schemas for S-RAMP. It also describes the generic query grammar used in S-RAMP. The Foundation document is not specific to any binding. Other documents in this specification provide material specific to a given binding for S-RAMP, including the syntax for interacting with an S-RAMP compliant repository. Those available at the time of this publication are:

- S-RAMP Atom Binding

When there is a discrepancy between a binding specific document and this Foundation document, the binding specific document always takes precedence. If there is a discrepancy between schema representations provided in this document and the S-RAMP schemas of record at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=s-ramp, the schemas of record SHALL take precedence.

1.1 Terminology

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

The characters "[" and "]" are used to indicate that contained items are to be treated as a group with respect to cardinality or choice.

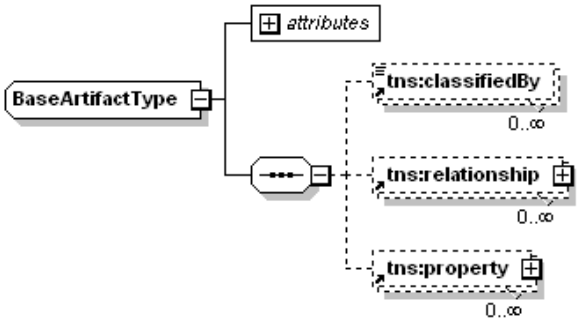
1.2 Diagrams used in this document

1.2.1 Attributes and elements

S-RAMP uses the XML Schema Language (See http://www.w3.org/TR/xmlschema-0/, http://www.w3.org/TR/xmlschema-1/ and http://www.w3.org/TR/xmlschema-2/) and its terminology, such as "sequence" and "choice" to formally describe its data structures. The diagrams[1] used in this specification show the structure and cardinality of the elements used in these structures. Attributes are not shown in the diagrams, but explained in the corresponding documentation.

[1] Diagrams provided in this specification were produced by the ©XML Spy editor, Altova GmbH and Altova, Inc.

1.2.2 Element structure



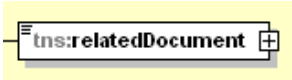
The octagonal symbol with the horizontal "dotted" line indicates "sequence of." This diagram says the type *BaseArtifactType* consists of elements *classifiedBy, relationship* and *properties*. All three elements are defined in the namespace whose prefix is "s-ramp".
The fact that *relationship* and *properties* have a box with a "+" in it at their right-hand end indicates that there is more structure to them than is shown in the diagram.

1.2.3 Cardinality

1.2.3.1 Optional, one

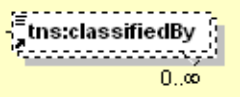
The dashed line indicates that the element *directsOrchestration* is optional. The fact that it is not adorned with some other cardinality indicator (see below) says there can be at most one of them.

1.2.3.2 Mandatory, one



There must be exactly one of the element *relatedDocument*.

1.2.3.3 Optional, repeating



The element *classifiedBy* is optional and may appear an indeterminate number of times. The number of times it may appear is given by the adornment "0..∞", a cardinality indicator meaning "zero to infinity". Other numbers may appear to indicate different cardinalities.

1.2.3.4 Mandatory, repeating

The element *policies* must appear at least once and may appear an indeterminate number of times.

1.3 Normative References

[RFC2119] Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels”, BCP 14, RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt.

[SOAONT] *Service Oriented Architecture Ontology*. October 2010. The Open Group. http://www.opengroup.org/projects/soa-ontology/

1.4 Non-Normative References

[XML] *Extensible Markup Language (XML) 1.0 Specification (Fifth Edition)*. November 2008. W3C Recommendation. http://www.w3.org/TR/2008/REC-xml-20081126/

[XMLNS] *Namespaces in XML 1.0 (Second Edition)*. August 2006. W3C Recommendation. http://www.w3.org/TR/2006/REC-xml-names-20060816/

[XSD] *XML Schema Part 1: Structures Second Edition, version 1.0*. October 2004. W3C Recommendation. http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/

[XPATH] *XML Path Language (XPath) 2.0 (Second Edition)*. December 2010. W3C Recommendation. http://www.w3.org/TR/2010/REC-xpath20-20101214/

[RDF] *RDF Primer*. February 2004. W3C Recommendation. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

[OWL] *OWL Web Ontology Language Guide*. February 2004. W3C Recommendation. http://www.w3.org/TR/2004/REC-owl-guide-20040210/

[WSDL] *Web Services Description Language (WSDL), Version 1.1*. March 2001. W3C Note. http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[ISO6392] *Codes for the Representation of Names and Languages – Part 2*. 1998. ISO 639-2. http://www.loc.gov/standards/iso639-2/normtext.html

[WSFWK] *Web Services Policy 1.5 – Framework*. September 2007. W3C Recommendation. http://www.w3.org/TR/2007/REC-ws-policy-20070904

[WSATTCH] *Web Services Policy 1.5 – Attachment*. September 2007. W3C Recommendation. http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904

[UUID] P. Leach, M. Mealling, and R. Salz, *A Universally Unique IDentifier (UUID) URN Namespace*. July 2005. IETF RFC 4122. http://www.ietf.org/rfc/rfc4122.txt

[QUERYOPS] *XQuery 1.0 and XPath 2.0 Functions and Operators (Second Edition).* December 2010. W3C Recommendation. http://www.w3.org/TR/2010/REC-xpath-functions-20101214/

1.5 Problem Statement and Objectives

Service Oriented Architecture (SOA) is an architectural approach to designing applications and business processes by consuming business logic from reusable software components exposed as network accessible services. In today's environment, vendors offer tools to facilitate various activities across the life cycle of a SOA artifact, such as design, assembly, quality assurance, deployment and runtime operation of SOA based applications and business processes. The SOA repository provides the foundation for all these activities. This specification describes how to represent SOA information models using artifacts and associated metadata, the Artifact Type Model that defines the artifacts, and the bindings to interact with the SOA repository, including create, read, update, delete, query, and subscription for notifications. This approach to providing flexible access to SOA artifacts will facilitate interoperability and provide customers with more choices of tools that can be used to interoperate with any S-RAMP compliant SOA repository implementation.

1.6 Use Case Scenarios

Table 1, Table 2 and Table 3 below provide some examples across different portions of the service lifecycle for which there are various use cases in which an S-RAMP compliant repository could be used. This does not necessarily imply that all vendors would support every scenario, or use an S-RAMP repository in each of these scenarios across all portions of the service lifecycle.

Table 1: Design Time Tool Repository Interaction Use Cases

Tool Category	Activities	S-RAMP Feature	Examples
Integrated Development Environment (IDE)	1. Design WSDL and schemas 2. Publish and consume services 3. Publish SCA into repository 4. Asset Relationship Visualization 5. Notification to service developer when WSDL changes	WsdlDocument XsdDocument OWL classifications PortType	WID RSA Together VisualStudio Oracle JDeveloper
Business Process Modeling Tools	1. Publish WSDL descriptions for processes 2. Look for process entry points 3. Search/find services to use in business processes 4. Impact analysis	wsdDocument XsdDocument OWL classifications PortType	WebSphere Business Modeling ARIS Platform Oracle (Collaxa) TIBCO Business Studio

Table 2: Run Time Tool Repository Interaction Use Cases

Tool Category	Activities	S-RAMP Feature	Examples
Testing	1. Search to find a WSDL 2. Understand policies associated with WSDL 3. Understand relationships between SOA components 4. Notification when WSDL changes	WsdlDocument Service PolicyAttachment Policy	HP Service Test Manager Rational Testing Tools Itko Lisa
ESB	1. Dynamic routing based on #services, requestor type, etc. 2. Notifications of new artifacts, changes/deletions 3. Track information on lifecycle and operational state (e.g., using classifications, properties, etc.)	PolicyAttachment WSDL Parts Service properties	DataPower WebSphere ESB Oracle Service Bus SAP XI WebMethods TIBCO ActiveMatrix
Policy Mgmt	1. Edit and store policies 2. Query repository for policies to deploy/provision 3. Execute (enforce) policy 4. Update managed endpoint in repository	PolicyAttachment policy service ServiceEndpoint	AmberPoint Actional HP SOA Policy Enforcer CentraSite TIBCO ActiveMatrix Policy Manager

114 *Table 3: Monitoring Tool Repository Interaction Use Cases*

Tool Category	Activities	S-RAMP Feature	Examples
Service Monitoring	1. Retrieve service definitions from repository 2. Update service information with performance and availability data 3. Discover dependencies between business services and web service instances 4. Discover what organizations provide a service 5. Discover operational data for the service for monitoring	Organization Service ServiceInstance ServiceOperation user properties	Tivoli CAM for SOA BAC for SOA AmberPoint Actional WebMethods Insight TIBCO ActiveMatrix Service Performance Manager

115

116 1.7 Design Principles

117 There are several high-level design principles to which S-RAMP has adhered:

118 • Use of existing standards where possible (e.g., XML, XML Schema, OWL, XPath2, APP (Atom
119 Publishing Protocol), ASF (Atom Syndication Format), etc.).

120 • Vendor neutrality.

121 • Does not include governance models, but may be used by them.

122 • Driven by use cases.

123 • Data model extensibility for new data types, and support for system and user defined metadata.

124 • Inclusion of an XML Schema based serialization for its data model.

125 • Use of XPath 2 to describe its query grammar.

126 • Use of OWL Lite to describe its classification system grammar.

127 • Separation of the data model from the bindings that describe the interaction APIs clients use to
128 interact with the repository.

129 1.8 S-RAMP Schemas

130 The schemas for the various S-RAMP Models are provided in the appendices. They closely follow the
131 conceptualized diagrams described in this document. The normative S-RAMP schemas of record define
132 the serialization for S-RAMP.

133 Notable points concerning the schemas in S-RAMP include:

134 • Built-in properties are typically represented as attributes.

135 • Types based on *BaseArtifactType* use an extension of that type as their base.

136 • Where practical, Global Element Declarations are provided.

137 • Extensibility in the Core Model is limited to the ##any attribute on most structures.

138 Schemas are provided for serialization purposes and the diagrams define the S-RAMP meta-model.

139 1.9 XML Namespaces

140 The XML namespace URI that MUST be used by implementations of this specification is:

141 http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0

142 Table 4 lists the XML namespaces that are used in this specification. The choice of any namespace prefix
143 is arbitrary and not semantically significant.

144

145 *Table 4: Prefixes and XML Namespaces Used in this Specification*

Prefix	XML Namespace	Specification(s)
xp2	http://www.w3.org/2005/xpath-functions	XPath 2.0
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	RDF namespace
rdfs	http://www.w3.org/2000/01/rdf-schema#	RDFS namespace
owl	http://www.w3.org/2002/07/owl#	OWL namespace
s-ramp	http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0	S-RAMP namespace
wsdl	http://schemas.xmlsoap.org/wsdl/	WSDL [WSDL 1.1]
wsp	http://www.w3.org/TR/2007/REC-ws-policy-20070904	WS-Policy [WS-Policy]
xsd	http://www.w3.org/2001/XMLSchema	XML Schema [Part 1, 2]

146

147

2 Artifact Type Model

The S-RAMP Artifact Type Model is a strongly typed data model for SOA repositories that enables interoperability among vendor repository implementations and tooling, within the context of a specific binding. It will also later serve as a foundation for developing data exchange and federation models. The Artifact Type Model is presented here using conceptualized models. Each of these models also has a XML Schema representation available in the Appendices.

2.1 Artifact Type Models

An artifact in S-RAMP is a container for all of the metadata that describes it. There are 4 major types of artifacts in S-RAMP. Each of these Artifact Types is discussed in more detail in later sections:

1. **Document Artifact**: Those S-RAMP defined artifacts that correspond to a physical document stored in the repository. Several important document types are pre-defined and have special support in S-RAMP (such as XML Schema or WSDL documents). But any document type can be placed in the repository.
2. **Logical Model Artifact**: Those S-RAMP defined artifacts that provide a representation of one of the pre-defined logical models (e.g. the SOA model or Service Implementation model).
3. **Derived Artifact**: Derived Artifacts (e.g., WSDL PortType, or WS-Policy PolicyExpression) are dynamically instantiated by the server as a consequence of publishing a document instance whose type is one of those supported with a Derived Model (see Table 5). These artifacts cannot be created or deleted directly, although clients can edit them to add or remove Generic relationships, properties and classifications. Derived Artifact Models are managed by the server and kept in synchronization with the document object with which they are associated. Derived artifacts provide a metadata model of the content components of a particular document. This allows much more powerful query capabilities at a granularity specific to the internal components of a document, when it is of a format supported with a Derived Model. Refer to Section **Error! Reference source not found.** for more information on the query model supported in S-RAMP, as well as to the individual binding document(s) for the query syntax pertaining to each binding.
4. **Extended Artifact**: These are created by the client in order to support artifact models not pre-defined by the S-RAMP specification. The means by which a client specifies a custom artifact model are beyond the scope of this specification, but some provision is made within the S-RAMP schema to facilitate basic interoperability for such artifacts. Regardless of the internal definition of these artifacts, they SHALL be serialized in S-RAMP as either an instance of *ExtendedArtifactType* (which extends BaseArtifactType), or as an instance of *ExtendedDocument* (which extends DocumentArtifactType). The latter should be used by clients when the custom artifact contains document content.

The pre-defined S-RAMP Artifact Types are organized into a set of logical models as summarized in Table 5 below. Each of these is discussed further in the sections that follow. Note that Derived Artifact Models are currently specified for each of the XSD, WSDL, and WS-Policy document types.

Table 5: Pre-Defined Artifact Type Models

Model	Purpose
Core	Defines the base data types used by the other models, as well as generic types for Documents and XML Documents.
SOA	Defines the artifact data types and relationships which are used to integrate The Open Group's SOA Ontology object model into S-RAMP's data model.
Service Implementation	Defines the artifact data types and relationships used to model the service implementation layer of a SOA environment.

Derived: XSD	Defines logical artifact data types for an XML Schema document.
Derived: WSDL	Defines logical artifact data types for a WSDL document.
Derived: SOAPWSDL	Defines artifact data types for the SOAP binding of a WSDL document.
Derived: Policy	Defines artifact data types for a WS-Policy document.

2.1.1 Artifact Metadata

An artifact can contain three major types of metadata. Each is discussed in detail in the sections that follow.

1. **Relationships**: These are directed associations that describe a conceptual link between two artifact instances. There are several types of relationships, which are defined below in Section **Error! Reference source not found.**.
2. **Properties**: These describe various named attributes associated with an artifact instance, and can be built-in or user-defined. Each S-RAMP property MUST have a single name that is unique to the artifact that it decorates. When present, an S-RAMP property SHALL have a single value.
3. **Classifications**: These define the classification system for a server, and are imported into a server as OWL documents. The means by which a client imports the system into the server is implementation specific and is beyond the scope of this specification. Clients MAY decorate artifacts with references to specific values in a classification system defined to the server.

Note that Artifact Type and Artifact Model values MUST also be unique.

2.1.2 Artifact Relationships

Relationships in S-RAMP are all directed from a source, to a target. Each relationship instance is the logical triple of the following 3 items of metadata:

1. **Relationship Type**. This is the name for the type of relationship. A number of these are pre-defined by S-RAMP in the various Artifact Models (e.g., “includedXsds”, “appliesTo”, …). There can be multiple relationship instances of the same Relationship Type.
2. **Source.** This is a reference to the artifact that is on the source side of the directed relationship. Relationships are always contained by the Source Artifact that “owns” them.
3. **Target.** This is a reference to the artifact that is on the target side of the directed relationship.

It is possible for a relationship of a given Relationship Type not to have a target, which is termed a “relationship with no targets”. In this case there is only one relationship instance with that Relationship Type for a given Source. Such relationships have a target cardinality of “0”. If there is a relationship instance with a given Relationship Type that does have a target, then there CANNOT also be a relationship instance with that Relationship Type which has no target.

There are 4 types of relationships supported in S-RAMP. Refer to the table in Appendix **Error! Reference source not found.** for a complete list of pre-defined relationships, an indication of whether the relationship is derived, and the model in which it occurs.

1. **Derived Relationships**. These are pre-defined relationships that cannot be directly created or deleted by the client. Cardinalities for such relationships are defined in the applicable Derived Model. All instances of an Artifact Type for which a Derived Relationship is defined, will always contain that Derived Relationship, even if there are no targets for that relationship. The S-RAMP serialization of a Derived Relationship uses named elements defined in the schema for the appropriate Artifact Type definition(s).
2. **Modeled Relationships.** These refer to the S-RAMP pre-defined relationships that may be edited by the client. Cardinalities for such relationships are defined in the applicable model (e.g., the Service Implementation Model or Core Model). All instances of an Artifact Type for which a Modeled Relationship is defined, will always contain that Modeled Relationship, even if there are no targets for that relationship. The S-RAMP serialization of a Modeled Relationship uses named

elements defined in the schema within the applicable Artifact Type. There are several considerations related to target cardinality of Modeled Relationships:

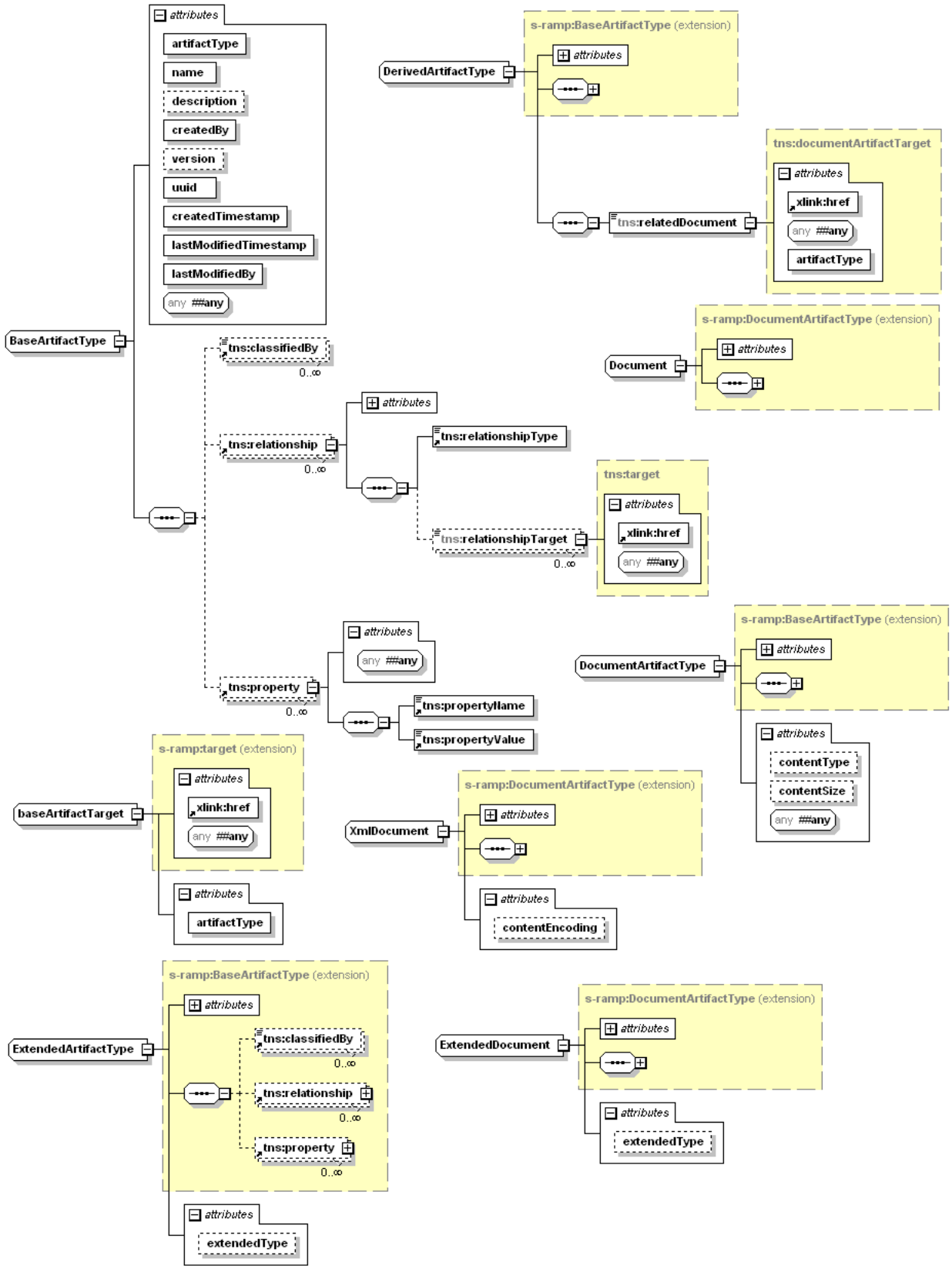
- **Modeled Relationships with Minimum Cardinality = 0:**
  - Instances of the Source Artifact can be created independently of the Target Artifact.
- **Modeled Relationships with Minimum Cardinality > 0:**
  - Instances of the Source Artifact cannot be created without the appropriate Target Artifact(s) based upon the required minimum cardinality of the relationship.
  - This can be accomplished by publishing the relevant artifacts at the same time, or by publishing the target artifact(s) first.
  - Actions that result in the deletion of a relationship instance are not permitted if that would result in a violation of the minimum cardinality.
- **Modeled Relationships with Maximum Cardinality < unbounded:**
  - Relationship instances cannot be created if that would result in a violation of the maximum cardinality limit for the Modeled Relationship.

Note that in cases where the minimum cardinality equals the maximum cardinality, such a relationship must be created or updated in a single step to avoid intermediate states that would violate these requirements.

3. **Generic Relationships**. These are user-defined ad-hoc directed relationship instances between any two artifacts in S-RAMP. They always have a minimum cardinality of 0 and an unbounded maximum cardinality. The Relationship Type value of a Generic Relationship instance is chosen by the client, but it MUST NOT match any pre-defined Relationship Type values already defined by the S-RAMP Modeled and Derived relationships (see Appendix **Error! Reference source not found.**). The S-RAMP serialization of a Generic Relationship uses the *relationship* structure defined in the Core Model.
4. **Extended Artifact Modeled Relationships**. Users may define their own extended artifact models, which may include modeled relationships. How and whether such models are supported is beyond the scope of this specification. Such models are called “Extended Models”. Since pre-defined relationships in a model are termed “Modeled”, then in this context they are called “Extended Modeled Relationships”. The S-RAMP serialization of an Extended Modeled Relationship uses the S-RAMP *relationship* structure defined in the Core Model.

2.2 The Core Model

There is a “core” model that defines all the basic Artifact Types used throughout S-RAMP. The Core Model contains abstract base artifacts for document artifacts and derived artifacts (which are associated with certain document types). Most Artifact Types in the other S-RAMP models are extensions of *BaseArtifactType*. Additionally, the Core Model includes the Document and XmlDocument concrete Artifact Types.



265

266 *Figure 1: Conceptualized Model of Core Model Artifacts*

267

Error! Reference source not found. provides a conceptualized illustration of the Core Model artifacts. In addition, there are a number of support types used by the core and other models. Note that the class attributes in the diagram are essentially built-in properties. The remaining sub-sections provided additional details on the Core Model.

2.2.1 Base Artifact Type

The *BaseArtifactType* is the fundamental abstract type used by all of the artifact models in S-RAMP. It contains all of the common metadata that describes an artifact instance. All artifact instances that are based on the *BaseArtifactType* contain the following metadata.

2.2.1.1 Built-in Properties:

- **artifactType**: A required string which is set when a specific type of artifact is created. These artifact types are enumerated in a list of core data types. The enumeration is defined by the baseArtifactEnum.
- **createdBy**: A required string assigned by the server identifying the user who created the artifact. S-RAMP does not define requirements on this value. These are implementation specific.
- **createdTimestamp**: A required timestamp which is set by the server at the time an artifact is first published. It conforms to xsd:dateTime, referenced to UTC.
- **description**: This optional property is used to provide a human consumable description of the artifact instance. This value is set by the client for all non-Derived Artifacts (although implementations MAY support setting it automatically using introspection. Derived Artifact descriptions are set by the server.
- **lastModifiedBy**: A required string assigned by the server identifying the user who last updated the artifact. S-RAMP does not define requirements on this value. These are implementation specific.
- **lastModifiedTimestamp**: A required timestamp which is updated by the server each time an artifact instance is modified. It conforms to xsd:dateTime, referenced to UTC.
- **name**: This required property is used to describe the artifact instance. This value is set by the client for all non-Derived Artifacts (although implementations MAY support setting it automatically using introspection). Derived Artifact names are set by the server.
- **uuid**: A required unique identifier of an artifact instance in the repository. This value conforms to Type 4 random-number format UUIDs **[UUID]**, and is set for the artifact at the time of its creation. The repository will assign a value if the user does not provide one.
- **version**: An optional string representing the version of the artifact instance. S-RAMP makes no attempt to define formatting rules for this property, which are implementation specific.

2.2.1.2 Generic Properties

- ***property***: These are optional properties which are defined by the client. They MUST have a single unique name (which SHALL NOT duplicate any other property name of any type, and SHALL NOT duplicate any Relationship Type value). A property name SHALL have 0 or 1 value.

2.2.1.3 Generic Relationships

- ***relationship***: These are optional relationship(s) defined by the client. A relationship contains a Relationship Type identifying the type of the relationship, and 0 or more target artifact references. Relationship Type values within a relationship SHALL NOT duplicate the name of any property.

2.2.1.4 Classifications

- ***classifiedBy***: This is a separate class of metadata. It MAY be set by the client with an unbounded upper cardinality limit. Each value SHALL be a URI that references a specific OWL

class from a classification system defined to the repository. For more on OWL classification systems in S-RAMP, see Section **Error! Reference source not found.**.

2.2.2 Document Artifact Types

The *DocumentArtifactType* is the fundamental abstract data type for all documents represented in the repository, and it extends *BaseArtifactType*. This Artifact Type includes several built-in properties:

contentType:

- A string indicating the MIME Media type of the content. This is set by the server as part of processing the publication of the document, and cannot be changed by the user.

contentSize:

- An integer representing the size of the content in bytes. This is set by the server as part of processing the publication of the document. It cannot be changed by the user.

contentHash:

- A string representing the SHA-1 hash of the document's byte contents. This is set by the server as part of processing the publication of the document. It cannot be changed by the user.

The Core Model also includes an *XmlDocument* type that all XML based document data types extend. The *Document* type provides a concrete artifact that can be used to represent arbitrary document types (e.g. PDF or Office documents).

Documents which have a Derived Model associated with them cannot be updated in the repository. They must be removed and republished.

Documents which are the target of a relationship cannot be deleted.

2.2.3 Miscellaneous Types

There are a few miscellaneous classes in the Core Model:

StoredQuery:

- This is a special Artifact Type that is used to persist queries in the repository. Additional information on this topic is available in Section **Error! Reference source not found.**, **Error! Reference source not found.**.

ExtendedArtifactType:

- The *ExtendedArtifactType* allows clients to create their own extended artifact model when it is not pre-defined by the S-RAMP specification. The *extendedType* property is intended to provide an indication of the artifact type.

ExtendedDocument:

- The *ExtendedDocument* allows clients to create their own extended artifact model when it is not pre-defined by the S-RAMP specification. The *extendedType* property is intended to provide an indication of the artifact type. This differs from the ExtendedArtifactType in that it extends DocumentArtifactType, thereby inheriting its properties. This type may be used by clients when adding extended artifact model artifacts that contain document content.

2.3 Modeling SOA Concepts

S-RAMP supports modeling of business level SOA concepts related to service and process representations and interactions. Since it is not the mission of S-RAMP to define a "SOA Ontology" for the industry, this specification has chosen to reference work done by The Open Group in their "SOA Working Group" on the "SOA Ontology" (see http://www.opengroup.org/projects/soa-ontology/). That work is compatible with both SCA and BPMN but draws both service and process concepts together at a higher level of abstraction.

S-RAMP supports modeling of SOA concepts using a layered approach:

1. S-RAMP SOA Model
2. S-RAMP Service Implementation Model

The sections below describe how the SOA Ontology work is integrated with S-RAMP as well as the implementation layer underneath it.

2.3.1 The SOA Model

The S-RAMP SOA Model exists to provide a mechanism to link work done by The Open Group SOA Ontology work group (**[SOAONT]**) with the rest of the S-RAMP internal models. It defines a very minimal set of linkages to artifacts defined in the SOA Ontology.

Several S-RAMP modeling features have been defined in order to provide linkage between the SOA Ontology and the S-RAMP data model. This is done using the S-RAMP SOA Model. Artifact Types in this model are all user instantiated, and are described in the S-RAMP SOA Model illustrated in *Figure 2* below.

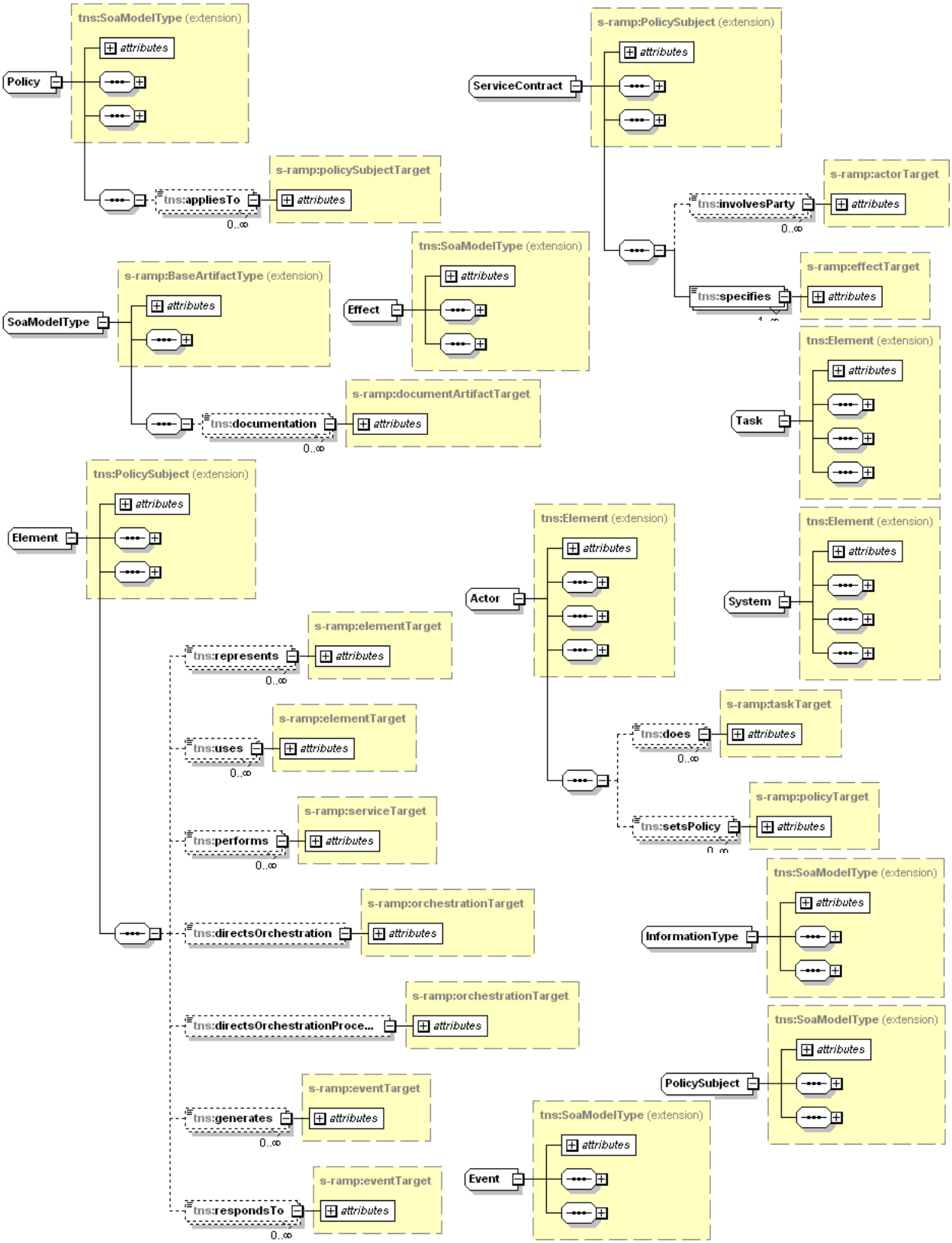


Figure 2: Conceptualized Model of SOA Model Artifacts (part 1)

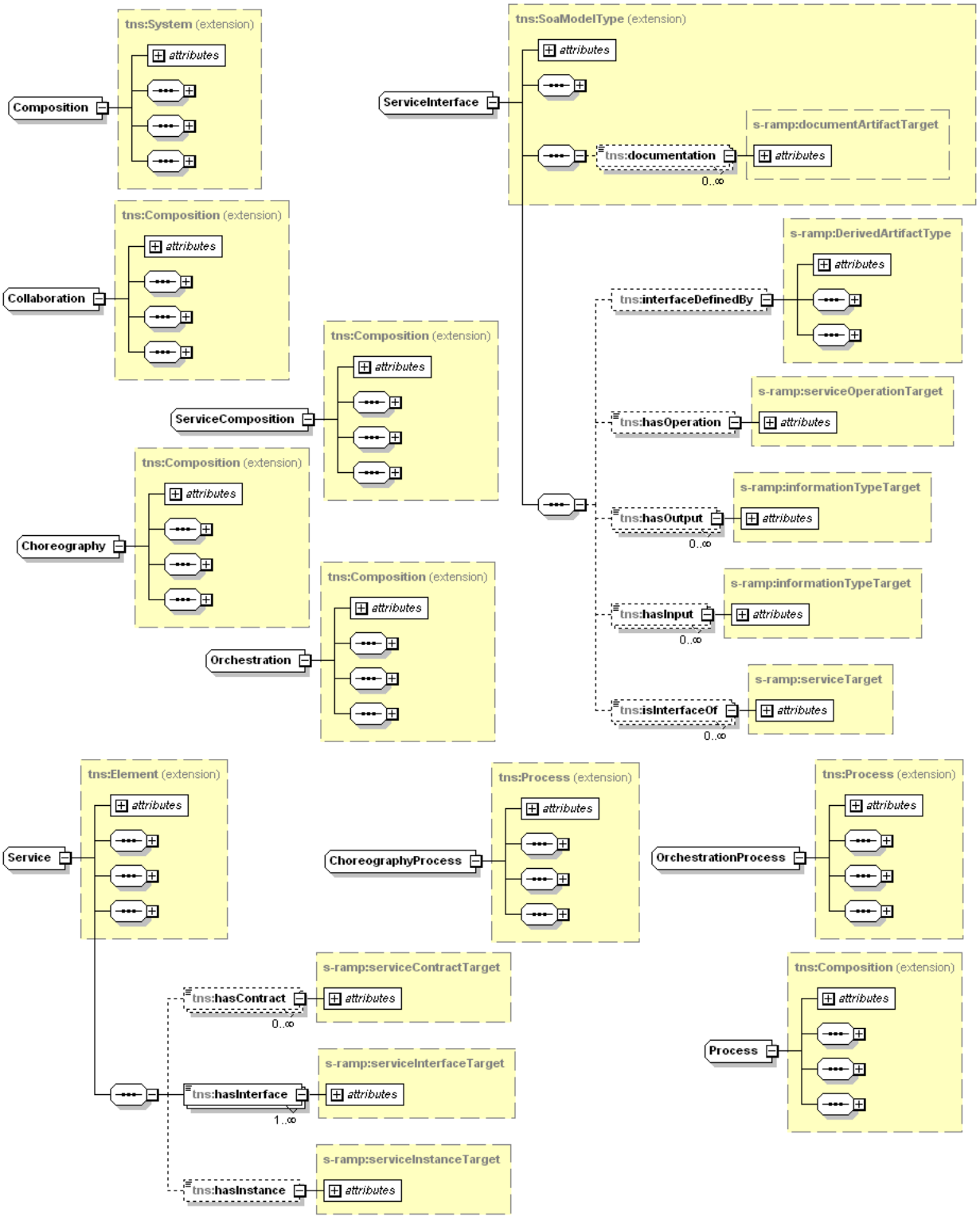


Figure 3: Conceptualized Model of SOA Model Artifacts (part 2)

SOA Model artifacts and the SOA Ontology artifacts referenced by the SOA Model are designed to provide clients the ability to create conceptual SOA representations. SOA Model Artifacts are all logical

artifacts and do not represent or correspond directly to a document instance, as do those in the Derived Models.

S-RAMP provides an XML Schema representation of the S-RAMP SOA Model, including elements corresponding to the base version of The Open Groups SOA Ontology's defined artifacts, within the context of the S-RAMP data model. It can be found in Appendix **Error! Reference source not found.**.

2.3.1.1 SOA Model Artifact Types and Relationships

The abstract *SoaModelType* Artifact Type implicitly acts as a super class for ALL top level SOA Ontology artifacts when they are used within the context of S-RAMP. This imbues all of them with the properties built into all S-RAMP Artifact Types. S-RAMP adds several relationships to SOA Ontology Artifacts used in the SOA Model in order to provide a connection from the SOA Ontology artifacts into the implementation level artifacts described in the Service Implementation Model in Section **Error! Reference source not found.**.

SOA Model Artifacts MAY have relationships to Document Artifacts and/or Derived Artifacts in other models, as well as among themselves. All the relationships defined in the SOA Model are Modeled Relationships. See Section **Error! Reference source not found.** for behavioral details associated with Modeled Relationships. The relationships that have been added to artifacts from the SOA Ontology are summarized in **Error! Reference source not found.** below.

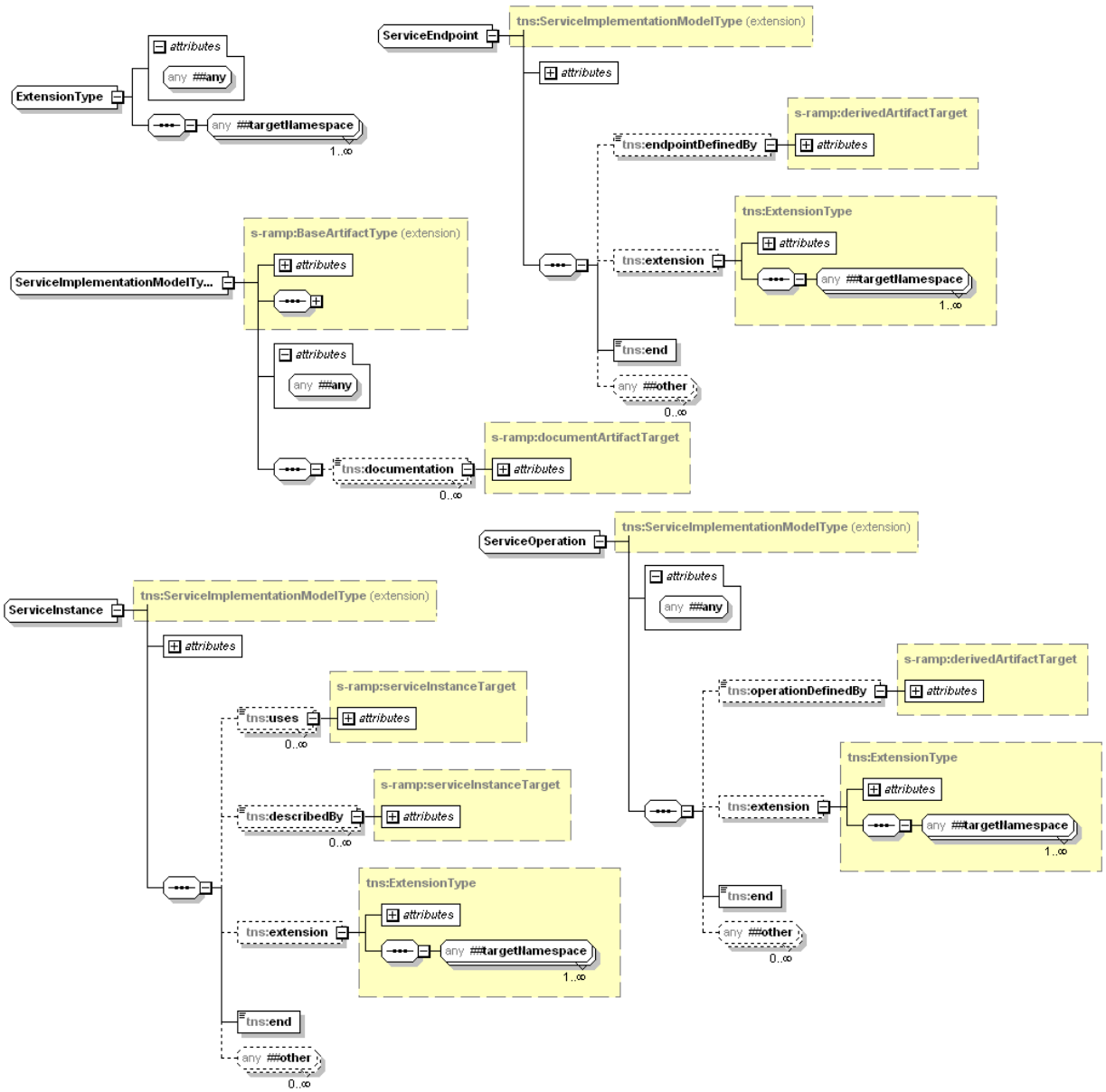
Table 6: SOA Model Relationships

Relationship Name	Source Artifact Type (from SOA Ontology)	Target Artifact Type (from S-RAMP Business & Core Models)	Notes
hasInstance	Service	ServiceInstance	
hasOperation	ServiceInterface	ServiceOperation	
interfaceDefinedBy	ServiceInterface	*DerivedArtifactType*	This allows one to indicate the Derived Artifact instance which defines this service interface. For example, this could be *PortType* artifact instance from the WSDL Model.
policyDefinedBy	Policy	*DerivedArtifactType*	This allows one to link the SOA Model Policy artifact with a concrete S-RAMP derived artifact type (e.g., PolicyAttachment)
informationTypeDefinedBy	InformationType	*DerivedArtifactType*	This allows one to link the SOA Model InformationType artifact with a concrete S-RAMP derived artifact type (e.g., PolicyAttachment)

2.3.2 The Service Implementation Model

S-RAMP defines a "Service Implementation Model" which describes the service implementation layer underneath the SOA Model. Artifact Types in this model are all user instantiated and most are extensible. Each of these artifacts derives from the *ServiceImplementationModelType* and may be created, changed, updated and deleted by the client. There are a number of pre-defined Modeled Relationships between artifacts of particular types. **Error! Reference source not found.** below illustrates the conceptualized Service Implementation Model artifacts.

406



407

408 *Figure 4: Conceptualized Model of Service Implementation Model Artifacts*

409

410 S-RAMP provides an XML Schema representation of the Service Implementation Model. It can be found
411 in Appendix **Error! Reference source not found.**. The sub-sections that follow discuss each of the
412 Service Implementation Artifact types in more detail.

413 2.3.2.1 Service Implementation Model Artifact Types and Relationships

414 The primary Artifact Type from which all Service Implementation Model Artifacts extend is the abstract
415 *ServiceImplementationModelType.* The concrete Service Implementation Model Artifacts that extend it
416 are designed to provide the implementation layer below the SOA Model which allows clients to build SOA
417 representations. Service Implementation Model Artifacts are all logical artifacts and do not represent or
418 correspond directly to a document instance as do those in the Derived Models.

Service Implementation Model Artifacts MAY have relationships to Document Artifacts and/or Derived Artifacts in other models, as well as among themselves. All the relationships shown in the Service Implementation Model are Modeled Relationships. See Section **Error! Reference source not found.** for behavioral details associated with Modeled Relationships. Of note for the *ServiceImplementationModelType* is the *documentation* Modeled Relationship which allows any artifact in the Service Implementation Model to reference a document describing it.

The concrete Service Implementation Model Artifact Types are then:

Organization

- The *Organization* type is used to describe an organizational entity. It is a subclass of the "Human Actor" artifact defined in the SOA Ontology. Clients can define an unlimited number of organizations and can use the *provides* relationship to link an Organization to any Service Implementation Model Artifact

ServiceInstance

- The *ServiceInstance* Artifact Type represents deployed instance(s) of a service. For example, a Web service running in WebSphere, or Oracle Fusion, etc. The *describedBy* Modeled Relationship can be used to reference document artifact(s) that describe it.

ServiceEndpoint

- The *ServiceEndpoint* Artifact Type represents a physical location at which the Service instance can be invoked, using its *url* Modeled Property. The *endpointDefinedBy* Modeled Relationship allows indicating the Derived Artifact instance that defines this Service endpoint is defined. For example, this could be a *Port* artifact instance from the WSDL Model.

ServiceOperation

- The *ServiceOperation* Artifact Type represents the specific operation performed by the Service. The *operationDefinedBy* Modeled Relationship can be used to link a *ServiceOperation* with a Derived Artifact which defines it. For example this could be an *Operation* artifact instance from the WSDL Model.

2.4 Derived Models

The sections that follow describe the logical models that are constructed by the repository in response to publication of a document for which a Derived Model is defined. The server parses these documents upon publication, dynamically constructing Derived Artifacts corresponding to the major data type components of the document. Every Derived Artifact instance has a *relatedDocument* relationship to the document from which it was created. The artifacts and relationships in a Derived Model provide a powerful tool for searching the repository based on specific components of such a document (e.g., a WSDL PortType, etc.). In most cases, the model diagrams illustrated in this section are adequate to define the Artifact Types and the Derived Relationships between them, so these sections are correspondingly brief.

Appendix **Error! Reference source not found.** describes all of the Derived Model schemas defined in S-RAMP.

2.4.1 The XSD Model

The XSD Model describes the Artifact Types that correspond to components of an XSD document stored in the repository, and yields structural metadata useful in performing queries. The conceptual model describing the XSD Model is illustrated below in Figure 5: Conceptualized Model of XSD Model Artifacts.

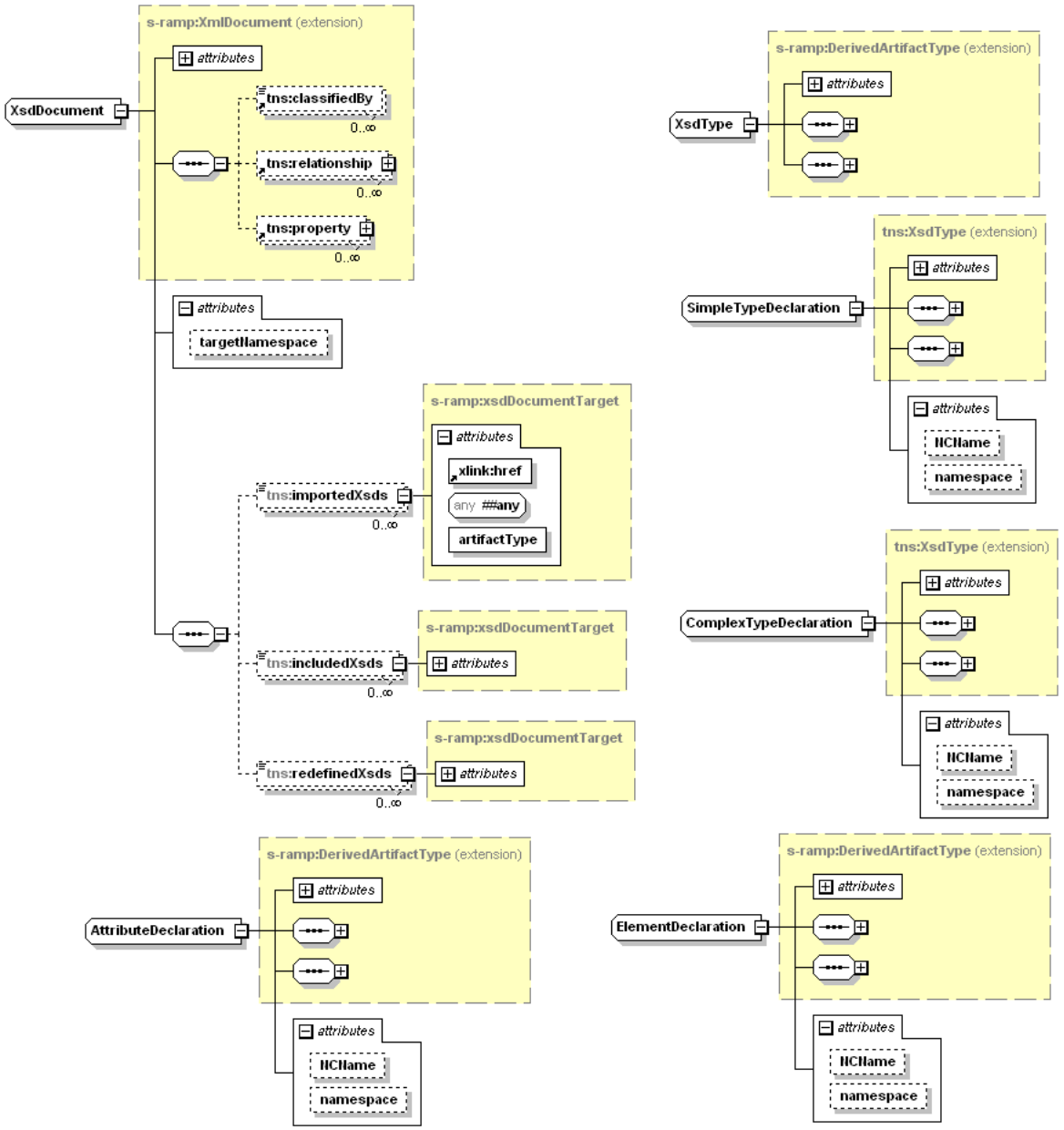


Figure 5: Conceptualized Model of XSD Model Artifacts

Note that an XSD document MAY include, import, or redefine other XSD documents. These capabilities are modeled using the *includedXsds*, *importedXsds* and *redefinedXsds* Derived Relationships, respectively.

A *SimpleTypeDeclaration* artifact instance is generated for each global simple type defined in the associated XML Schema document. Similarly, a *ComplexTypeDefinition* instance is generated for each global complex type defined in that XML Schema document. Each global attribute declared in the XML Schema document will generate a corresponding *AttributeDeclaration* artifact instance, and finally, each global element declared in the XML Schema document will generate a corresponding *ElementDeclaration* artifact instance.

While it is possible to construct a more fine-grained model of an XML Schema document, this level of modeling provides a suitably rich context for discovery queries without creating an overly complex model.

2.4.2 The WSDL Model

The WSDL Model is one of the most complex Derived Models, owing to the complexity of WSDL itself. The WSDL Model contains substantial richness because of the value in being able to perform highly refined queries using logical artifact representations of most of a WSDL document's constituent components.

Figure 6 and Figure 7 below provide a conceptual model representing the logical artifacts and their Derived Relationships in the WSDL Model. This model is intended to mirror the structure of a WSDL file. In most cases, the artifact names exactly match corresponding data types in WSDL (e.g., *Message, PortType, Operation, Binding, Service, Port*, and so on).

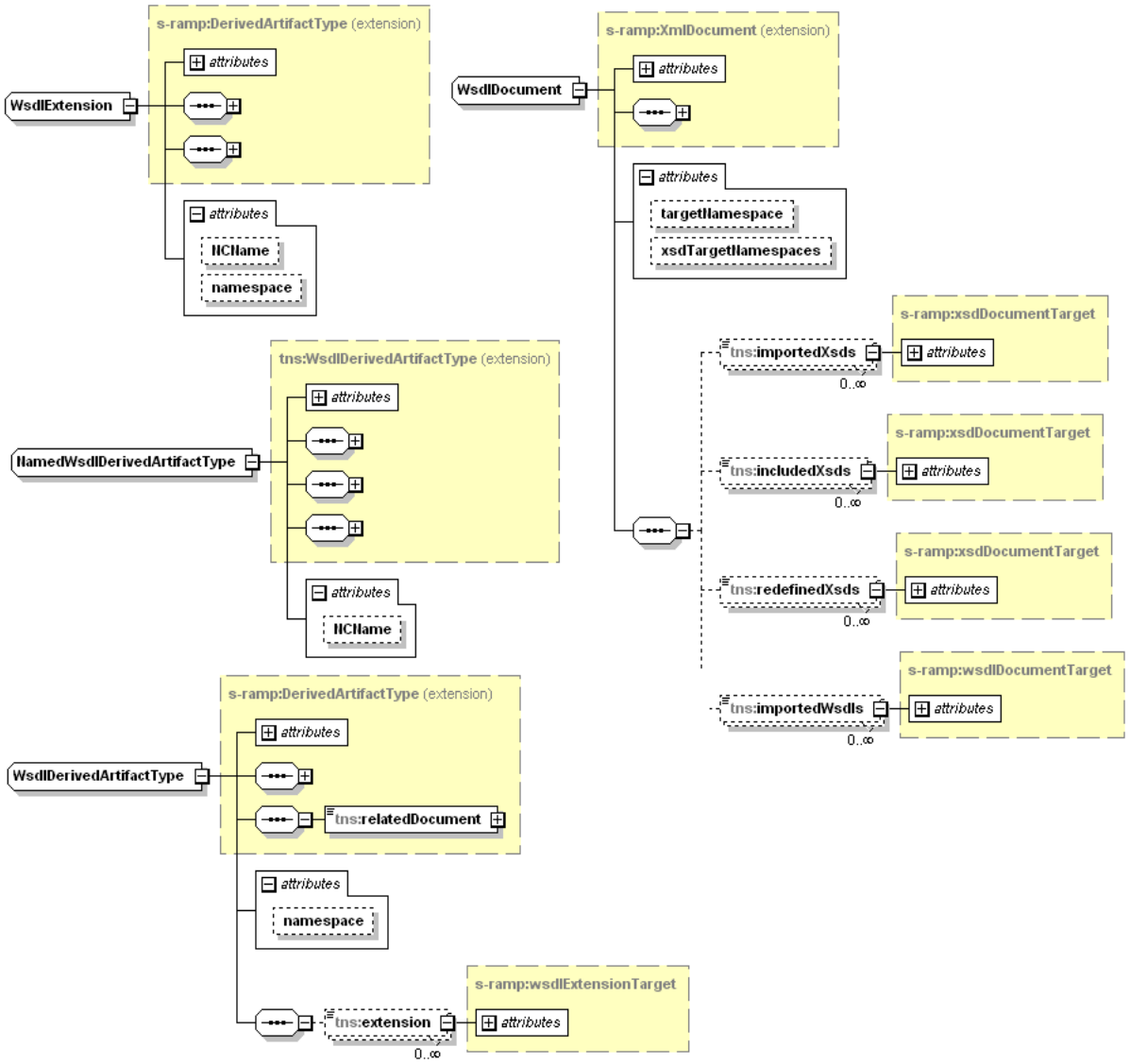


Figure 6: Conceptual Diagram of WSDL Model: Part 1

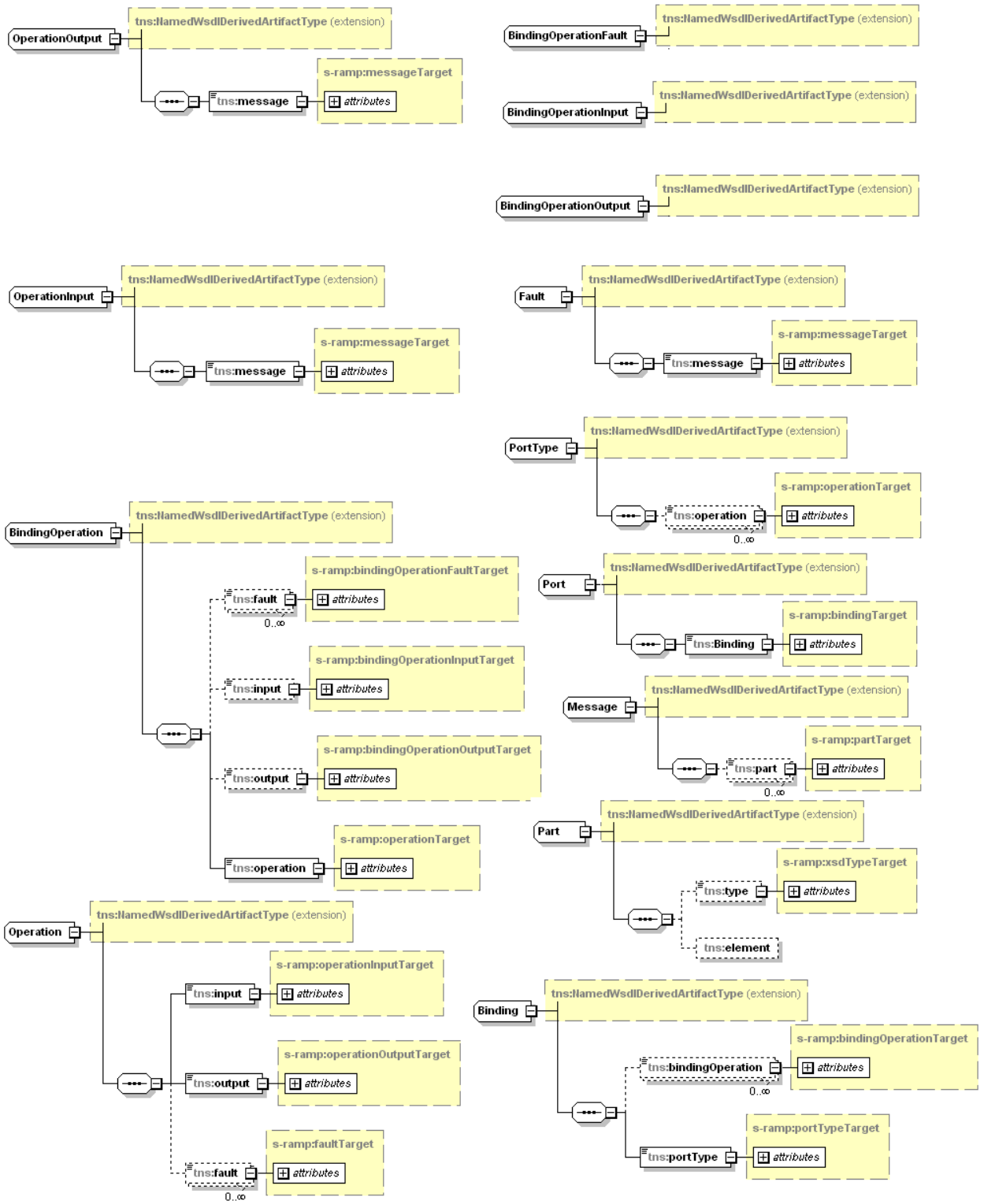


Figure 7: Conceptual Diagram of WSDL Model: Part 2

Note that aggregation relationships in these diagrams are modeled as Derived Relationships in the WSDL Model schema found in Appendix **Error! Reference source not found.**.

Note that the WsdlExtension Artifact Type illustrated in Figure 7 below is used to model extensibility elements for any given WSDL element. This supports WSDL extensions that the repository does not recognize.

The WsdlDerivedArtifactType is a modeling convenience from which all WSDL related metadata artifacts extend. It contains a *namespace* attribute that is the namespace of the WSDL document and it applies to every Derived Artifact instance for that Document. It also has an *extension* Derived Relationship which serves to identify the set of WSDL extensions which apply to the Derived Artifact.

A WSDL document can import, include and redefine XSD document(s), as well as import other WSDL documents. These capabilities are modeled using the *importedXsds*, *includedXsds*, *redefinedXsds*, and *importedWsdls* relationships, respectively. .

2.4.3 The SOAPWSDL Model

The SOAPWSDL Model contains the SOAP 1.1 binding specific WSDL Model artifacts for WSDL 1.1. It is separated into its own Model and schema to simplify its use. Figure 8 below illustrates a conceptual model of the relevant SOAP WSDL Model artifacts.

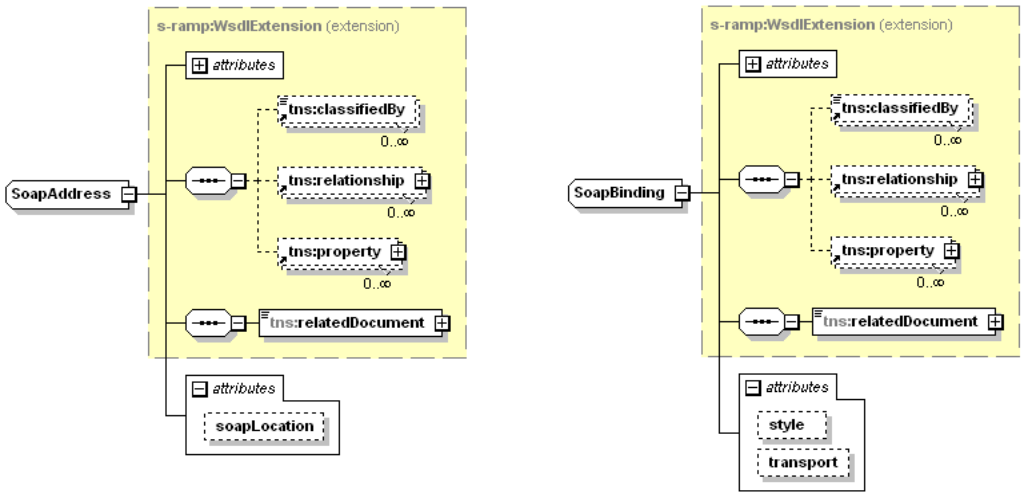


Figure 8: Conceptualized Diagram of the SOAP WSDL Model

2.4.4 The Policy Model

The Policy Model describes the Artifact Types that correspond to the primary components of a WS-Policy document. Policy expressions can be in a standalone policy document, or embedded in another document such as a WSDL. Figure 9 below is a conceptual diagram of the Policy Model:

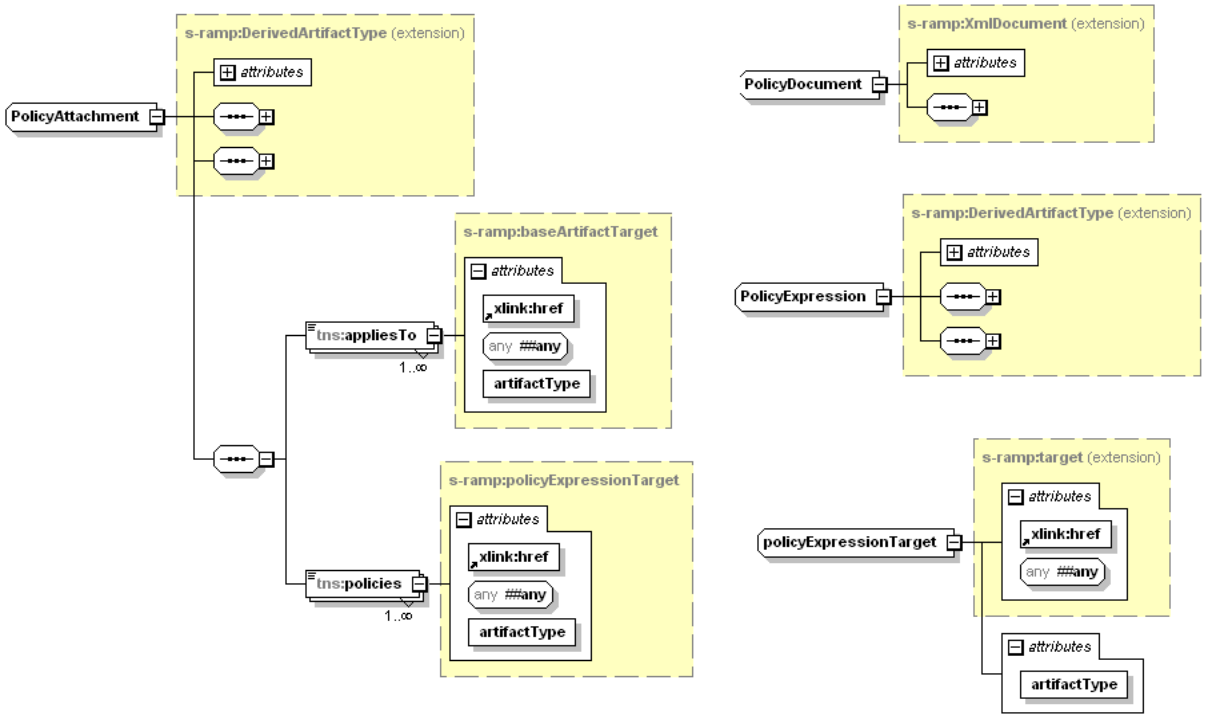


Figure 9: Conceptualized Model of Policy Model Artifacts

All *appliesTo* relationships are instantiated in the repository based on the content in the policy attachment document. There is no restriction on the artifacts it can reference.

2.5 Referencing S-RAMP Artifacts

The syntax for referencing Artifact Type(s) is defined in each of the S-RAMP bindings (e.g., the S-RAMP Atom Binding). Please refer to the appropriate binding specific document of this specification for details.

2.5.1 Notional Syntax

One possible syntax is as follows and is used in several examples within this specification:

```
/s-ramp/{ArtifactModel}/{ArtifactType}
```

Note that each successive component of this syntax is optional. Specifically

- A reference of "/s-ramp" refers to all Artifact Types in all models
- A reference of "/s-ramp/{Artifact Model}" refers to all Artifact Types within the specified Artifact Model.
- References of the form "//{ArtifactModel}" or "//{ArtifactType}" are also permitted.

Table 7: Artifact Models and Types below provides the pre-defined values for Artifact Model and Artifact Types. Abstract types are not included since they cannot be instantiated.

Table 7: Artifact Models and Types

Artifact Model	Artifact Type

core	Document
	XmlDocument
xsd	XsdDocument
	AttributeDeclaration
	ElementDeclaration
	SimpleTypeDeclaration
	ComplexTypeDeclaration
policy	PolicyDocument
	PolicyExpression
	PolicyAttachment
soapWsdl	SoapAddress
	SoapBinding
wsdl	WsdlDocument
	WsdlService
	Port
	WsdlExtension
	Part
	Message
	Fault
	PortType
	Operation
	OperationInput
	OperationOutput
	Binding
	BindingOperation
	BindingOperationInput
	BindingOperationOutput
	BindingOperationFault
serviceImplementation	Organization
	ServiceEndpoint
	ServiceInstance
	ServiceOperation
ext	{ExtendedType}
soa	{See The Open Group's SOA Ontology for the normative list of Artifact names. They are reproduced here for clarity.}

HumanActor
Choreography
ChoreographyProcess
Collaboration
CollaborationProcess
Composition
Effect
Element
Event
InformationType
Orchestration
OrchestrationProcess
Policy
PolicySubject
Process
Service
ServiceContract
ServiceComposition
ServiceInterface
System
Task

Below are some examples of S-RAMP model and Artifact Type references using the notional syntax described above.

Example 1: Artifact Model and Type References

```
/s-ramp
    o   References all Artifact Types in the repository
/s-ramp/xsd    or     //xsd
    o   References all Artifact Types in the XSD Model (e.g., XsdDocument, XsdType, …)
/s-ramp/xsd/XsdType    or     //XsdType
    o   References the XsdType Artifact Type in the XSD Model
/s-ramp/soa/Service
    o   References all Service Artifacts Types in the SOA Model (included by reference to
        The Open Group's SOA Ontology)
/s-ramp/wsdl/Port
    o   References the Port Artifact Type in the WSDL Model
```


3 Classification Systems in S-RAMP

A classification system allows classification values to be organized in a hierarchy, allowing artifacts to be grouped into sets and to identify subsets within those sets. Query and display of all artifacts in a set then becomes possible and provides a simple yet powerful means of classifying and finding related objects. Artifact instances in the repository MAY have classification values applied to them.

A simple example of a classification system hierarchy is a geographical region hierarchy that could, for example, be used to indicate which location produced an artifact. Using '/' to delineate levels in the hierarchy and ',' to separate members at the same level, one part of an example hierarchy could be World / Asia / Japan, China. The hierarchy could also contain World / Europe / United Kingdom, Germany. In other words, subsets of World are Asia and Europe, subsets of Asia are Japan and China, and subsets of Europe are United Kingdom and Germany. To find artifacts produced by teams in Asia, a search can be issued to return artifacts classified by the Asia classification. To give the behavior desired, a repository implementation interprets this query as requiring the retrieval of artifacts classified by Asia and its sub-groups i.e. Asia or Japan or China.

Classification systems used in S-RAMP are expressed in the Web Ontology Language (OWL) that builds upon the Resource Description Framework (RDF). The RDF/XML serialization format is commonly used for files containing OWL constructs and is a format that SHALL be understood by an S-RAMP repository. OWL consists of three increasingly expressive sublanguages, OWL Lite, OWL DL and OWL Full. The W3C "OWL Web Ontology Language Overview" document indicates that "OWL Lite supports those users primarily needing a classification hierarchy and simple constraints", and accordingly the OWL elements that S-RAMP uses to define classification systems come from OWL Lite. To enable the classification capabilities required in S-RAMP, a restricted set of OWL Lite elements is sufficient. Also, it should be noted that S-RAMP does not support multiple inheritance.

The elements that SHALL be supported and a brief description of each follow (refer to the W3C documents detailing OWL and RDF for further detail on these elements).

rdf:ID

- In an element that requires a resource to be identified, an *rdf:ID* attribute MAY be used. This attribute has slightly different behavior to *rdf:about*, including the requirement that the value only appear once in the scope of a document, so it provides a useful check when defining distinct resources. RDF and OWL use URIs to identify resources. The attribute value is a string indicating a URI fragment relative to the currently in-scope base value (typically set using the xml:base attribute).

rdf:about

- As an alternative to using an *rdf:ID* attribute to identify a resource, an *rdf:about* attribute MAY be used. The attribute value is either a string indicating an absolute URI, or a string indicating a relative URI resolved against the currently in-scope base value (typically set using the xml:base attribute).

owl:Ontology

- In OWL, classification systems are represented by an OWL ontology. The ontology groups a number of related classifications together. In the example the geographical regions classification system would be defined using an *owl:Ontology* element.

owl:Imports

- owl:Imports elements are permitted under owl:Ontology. This means that classes declared in the ontology MAY subclass classes declared in any imported ontologies, although multiple inheritance is not permitted. How owl ontologies are imported is vendor specific and therefore the resolution of owl:import references is also vendor specific.

owl:Class

- OWL represents classifications with OWL classes. In the example, the World, Europe, Asia, United Kingdom, Germany, Japan and China classifications would be defined using an *owl:Class* element.

rdfs:subClassOf

- To define the hierarchy, classes are related to each other via the *rdfs:subClassOf* element. If class B is declared to be a subclass of class A, then the instances of class B represent a subset of the instances of class A. In the example, Asia would be declared to be a subclass of World, Japan a subclass of Asia, and so on.

rdfs:label

- Ontologies and classes MAY be given a human readable name using the *rdfs:label* element. Names in multiple languages are supported by this element, using the xml:lang attribute.

rdfs:comment

- Ontologies and classes MAY be given a human readable comment or description using the *rdfs:comment* element. Comments in multiple languages are also supported by this element using the xml:lang attribute.

All other OWL elements are not supported (in terms of OWL Lite this means that property-related elements, and the *owl:equivalentClass*, *owl:imports*, *owl:intersectionOf*, and versioning elements are not supported).

OWL files used in S-RAMP MUST conform to the following rules, which result in ontologies that are self-contained in a single OWL file:

- There MUST be exactly one *owl:Ontology* element in any OWL file
- A class MUST only be defined in one ontology

The example ontology can be expressed in OWL as follows:

Example 2: An OWL Ontology

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE rdf:RDF [
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
  <!ENTITY owl "http://www.w3.org/2002/07/owl#">
  <!ENTITY ns_region "http://www.regions.com/geographicalregion">
]>

<rdf:RDF
  xmlns:xsd="&xsd;"
  xmlns:rdf="&rdf;"
  xmlns:rdfs="&rdfs;"
  xmlns:owl="&owl;"
  xmlns:ns_region="&ns_region;"
  xml:base="&ns_region;"
>

  <!-- ontology -->

  <owl:Ontology rdf:ID="">
    <rdfs:label>Geographical Regions</rdfs:label>
    <rdfs:comment>An ontology used to provide classifications describing geographical
regions.</rdfs:comment>
  </owl:Ontology>

  <!-- root class -->

  <owl:Class rdf:ID="World">
    <rdfs:label>World</rdfs:label>
    <rdfs:label xml:lang="en">World</rdfs:label>
    <rdfs:label xml:lang="fr">Monde</rdfs:label>
```



```
    </owl:Class>

    <!-- sub classes -->

    <owl:Class rdf:ID="Asia">
      <rdfs:subClassOf rdf:resource="&ns_region;#World"/>
      <rdfs:label>Asia</rdfs:label>
    </owl:Class>

    <owl:Class rdf:ID="Europe">
      <rdfs:subClassOf rdf:resource="&ns_region;#World"/>
      <rdfs:label>Europe</rdfs:label>
    </owl:Class>

    <!-- sub sub classes -->

    <owl:Class rdf:ID="Japan">
      <rdfs:subClassOf rdf:resource="&ns_region;#Asia"/>
      <rdfs:label>Japan</rdfs:label>
    </owl:Class>

    <owl:Class rdf:ID="China">
      <rdfs:subClassOf rdf:resource="&ns_region;#Asia"/>
      <rdfs:label>China</rdfs:label>
    </owl:Class>

    <owl:Class rdf:ID="UnitedKingdom">
      <rdfs:subClassOf rdf:resource="&ns_region;#Europe"/>
      <rdfs:label>United Kingdom</rdfs:label>
    </owl:Class>

    <owl:Class rdf:ID="Germany">
      <rdfs:subClassOf rdf:resource="&ns_region;#Europe"/>
      <rdfs:label>Germany</rdfs:label>
    </owl:Class>

  </rdf:RDF>
```

Some points of interest in the foregoing example:

- An *rdfs:comment* element is shown within the Geographical Regions *owl:ontology* element, but comments may be used in *owl:Class* constructs if desired.
- *rdfs:label* elements have generally been used with no xml:lang attribute specified. However, multiple *rdfs:label* elements MAY be specified within an *owl:Ontology* or *owl:Class* element, and these may contain a variety of xml:lang attribute values (and an *rdfs:label* element with no xml:lang attribute may be specified in addition to elements with such attributes present). The World class in the ontology listed in Example 2 illustrates this. An *owl:Ontology* or *owl:Class* element SHOULD contain only one *rdfs:label* element with a given xml:lang value to ensure well-defined behavior when requesting a label for a specific language.
- The behavior of *rdfs:comment* elements with respect to xml:lang attributes is identical to the behavior for *rdfs:label* elements.
- Although the example shows a single root class which the other classes subclass either directly or indirectly, multiple root classes are permitted in an ontology. Also permitted are stand-alone classes that neither subclasses some parent class nor are themselves parents to other child classes.
- Multiple inheritance (a class being a subclass of multiple parent classes) is NOT permitted.

4 S-RAMP Query Model

The S-RAMP specification supports a robust query interface that compliant implementations MUST support. It provides a way to find repository artifacts using a rich set of constraints. The query expression in S-RAMP uses an XPath 2.0 based dialect. Refer to Section **Error! Reference source not found.** for additional information on the S-RAMP query grammar. The expression predicates act as filters to identify matching artifact instances. Filtering can be done based upon artifact metadata, including properties, relationships and classifications. Complex criteria can be formed. One can for example, search for "all services which are categorized as being in production", or find "all the XML Schema documents which are referenced by any WSDL document", and so on. Queries are executed against a set of S-RAMP artifacts using the criteria provided and the result returned is a set of S-RAMP artifacts. Furthermore, all artifact types both concrete and abstract are eligible for use in the query expressions.

Specific request and response syntax for executing query requests is, however, binding specific. Details can be found in the relevant binding document(s) of this specification. This document covers the common features of an S-RAMP query, independent of their invocation syntax.

4.1 Query Dialect (XPath2) Context

Since the S-RAMP query model is based on XPath2 **[XPATH]**, this specification defines a static context for the information available during static analysis of a query expression prior to its evaluation, as well as a dynamic context for the information available when the expression is evaluated.

Static analysis of query expressions is performed using the following static context:

Table 8: Static Context for S-RAMP Query Expressions

Component	Value
XPath 1.0 Compatibility Mode	False
Statically known namespaces	See Table 4: Prefixes and XML Namespaces Used in this Specification
Default element/type namespaces	http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0
Default function namespace	http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0
In-scope schema definitions	S-RAMP schemas are defined at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=s-ramp. A copy is included in the Appendices of this document.
In-scope variables	Only the variables used in a template based query expression (typically applies to Stored Query)
Context item static type	element(http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0, record)
Function signatures	Functions defined in XQuery 1.0 and XPath 2.0 Functions and Operators document**[QUERYOPS]**, plus functions defined in Section Query Functions
Statically known collations	none
Default collation	http://www.w3.org/2005/xpath-functions/collation/codepoint
Base URI	`none`
Statically known documents	`none`

Statically known collections	`none`
Statically known default collection type	`none`

The following dynamic context defines the default values that are available during evaluation of an S-RAMP query expression:

Table 9: Dynamic Context for S-RAMP Query Expressions

Component	**Value**
context item	dynamic; node changes during evaluation of the expression
context position	`1`
context size	`1`
Variable values	`none`
Function implementations	function implementations are on a per-server basis
Current dateTime	date and time on server against which request is made
Implicit time zone	UTC+/-0
Available documents	`none`
Available collections	`none`
Default collection	“s-ramp”

The domain of an S-RAMP query is the structure of the data model defined by its schemas, using the Artifact Model and Artifact Type reference syntax described in **Error! Reference source not found.**.

Additional features and limitations of S-RAMP query support:

- Uses the XPath2 default Axis of ancestor / child
- The XPath abbreviated paths listed below are not supported:
  - ‘..’
- XPath2 Node types are not supported
  - Comments, text and so on are not relevant in an S-RAMP query

4.2 Query Expression Predicates

The default namespace assumed in an S-RAMP query is http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0. S-RAMP query expressions allow filtering of the path expression using XPath 2.0 predicates, in the form:

```
{path-expression}[{predicate}]
```

where:

```
{path-expression} = /s-ramp/{ArtifactModel}/{ArtifactType}
```


corresponds to the S-RAMP Artifact Model and Type reference syntax (see Table 5: Pre-Defined Artifact Type Models); and:

```
{predicate}
```

is an XPath2 predicate which conforms to the S-RAMP query grammar (see SectionQuery Grammar).

When evaluated, the predicate returns true or false, and thus filters the set of artifacts returned from the query. Predicates can use combinations of artifact properties, relationships and classifications. The predicate sets the context for evaluating the query, making it wider or narrower. If a query predicate uses a property which is not defined on an artifact instance whose scope is within the path-expression, then the predicate is false for that artifact and it is not returned from the query. Several examples follow:

Example 3: Query Expressions Using Properties

/s-ramp/xsd/XsdDocument[@someProperty]

- Returns *XsdDocument* artifact instances which have a generic property named "someProperty".

/s-ramp/xsd/XsdDocument[@name = 'bob']

- Returns *XsdDocument* artifact instances that have an artifact name whose value is "bob".

/s-ramp/serviceImplementation/ServiceInstance[@someProperty ='high']

- Returns *ServiceInstance* artifact instances having a generic property called *someProperty* whose value equals 'high'.
- If the requested property is not defined on the artifact within the set of artifacts selected for search, then the predicate is false and that artifact is therefore not returned.

Example 4: Query Expression Using Relationships

/s-ramp/wsdl/WsdlDocument[includedXsds]

- Returns all the *WsdlDocument* artifact instance(s) representing WSDL documents for which the query predicate evaluates to a non-empty context. The context here consists of the targets of the *includedXsds* relationship; therefore any *WsdlDocument* artifact that has at least one includedXsds relationship is returned.
- If the relationship in the predicate is not defined on a given artifact within the set of artifacts selected for search, then the predicate expression is false and that artifact is not returned.

Example 5: Query Expressions Using Relationships and Properties

/s-ramp/wsdl/WsdlDocument[includedXsds[@someProperty='true']]

- Only *WsdlDocument* artifact instances corresponding to WSDL documents which have at least one *includedXsds* relationship instance, that has a property name of "someProperty", whose value is "true" will be returned.
- If the relationship in the predicate is not defined on a given artifact whose scope is within the path expression, then the predicate evaluates to false and that artifact is not returned.

Example 6: Extended Artifacts

/s-ramp/ext/BpmnDocument[@name = 'LoanApproval']

- Returns only BpmnDocument extended artifact instances named 'LoanApproval'.

Example 7: Query Expressions Using Relationships as Sub-Artifact Sets

/s-ramp/wsdl/Message[@name='PurchaseRequestMessage']/part

- Returns all Part artifacts of the Message artifact named 'PurchaseRequestMessage'.

4.3 Query Functions

S-RAMP defines a number of its own query functions in addition to using some already defined in XPath 2.0 **[XPATH]**. This includes functions that provide several useful simplifications associated with how an artifact is classified. These query functions use the following syntax:

```
s-ramp:{function-name}({artifact}, {category-value-1}, {category-value-2}, …)
```

Table 10: Query Functions Used in S-RAMP below lists the functions used in S-RAMP. Each of the examples for the classification-based functions it describes assumes the following conceptual OWL ontology:

- class color
  - class red
  - class white
- class taste
  - class sour
  - class sweet
  - class spicy

where this ontology is used to classify a set of WsdlService artifact instances:

- name = "Bread" classifiedBy = ["white", "sweet"]
- name = "Wine" classifiedBy = ["red", "sweet"]
- name = "Chili" classifiedBy = ["red", "spicy"]

and each of the classifiedBy URI references to these will be abbreviated simply as the class name here for brevity:

Table 10: Query Functions Used in S-RAMP

Function	Descriptions & Examples
s-ramp:classifiedByAnyOf	Returns all artifact instances classified by at least one of the specified OWL URIs or their subtypes. `Syntax: classifiedByAnyOf({artifact}, {classifiedBy_1}, {classifiedBy_2},…)`

	Using the classification example setup preceding this table, this example returns all 3 *WsdlService* artifact instances:
	`/s-ramp/wsdl/WsdlService[classifiedByAnyOf(., 'taste')]`
	This example returns the Wine and Chili *WsdlService* artifact instances:
	`/s-ramp/wsdl/WsdlService[classifiedByAnyOf(., 'red', 'spicy')]`

s-ramp:classifiedByAllOf	Returns all artifact instances classified by every one of the specified OWL URIs or their subtypes.
	`Syntax: classifiedByAllOf({artifact}, {classifiedBy_1}, {classifiedBy_2},…)`
	This example returns all 3 *WsdlService* artifact instances:
	`/s-ramp/wsdl/WsdlService[classifiedByAllOf(., 'taste')]`
	This example returns only the Wine *WsdlService* artifact instance:
	`/s-ramp/wsdl/WsdlService[classifiedByAllOf(., 'red', 'sweet')]`

s-ramp:exactlyClassifiedByAnyOf	Returns all artifact instances classified by at least one of the specified OWL URIs. Subtypes are not considered:
	`Syntax: exactlyclassifiedByAnyOf({artifact}, {classifiedBy_1}, {classifiedBy_2},…)`
	This example returns none of the *WsdlService* artifact instances:
	`/s-ramp/wsdl/WsdlService[exactlyClassifiedByAnyOf(., 'taste')]`
	This example returns only the Wine and Chili *WsdlService* artifact instances:
	`/s-ramp/wsdl/WsdlService[exactlyClassifiedByAnyOf(., 'red', 'taste')]`

s-ramp:exactlyClassifiedByAllOf	Returns all artifact instances classified by every one of the specified OWL URIs. Subtypes are not considered:
	`Syntax: exactlyclassifiedByAllOf({artifact}, {classifiedBy_1}, {classifiedBy_2},…)`
	This example returns none of the *WsdlService* artifact instances:
	`/s-ramp/wsdl/WsdlService[exactlyClassifiedByAllOf(., 'taste')]`
	This example returns only the Wine *WsdlService* artifact instance:
	`/s-ramp/wsdl/WsdlService[exactlyClassifiedByAllOf(., 'red', 'sweet')]`

xp2:matches	This is the XPath 2.0 matches function, which returns an xsd:boolean indicating whether {value} of its first argument matches the regular {expression} that is its second argument.
	
	`Syntax: matches( {value}, {expression} )`
	
	The expression may use wildcards, but only in the form '.*'
	
	This example returns all artifact instances (of any Artifact Type) from the Service Implementation Model, whose artifact *name* matches "`.*account.*`", and a user-defined *version* property whose value is '1'.
	`/s-ramp/serviceImplementation[xp2:matches(@name, '.*account.*')`
	`and @version = '1']`

xp2:not	This is the XPath 2.0 "not" (negation) function, which accepts an xsd:boolean as its input and returns the inversion.
	
	`Syntax: not( {boolean-expression} )`
	
	Within the context of the S-RAMP query language, this function can only be applied within the predicate, and the 'boolean-expression' is restricted to a property or relationship. This effectively addresses the use-case of searching for artifacts that do not have a given property or relationship.
	
	`/s-ramp/core/Document[xp2:not(@user-property)]`
	`/s-ramp/wsdl/Part[xp2:not(element)]`
	
	Note: to invert a property value, simply use the '!=' operator rather than the 'not' function.

4.4 Query Grammar

This section describes the XPath2 based query grammar used in S-RAMP. It is based on a redacted subset of the XPath 2.0 grammar **[XPATH]**.

QName ::=

[http://www.w3.org/TR/REC-xml-names/#NT-QName]

s-ramp query ::=

artifact -set

| artifact -set '[' predicate '] |

| artifact -set '[' predicate ']' '/' subartifact-set

artifact- set::=

location-path

location-path::=

/s-ramp

|/s-ramp/<artifact-model>

| /s-ramp/<artifact-model>/<artifact-type>

| //<artifact-model>

| //<artifact-type>

subartifact-set:: =

relationship-path

| relationship-path '[' predicate ']'

| relationship-path '[' predicate ']' '/' subartifact-set

| FunctionCall

relationship-path::=

any-outgoing-relationship

| <s-ramp-relationship-type>

any-outgoing-relationship::=

outgoing

Predicate::=

Expr

Expr::=

AndExpr

AndExpr::=

OrExpr

| AndExpr 'and' OrExpr

OrExpr::=

EqualityExpr

| OrExpr 'or' EqualityExpr

EqualityExpr::=

subartifact-set

|ForwardPropertyStep

| ForwardPropertyStep '=' PrimaryExpr

| ForwardPropertyStep '!=' PrimaryExpr

| ForwardPropertyStep '<' PrimaryExpr

| ForwardPropertyStep '>' PrimaryExpr

| ForwardPropertyStep '<=' PrimaryExpr

| ForwardPropertyStep '>=' PrimaryExpr

| '(' Expr ')'

PropertyQName ::=

QName

PrimaryExpr ::=

Literal
| Number
|'$' <PropertyQName>

ForwardPropertyStep ::=

Subaartifact-set'/'@<PropertyQName>|@<PropertyQName>

FunctionCall ::=

FunctionName '(' (Argument (',' Argument)*)? ')'

Argument ::=

PrimaryExpr
| Expr

Literal ::=

'"' [^"]* '"' | "'" [^']* "'"

Number ::=

Digits ('.' Digits?)? | '.' Digits

Digits ::=

[0-9]+

FunctionName ::=

QName – NodeType

NodeType ::=

'comment'
| 'text'
| 'processing-instruction'
| 'node'

4.5 Stored Queries

S-RAMP provides support for storing queries in the repository using the *StoredQuery* Artifact Type. This can be convenient because it allows quick execution of a frequently performed query. The syntax associated with creation, retrieval, update and deletion of a Stored Query is binding specific. Refer to the appropriate binding document of this specification for details.

939 The *StoredQuery* Artifact Type does NOT extend *BaseArtifactType* as do most other Artifact Types in S-
940 RAMP, which means it is simpler and possesses only these built-in attributes:

941 • queryName: The name of the Stored Query instance. This must be unique.
942 • queryExpression: The specification of the query expression.
943
944 A *StoredQuery* MAY also contain a list of propertyName values. These are used to indicate to the server
945 that the results returned from the execution of the query SHALL include values for those property names
946 when they are present in the artifact instance(s) returned. This can be valuable in bindings that may not
947 necessarily return the complete artifact in query results. The actual format of the query response is
948 binding specific.

5 Conformance

5.1 Introduction

S-RAMP defines a data model and protocol for Service Repository implementations.

An implementation is not compliant with this specification if it fails to satisfy one or more of the MUST or REQUIRED level requirements defined herein.

The XML Schemas take precedence over the normative text within this specification that takes precedence over the S-RAMP XML Schema in the appendix of this document. The authoritative S-RAMP XML Schema is published at:

https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=s-ramp

This specification addresses the following aspects of conformance:

- Data Model
- Classification Systems
- Query Model

5.2 Data Model

A conformant implementation MUST include support for the following data models defined in Section 2 of this specification:

Note: "M" means mandatory-to-implement. "O" means optional.

Table 11: Required Data Models

Data Model	**M/O**
Core Model: Document	M
Core Model: XmlDocument	M
Extended Model: ExtendedArtifactType	M
Extended Model: ExtendedDocument	M
XSD Model: XsdDocument	M
XSD Model: ElementDeclaration	M
XSD Model: AttributeDeclaration	M
XSD Model: ComplexTypeDeclaration	M
XSD Model: SimpleTypeDeclaration	M
WSDL Model: WsdlDocument	M
WSDL Model: WsdlService	M
WSDL Model: Port	M

WSDL Model: Binding	M
WSDL Model: PortType	M
WSDL Model: BindingOperation	M
WSDL Model: BindingOperationInput	M
WSDL Model: BindingOperationFault	M
WSDL Model: BindingOperationOutput	M
WSDL Model: Operation	M
WSDL Model: OperationInput	M
WSDL Model: OperationOutput	M
WSDL Model: Fault	M
WSDL Model: Message	M
WSDL Model: Part	M
WSDL Model: WsdlExtension	O
SOAP WSDL Model: SoapAddress	M
SOAP WSDL Model: SoapBinding	M
Policy Model: PolicyDocument	M
Policy Model: PolicyAttachment	M
Policy Model: PolicyExpression	M
SOA Model: SoaModelType	M
SOA Model: ServiceInterface	M
SOA Model: Effect	M
SOA Model: Event	M
SOA Model: InformationType	M
SOA Model: Policy	M
SOA Model: PolicySubject	M
SOA Model: ServiceContract	M
SOA Model: Element	M
SOA Model: Service	M
SOA Model: Actor	M
SOA Model: Organization	M
SOA Model: System	M
SOA Model: Task	M
SOA Model: Composition	M

SOA Model: ServiceComposition	M
SOA Model: Choreography	M
SOA Model: Collaboration	M
SOA Model: Orchestration	M
SOA Model: Process	M
SOA Model: ChoreographyProcess	M
SOA Model: CollaborationProcess	M
SOA Model: OrchestrationProcess	M
Service Implementation Model: ServiceImplementationModelType	M
Service Implementation Model: ServiceInstance	M
Service Implementation Model: ServiceOperation	M
Service Implementation Model: ServiceEndpoint	M
Service Implementation Model: ServiceOperation	M

971

5.3 Classification Systems

973 A conformant implementation MAY support OWL Lite classification systems as defined in Classification
974 Systems in S-RAMP of this specification. If an implementation chooses not to support OWL Lite
975 classification systems, then it MUST fail if a value is added to the 'classifiedBy' attribute as defined in
976 Classifications.

5.4 Query Model

978 S-RAMP defines an XPath 2 based query model. The following table lists the features of the S-RAMP
979 Query Model and their conformance requirement:

980

981 Note: "M" means mandatory-to-implement. "O" means optional.

Query Model Feature	**Example**	**M/O**
All Artifact Query	/s-ramp	M
Filter by Artifact Model	/s-ramp/wsdl	M
Filter by Artifact Type	//ElementDeclaration	M
Filter by Both	/s-ramp/policy/PolicyAttachment	M
Filter by Core Attribute	//ElementDeclaration[@version='1.0']	M
Filter by Custom Attribute	//ElementDeclaration[@foo='bar']	M
Filter by Relationship	//ElementDeclaration[includedXsds]	M
Filter by Relationship Target	//Part[includedXsds[@name='xyz.xsd']]	M
Negation by Attribute	//ElementDeclaration[not(@foo)]	M

Negation by Relationship	//Part[not(element)]	M
Function: matches()	//Part[xp2:matches(@name, 'foo.*')	M
Function: s-ramp:classifiedByAnyOf	//Document[s-ramp:classifiedByAnyOf(., 'spicy')]	O
Function: s-ramp:classifiedByAllOf	//Document[s-ramp:classifiedByAllOf(., 'spicy')]	O
Function: s-ramp:exactlyClassifiedByAnyOf	//Document[s-ramp:exactlyClassifiedByAnyOf(., 'spicy')]	O
Function: s-ramp:exactlyClassifiedByAllOf	//Document[s-ramp:exactlyClassifiedByAllOf(., 'spicy')]	O
Boolean Filters	//Part[@name='invoice' and @prop1='val1']	M
Sub-Artifact Sets	//Message[@name='customer']/part	O

982
983
984
985

Appendix A. Acknowledgments

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:

Joel Fleck II, Hewlett-Packard
Jishnu Mukerji, Hewlett-Packard
Radek Pospisil, Hewlett-Packard
Vincent Brunssen, IBM
John Colgrave, IBM
Diane Jordan, IBM
Bernard Kufluk, IBM
Kelvin Lawrence, IBM
Martin Smithson, IBM
Gershon Janssen, Individual
Carl Mattocks, Individual
Rex Brooks, Network Centric Operations Industry Consortium
jian zhang, Primeton Technologies, Inc.
Randall Hauch, Red Hat
Kurt Stam, Red Hat
Eric Wittmann, Red Hat
Steve Fanshier, Software AG, Inc.
Gary Woods, Software AG, Inc.
Prasad Yendluri, Software AG, Inc.
Eric Johnson, TIBCO Software Inc.
Senaka Fernando, WSO2
Paul Fremantle, WSO2
Jonathan Marsh, WSO2

Appendix B. Non-Normative Text

This specification provides no additional non-normative information at this time. It is expected that best practices for S-RAMP Repositories will emerge over time as this specification is adopted by vendors and users.

Appendix C. Revision History

Revision	Date	Editor	Changes Made
01	2012-11-30	Kurt Stam	S-RAMP JIRA issues 1, 18
02	2012-11-30	Eric Wittmann	S-RAMP JIRA issue 45
03	2012-12-01	Eric Wittmann	S-RAMP JIRA issue 38
04	2012-12-04	Eric Wittmann	S-RAMP JIRA issue 2
05	2012-12-04	Eric Wittmann	S-RAMP JIRA issue 15
06	2012-12-04	Eric Wittmann	S-RAMP JIRA issue 25
07	2013-01-09	Eric Wittmann	S-RAMP JIRA issue 46
08	2013-01-24	Eric Wittmann	S-RAMP JIRA issues 30, 40, 41
09	2013-02-04	Eric Wittmann	Update document to latest OASIS template
10	2013-02-11	Eric Wittmann	Added conformance section (Section 5) – S-RAMP JIRA issue 27
11	2013-02-18	Eric Wittmann	Changes based on review of the Foundation document, preparing for submission.
12	2013-02-20	Kurt Stam	Updated S-RAMP figures with latest generated from XMLSpy.
13	2013-02-20	Eric Wittmann	S-RAMP JIRA issue 48, namespace find/replace.
14	2013-03-06	Eric Wittmann	Negation support in the S-RAMP Query Language
15	2013-03-12	Kurt Stam	S-RAMP JIRA issue 51, Adding ExtendedDocument

1020 # Appendix D. Glossary

Term	Definition
Artifact Type	The data type of an S-RAMP artifact.
Artifact Type Model	Grouping of similar S-RAMP Artifact Types (e.g. wsdl, policy).
Service Implementation Model	Set of S-RAMP Artifact Types and relationships that describe the service implementation layer associated with the SOA Model.
Core Model	Set of basic Artifact Types.
Policy Model	Set of Policy document related derived Artifact Types.
Relationship	The logical triple of a Relationship Type, Source and Target. Relationships in S-RAMP are all directed from a source, to a target.
Relationship Type	A name that represents the type of the relationship (e.g., “includedXsds”). Multiple relationships can share the same Relationship Type.
SOA Model	Set of Artifact Types and relationships used to link The Open Group’s SOA Ontology artifact types with those in the S-RAMP data model.
WSDL Model	Set of WSDL document related derived data types.
XSD Model	Set of XSD document related derived data types.
Extended Artifact Model	An S-RAMP model whose content and structure has been defined by the client.

1021

Appendix E. Core Model Schema

The S-RAMP Core Model Schema XSD file is also provided at:

http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0/

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
version="1.0" xmlns:tns="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0" xmlns:s-
ramp="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xlink="http://www.w3.org/1999/xlink" elementFormDefault="qualified">
  <!--
      (c) 2010 Hewlett-Packard Company (HP), International Business Machines
      Corporation (IBM), Software AG (SAG) and TIBCO Software Inc. All
      rights reserved. Permission to copy and display the SOA Repository
      Artifact Model and Protocol (the “Specification”), in any medium
      without fee or royalty is hereby granted by Hewlett-Packard Company
      (HP), International Business Machines Corporation (IBM), Software AG
      (SAG) and TIBCO Software Inc. (collectively, the "Authors"), provided
      that you include the following on ALL copies of this document or
      portions thereof, that you make:

      1. A link or URL to this document at this location:
      http://s-ramp.org/2010/s-ramp/specification/documents/{this document
      name}
      2. The copyright notice as shown in the Specification.

      The Authors each agree to grant you a royalty-free license, under
      reasonable, non-discriminatory terms and conditions to their
      respective patents that they deem necessary to implement the "SOA
      Repository Artifact Model and Protocol” Specification, including all
      its constituent documents. THIS DOCUMENT IS PROVIDED "AS IS," AND THE
      AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED,
      INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS
      FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE
      CONTENTS OF THIS DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE
      IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY
      PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. THE AUTHORS WILL NOT
      BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR
      CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR
      DISTRIBUTION OF THIS DOCUMENT.
  -->

  <xsd:import namespace="http://www.w3.org/1999/xlink" schemaLocation="xlink.xsd" />

  <!-- Core data types: -->
  <xsd:simpleType name="baseArtifactEnum">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="BaseArtifactType" />
      <xsd:enumeration value="DocumentArtifactType" />
      <xsd:enumeration value="Document" />
      <xsd:enumeration value="XmlDocument" />
      <!-- xsd model -->
```



```
        <xsd:enumeration value="XsdDocument" />
        <!-- wsdl model -->
        <xsd:enumeration value="WsdlDocument" />
        <!-- policy model -->
        <xsd:enumeration value="PolicyDocument" />
        <!-- derived artifacts -->
        <xsd:enumeration value="DerivedArtifactType" />
        <!-- policy model -->
        <xsd:enumeration value="PolicyAttachment" />
        <xsd:enumeration value="PolicyExpression" />
        <!-- xsd model -->
        <xsd:enumeration value="AttributeDeclaration" />
        <xsd:enumeration value="ElementDeclaration" />
        <xsd:enumeration value="XsdType" />
        <xsd:enumeration value="ComplexTypeDeclaration" />
        <xsd:enumeration value="SimpleTypeDeclaration" />
        <!-- wsdl model -->
        <xsd:enumeration value="WsdlDerivedArtifactType" />
        <xsd:enumeration value="NamedWsdlDerivedArtifactType" />
        <xsd:enumeration value="WsdlService" />
        <xsd:enumeration value="Port" />
        <xsd:enumeration value="Binding" />
        <xsd:enumeration value="PortType" />
        <xsd:enumeration value="BindingOperation" />
        <xsd:enumeration value="BindingOperationInput" />
        <xsd:enumeration value="BindingOperationFault" />
        <xsd:enumeration value="Operation" />
        <xsd:enumeration value="OperationInput" />
        <xsd:enumeration value="Fault" />
        <xsd:enumeration value="Message" />
        <xsd:enumeration value="Part" />
        <xsd:enumeration value="BindingOperationOutput" />
        <xsd:enumeration value="OperationOutput" />
        <xsd:enumeration value="WsdlExtension" />
        <!-- soapwsdl model -->
        <xsd:enumeration value="SoapAddress" />
        <xsd:enumeration value="SoapBinding" />
        <!-- extended artifacts -->
        <xsd:enumeration value="ExtendedArtifactType" />
        <xsd:enumeration value="ExtendedDocument" />
        <!-- soa model -->
        <xsd:enumeration value="SoaModelType" />
        <xsd:enumeration value="ServiceInterface" />
        <xsd:enumeration value="Effect" />
        <xsd:enumeration value="Event" />
        <xsd:enumeration value="InformationType" />
        <xsd:enumeration value="Policy" />
        <xsd:enumeration value="PolicySubject" />
        <xsd:enumeration value="ServiceContract" />
        <xsd:enumeration value="Element" />
        <xsd:enumeration value="Service" />
        <xsd:enumeration value="Actor" />
        <xsd:enumeration value="Organization" />
        <xsd:enumeration value="System" />
        <xsd:enumeration value="Task" />
        <xsd:enumeration value="Composition" />
```



```
        <xsd:enumeration value="ServiceComposition" />
        <xsd:enumeration value="Choreography" />
        <xsd:enumeration value="Collaboration" />
        <xsd:enumeration value="Orchestration" />
        <xsd:enumeration value="Process" />
        <xsd:enumeration value="ChoreographyProcess" />
        <xsd:enumeration value="CollaborationProcess" />
        <xsd:enumeration value="OrchestrationProcess" />
        <!-- serviceimplementation model -->
        <xsd:enumeration value="ServiceImplementationModelType" />
        <xsd:enumeration value="ServiceInstance" />
        <xsd:enumeration value="ServiceOperation" />
        <xsd:enumeration value="ServiceEndpoint" />
        <xsd:enumeration value="ServiceOperation" />
      </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="derivedArtifactEnum">
      <xsd:restriction base="s-ramp:baseArtifactEnum">
        <xsd:enumeration value="DerivedArtifactType" />
        <!-- policy model -->
        <xsd:enumeration value="PolicyAttachment" />
        <xsd:enumeration value="PolicyExpression" />
        <!-- xsd model -->
        <xsd:enumeration value="AttributeDeclaration" />
        <xsd:enumeration value="ElementDeclaration" />
        <xsd:enumeration value="XsdType" />
        <xsd:enumeration value="ComplexTypeDeclaration" />
        <xsd:enumeration value="SimpleTypeDeclaration" />
        <!-- wsdl model -->
        <xsd:enumeration value="WsdlDerivedArtifactType" />
        <xsd:enumeration value="NamedWsdlDerivedArtifactType" />
        <xsd:enumeration value="WsdlService" />
        <xsd:enumeration value="Port" />
        <xsd:enumeration value="Binding" />
        <xsd:enumeration value="PortType" />
        <xsd:enumeration value="BindingOperation" />
        <xsd:enumeration value="BindingOperationInput" />
        <xsd:enumeration value="BindingOperationFault" />
        <xsd:enumeration value="Operation" />
        <xsd:enumeration value="OperationInput" />
        <xsd:enumeration value="Fault" />
        <xsd:enumeration value="Message" />
        <xsd:enumeration value="Part" />
        <xsd:enumeration value="BindingOperationOutput" />
        <xsd:enumeration value="OperationOutput" />
        <xsd:enumeration value="WsdlExtension" />
        <!-- soapwsdl model -->
        <xsd:enumeration value="SoapAddress" />
        <xsd:enumeration value="SoapBinding" />
      </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="documentArtifactEnum">
      <xsd:restriction base="s-ramp:baseArtifactEnum">
        <xsd:enumeration value="DocumentArtifactType" />
```



```
        <xsd:enumeration value="Document" />
        <xsd:enumeration value="XmlDocument" />
        <xsd:enumeration value="ExtendedDocument" />
        <!-- xsd model -->
        <xsd:enumeration value="XsdDocument" />
        <!-- wsdl model -->
        <xsd:enumeration value="WsdlDocument" />
        <!-- policy model -->
        <xsd:enumeration value="PolicyDocument" />
      </xsd:restriction>
    </xsd:simpleType>

    <!-- Base type for almost all Artifacts in S-RAMP. Most other types in S-RAMP
  extend this one. Extensions of BaseArtifactType are limited to attributes. -->
    <xsd:complexType abstract="true" name="BaseArtifactType">
      <xsd:sequence>
        <xsd:element ref="tns:classifiedBy" minOccurs="0" maxOccurs="unbounded" />
        <xsd:element ref="tns:relationship" minOccurs="0" maxOccurs="unbounded" />
        <xsd:element ref="tns:property" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
      <xsd:attribute name="artifactType" type="s-ramp:baseArtifactEnum" use="required"
  />
      <xsd:attribute name="name" type="xsd:string" use="required" />
      <xsd:attribute name="description" type="xsd:string" use="optional" />
      <xsd:attribute name="createdBy" type="xsd:string" use="required" />
      <xsd:attribute name="version" type="xsd:string" use="optional" />
      <xsd:attribute name="uuid" type="xsd:string" use="required" />
      <xsd:attribute name="createdTimestamp" type="xsd:dateTime" use="required" />
      <xsd:attribute name="lastModifiedTimestamp" type="xsd:dateTime" use="required" />
      <xsd:attribute name="lastModifiedBy" type="xsd:string" use="required" />
      <xsd:anyAttribute namespace="##any" />
    </xsd:complexType>

    <!-- Base type for all Derived Artifacts in S-RAMP -->
    <xsd:complexType abstract="true" name="DerivedArtifactType">
      <xsd:complexContent>
        <xsd:extension base="s-ramp:BaseArtifactType">
          <xsd:sequence>
            <xsd:element name="relatedDocument" type="tns:documentArtifactTarget"
  minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <xsd:element name="DerivedArtifactType" type="tns:DerivedArtifactType" />

    <!-- Base type for all Documents in S-RAMP -->
    <xsd:complexType abstract="true" name="DocumentArtifactType">
      <xsd:complexContent>
        <xsd:extension base="s-ramp:BaseArtifactType">
          <xsd:attribute name="contentType" type="xsd:string" />
          <xsd:attribute name="contentSize" type="xsd:long" />
          <xsd:attribute name="contentHash" type="xsd:string" />
          <xsd:anyAttribute namespace="##any" />
        </xsd:extension>
      </xsd:complexContent>
```



```
  </xsd:complexType>

  <!-- Document type implements DocumentArtifactType -->
  <xsd:complexType name="Document">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:DocumentArtifactType" />
    </xsd:complexContent>
  </xsd:complexType>

  <!-- Base Type for all XML documents. Specific document types extend XmlDocument --
>
  <xsd:complexType name="XmlDocument">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:DocumentArtifactType">
        <xsd:attribute name="contentEncoding" type="xsd:string" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- Types used for Extended Artifact Models in S-RAMP -->
  <xsd:complexType name="ExtendedArtifactType">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:BaseArtifactType">
        <xsd:attribute name="extendedType" type="xsd:string" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ExtendedDocument">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:DocumentArtifactType">
        <xsd:attribute name="extendedType" type="xsd:string" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- Relationship target artifact's UUID. Used by all types of relationships -->
  <xsd:complexType name="target">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute ref="xlink:href" use="required" />
        <xsd:anyAttribute namespace="##any" />
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <!-- Relationship referencing the artifact's UUID, to reference any BaseArtifact. -
->
  <xsd:complexType name="baseArtifactTarget">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:target">
        <xsd:attribute name="artifactType" type="s-ramp:baseArtifactEnum"
use="required" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

```



```
  <!-- Relationship referencing the artifact's UUID, to reference any
DerivedArtifact. -->
  <xsd:complexType name="derivedArtifactTarget">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:target">
        <xsd:attribute name="artifactType" type="s-ramp:derivedArtifactEnum"
use="required" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- Relationship referencing the artifact's UUID, to reference any
DocumentArtifact. -->
  <xsd:complexType name="documentArtifactTarget">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:target">
        <xsd:attribute name="artifactType" type="s-ramp:documentArtifactEnum"
use="required" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- Global Element Declarations -->

  <!-- Relationship element used for all GENERIC (user-defined) Relationships -->
  <xsd:element name="relationship">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="tns:relationshipType" minOccurs="1" maxOccurs="1" />
        <xsd:element name="relationshipTarget" type="tns:target" minOccurs="0"
maxOccurs="unbounded" />
      </xsd:sequence>
      <xsd:anyAttribute namespace="##any" />
    </xsd:complexType>
  </xsd:element>

  <!-- Stored Queries Artifact element -->
  <xsd:element name="StoredQuery">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="tns:propertyName" minOccurs="0" maxOccurs="unbounded" />
        <xsd:element ref="tns:queryExpression" />
      </xsd:sequence>
      <xsd:attribute name="queryName" type="xsd:string" />
      <xsd:anyAttribute namespace="##any" />
    </xsd:complexType>
  </xsd:element>

  <!-- Property -->
  <xsd:element name="property">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="tns:propertyName" minOccurs="1" maxOccurs="1" />
        <xsd:element ref="tns:propertyValue" minOccurs="1" maxOccurs="1" />
      </xsd:sequence>
      <xsd:anyAttribute namespace="##any" />
```



```
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="relationshipType" type="xsd:string" />
    <xsd:element name="classifiedBy" type="xsd:anyURI" />
    <xsd:element name="propertyName" type="xsd:string" />
    <xsd:element name="propertyValue" type="xsd:string" />
    <xsd:element name="queryExpression" type="xsd:string" />
    <!-- The sourceId and targetId contain the UUID's corresponding to a relationship's
  source and target -->
    <xsd:element name="sourceId" type="xsd:string" />
    <xsd:element name="targetId" type="xsd:string" />

  </xsd:schema>
```


Appendix F. SOA Model Schema

The S-RAMP SOA Model Schema XSD file is also provided at:

http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0/

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:tns="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0" xmlns:s-
ramp="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
elementFormDefault="qualified" version="1.0">
  <!--
      (c) 2010 Hewlett-Packard Company (HP), International Business Machines
      Corporation (IBM), Software AG (SAG) and TIBCO Software Inc. All
      rights reserved. Permission to copy and display the SOA Repository
      Artifact Model and Protocol (the “Specification”), in any medium
      without fee or royalty is hereby granted by Hewlett-Packard Company
      (HP), International Business Machines Corporation (IBM), Software AG
      (SAG) and TIBCO Software Inc. (collectively, the "Authors"), provided
      that you include the following on ALL copies of this document or
      portions thereof, that you make:

      1. A link or URL to this document at this location:
      http://s-ramp.org/2010/s-ramp/specification/documents/{this document
      name}
      2. The copyright notice as shown in the Specification.

      The Authors each agree to grant you a royalty-free license, under
      reasonable, non-discriminatory terms and conditions to their
      respective patents that they deem necessary to implement the "SOA
      Repository Artifact Model and Protocol” Specification, including all
      its constituent documents. THIS DOCUMENT IS PROVIDED "AS IS," AND THE
      AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED,
      INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS
      FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE
      CONTENTS OF THIS DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE
      IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY
      PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. THE AUTHORS WILL NOT
      BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR
      CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR
      DISTRIBUTION OF THIS DOCUMENT.
  -->

  <xsd:include schemaLocation="coremodel.xsd" />
  <xsd:include schemaLocation="serviceimplementationmodel.xsd" />

  <xsd:simpleType name="informationTypeEnum">
    <xsd:restriction base="s-ramp:baseArtifactEnum">
      <xsd:enumeration value="InformationType" />
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="serviceEnum">
    <xsd:restriction base="s-ramp:baseArtifactEnum">
```



```
        <xsd:enumeration value="Service" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="serviceContractEnum">
      <xsd:restriction base="s-ramp:baseArtifactEnum">
        <xsd:enumeration value="ServiceContract" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="serviceInterfaceEnum">
      <xsd:restriction base="s-ramp:baseArtifactEnum">
        <xsd:enumeration value="ServiceInterface" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="policySubjectEnum">
      <xsd:restriction base="s-ramp:baseArtifactEnum">
        <xsd:enumeration value="PolicySubject" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="taskEnum">
      <xsd:restriction base="s-ramp:baseArtifactEnum">
        <xsd:enumeration value="Task" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="policyEnum">
      <xsd:restriction base="s-ramp:baseArtifactEnum">
        <xsd:enumeration value="Policy" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="elementEnum">
      <xsd:restriction base="s-ramp:baseArtifactEnum">
        <xsd:enumeration value="Element" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="orchestrationEnum">
      <xsd:restriction base="s-ramp:baseArtifactEnum">
        <xsd:enumeration value="Orchestration" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="orchestrationProcessEnum">
      <xsd:restriction base="s-ramp:baseArtifactEnum">
        <xsd:enumeration value="OrchestrationProcess" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="eventEnum">
      <xsd:restriction base="s-ramp:baseArtifactEnum">
        <xsd:enumeration value="Event" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="actorEnum">
      <xsd:restriction base="s-ramp:baseArtifactEnum">
        <xsd:enumeration value="Actor" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="effectEnum">
      <xsd:restriction base="s-ramp:baseArtifactEnum">
        <xsd:enumeration value="Effect" />
```



```
      </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType abstract="true" name="SoaModelType">
      <xsd:complexContent>
        <xsd:extension base="s-ramp:BaseArtifactType">
          <xsd:sequence>
            <!-- Modeled Relationship to abstract DocumentArtifactType -->
            <xsd:element minOccurs="0" name="documentation" type="s-
ramp:documentArtifactTarget" maxOccurs="unbounded" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="ServiceInterface">
      <xsd:complexContent>
        <xsd:extension base="tns:SoaModelType">
          <xsd:sequence>
            <!-- Modeled Relationship to abstract DerivedArtifactType: -->
            <xsd:element minOccurs="0" maxOccurs="1" name="interfaceDefinedBy" type="s-
ramp:DerivedArtifactType" />
            <!-- Modeled Relationship to ServiceOperation: -->
            <xsd:element minOccurs="0" name="hasOperation" type="s-
ramp:serviceOperationTarget" />
            <!-- Modeled Relationship to InformationType: -->
            <xsd:element minOccurs="0" name="hasOutput" type="s-
ramp:informationTypeTarget" maxOccurs="unbounded" />
            <!-- Modeled Relationship to InformationType: -->
            <xsd:element minOccurs="0" name="hasInput" type="s-
ramp:informationTypeTarget" maxOccurs="unbounded" />
            <!-- Modeled Relationship to Service: -->
            <xsd:element minOccurs="0" name="isInterfaceOf" type="s-ramp:serviceTarget"
maxOccurs="unbounded" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="Service">
      <xsd:complexContent>
        <xsd:extension base="tns:Element">
          <xsd:sequence>
            <!-- Modeled Relationship to ServiceContract: -->
            <xsd:element minOccurs="0" name="hasContract" type="s-
ramp:serviceContractTarget" maxOccurs="unbounded" />
            <!-- Modeled Relationship to ServiceInterface: -->
            <xsd:element minOccurs="1" name="hasInterface" type="s-
ramp:serviceInterfaceTarget" maxOccurs="unbounded" />
            <!-- Modeled Relationship to ServiceInstance: -->
            <xsd:element minOccurs="0" maxOccurs="1" name="hasInstance" type="s-
ramp:serviceInstanceTarget" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
```



```
1533
1534      <xsd:complexType name="Effect">
1535        <xsd:complexContent>
1536          <xsd:extension base="tns:SoaModelType" />
1537        </xsd:complexContent>
1538      </xsd:complexType>
1539
1540      <xsd:complexType name="Event">
1541        <xsd:complexContent>
1542          <xsd:extension base="tns:SoaModelType" />
1543        </xsd:complexContent>
1544      </xsd:complexType>
1545
1546      <xsd:complexType name="InformationType">
1547        <xsd:complexContent>
1548          <xsd:extension base="tns:SoaModelType" />
1549        </xsd:complexContent>
1550      </xsd:complexType>
1551
1552      <xsd:complexType name="Policy">
1553        <xsd:complexContent>
1554          <xsd:extension base="tns:SoaModelType">
1555            <xsd:sequence>
1556              <!-- Modeled Relationship to PolicySubject: -->
1557              <xsd:element minOccurs="0" name="appliesTo" type="s-
1558    ramp:policySubjectTarget" maxOccurs="unbounded" />
1559            </xsd:sequence>
1560          </xsd:extension>
1561        </xsd:complexContent>
1562      </xsd:complexType>
1563
1564      <xsd:complexType name="PolicySubject">
1565        <xsd:complexContent>
1566          <xsd:extension base="tns:SoaModelType" />
1567        </xsd:complexContent>
1568      </xsd:complexType>
1569
1570      <xsd:complexType name="Actor">
1571        <xsd:complexContent>
1572          <xsd:extension base="tns:Element">
1573            <xsd:sequence>
1574              <!-- Modeled Relationship to Task: -->
1575              <xsd:element minOccurs="0" name="does" type="s-ramp:taskTarget"
1576    maxOccurs="unbounded" />
1577              <!-- Modeled Relationship to Policy: -->
1578              <xsd:element minOccurs="0" name="setsPolicy" type="s-ramp:policyTarget"
1579    maxOccurs="unbounded" />
1580            </xsd:sequence>
1581          </xsd:extension>
1582        </xsd:complexContent>
1583      </xsd:complexType>
1584
1585      <xsd:complexType name="Element">
1586        <xsd:complexContent>
1587          <xsd:extension base="tns:PolicySubject">
1588            <xsd:sequence>
```



```
1589            <!-- Modeled Relationship to Element: -->
1590            <xsd:element minOccurs="0" name="represents" type="s-ramp:elementTarget"
1591    maxOccurs="unbounded" />
1592            <!-- Modeled Relationship to Element: -->
1593            <xsd:element minOccurs="0" name="uses" type="s-ramp:elementTarget"
1594    maxOccurs="unbounded" />
1595            <!-- Modeled Relationship to Service: -->
1596            <xsd:element minOccurs="0" name="performs" type="s-ramp:serviceTarget"
1597    maxOccurs="unbounded" />
1598            <!-- Modeled Relationship to Orchestration: -->
1599            <xsd:element minOccurs="0" name="directsOrchestration" type="s-
1600    ramp:orchestrationTarget" maxOccurs="1" />
1601            <!-- Modeled Relationship to OrchestrationProcess: -->
1602            <xsd:element minOccurs="0" name="directsOrchestrationProcess" type="s-
1603    ramp:orchestrationTarget" maxOccurs="1" />
1604            <!-- Modeled Relationship to Event: -->
1605            <xsd:element minOccurs="0" name="generates" type="s-ramp:eventTarget"
1606    maxOccurs="unbounded" />
1607            <!-- Modeled Relationship to Event: -->
1608            <xsd:element minOccurs="0" name="respondsTo" type="s-ramp:eventTarget"
1609    maxOccurs="unbounded" />
1610          </xsd:sequence>
1611        </xsd:extension>
1612      </xsd:complexContent>
1613    </xsd:complexType>
1614
1615    <xsd:complexType name="ServiceContract">
1616      <xsd:complexContent>
1617        <xsd:extension base="s-ramp:PolicySubject">
1618          <xsd:sequence>
1619            <!-- Modeled Relationship to Actor: -->
1620            <xsd:element minOccurs="0" name="involvesParty" type="s-ramp:actorTarget"
1621    maxOccurs="unbounded" />
1622            <!-- Modeled Relationship to Effect: -->
1623            <xsd:element minOccurs="1" name="specifies" type="s-ramp:effectTarget"
1624    maxOccurs="unbounded" />
1625          </xsd:sequence>
1626        </xsd:extension>
1627      </xsd:complexContent>
1628    </xsd:complexType>
1629
1630    <xsd:complexType name="System">
1631      <xsd:complexContent>
1632        <xsd:extension base="tns:Element" />
1633      </xsd:complexContent>
1634    </xsd:complexType>
1635
1636    <xsd:complexType name="Composition">
1637      <xsd:complexContent>
1638        <xsd:extension base="tns:System" />
1639      </xsd:complexContent>
1640    </xsd:complexType>
1641
1642    <xsd:complexType name="Choreography">
1643      <xsd:complexContent>
1644        <xsd:extension base="tns:Composition" />
```



```
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="Collaboration">
      <xsd:complexContent>
        <xsd:extension base="tns:Composition" />
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="Orchestration">
      <xsd:complexContent>
        <xsd:extension base="tns:Composition" />
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="Process">
      <xsd:complexContent>
        <xsd:extension base="tns:Composition" />
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="ChoreographyProcess">
      <xsd:complexContent>
        <xsd:extension base="tns:Process" />
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="CollaborationProcess">
      <xsd:complexContent>
        <xsd:extension base="tns:Process" />
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="OrchestrationProcess">
      <xsd:complexContent>
        <xsd:extension base="tns:Process" />
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="Task">
      <xsd:complexContent>
        <xsd:extension base="tns:Element" />
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="ServiceComposition">
      <xsd:complexContent>
        <xsd:extension base="tns:Composition" />
      </xsd:complexContent>
    </xsd:complexType>

    <!-- Relationship referencing the artifact's UUID, to reference an InformationType.
  -->
    <xsd:complexType name="informationTypeTarget">
      <xsd:complexContent>
        <xsd:extension base="s-ramp:target">
```



```
        <xsd:attribute name="artifactType" type="s-ramp:informationTypeEnum"
use="required" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <!-- Relationship referencing the artifact's UUID, to reference a Service. -->
  <xsd:complexType name="serviceTarget">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:target">
        <xsd:attribute name="artifactType" type="s-ramp:serviceEnum" use="required"
/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <!-- Relationship referencing the artifact's UUID, to reference a ServiceContract.
-->
  <xsd:complexType name="serviceContractTarget">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:target">
        <xsd:attribute name="artifactType" type="s-ramp:serviceContractEnum"
use="required" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <!-- Relationship referencing the artifact's UUID, to reference a ServiceInterface.
-->
  <xsd:complexType name="serviceInterfaceTarget">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:target">
        <xsd:attribute name="artifactType" type="s-ramp:serviceInterfaceEnum"
use="required" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <!-- Relationship referencing the artifact's UUID, to reference a PolicySubject. --
>
  <xsd:complexType name="policySubjectTarget">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:target">
        <xsd:attribute name="artifactType" type="s-ramp:policySubjectEnum"
use="required" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <!-- Relationship referencing the artifact's UUID, to reference a Task. -->
  <xsd:complexType name="taskTarget">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:target">
        <xsd:attribute name="artifactType" type="s-ramp:taskEnum" use="required" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <!-- Relationship referencing the artifact's UUID, to reference a Policy. -->
  <xsd:complexType name="policyTarget">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:target">
```



```
          <xsd:attribute name="artifactType" type="s-ramp:policyEnum" use="required" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <!-- Relationship referencing the artifact's UUID, to reference an Element. -->
    <xsd:complexType name="elementTarget">
      <xsd:complexContent>
        <xsd:extension base="s-ramp:target">
          <xsd:attribute name="artifactType" type="s-ramp:elementEnum" use="required"
/>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <!-- Relationship referencing the artifact's UUID, to reference an Orchestration. -
->
    <xsd:complexType name="orchestrationTarget">
      <xsd:complexContent>
        <xsd:extension base="s-ramp:target">
          <xsd:attribute name="artifactType" type="s-ramp:orchestrationEnum"
use="required" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <!-- Relationship referencing the artifact's UUID, to reference an
OrchestrationProcess. -->
    <xsd:complexType name="orchestrationProcessTarget">
      <xsd:complexContent>
        <xsd:extension base="s-ramp:target">
          <xsd:attribute name="artifactType" type="s-ramp:orchestrationProcessEnum"
use="required" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <!-- Relationship referencing the artifact's UUID, to reference an Event. -->
    <xsd:complexType name="eventTarget">
      <xsd:complexContent>
        <xsd:extension base="s-ramp:target">
          <xsd:attribute name="artifactType" type="s-ramp:eventEnum" use="required" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <!-- Relationship referencing the artifact's UUID, to reference an Actor. -->
    <xsd:complexType name="actorTarget">
      <xsd:complexContent>
        <xsd:extension base="s-ramp:target">
          <xsd:attribute name="artifactType" type="s-ramp:actorEnum" use="required" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <!-- Relationship referencing the artifact's UUID, to reference an Effect. -->
    <xsd:complexType name="effectTarget">
      <xsd:complexContent>
        <xsd:extension base="s-ramp:target">
          <xsd:attribute name="artifactType" type="s-ramp:effectEnum" use="required" />
        </xsd:extension>
      </xsd:complexContent>
```



```
    </xsd:complexType>
</xsd:schema>
```


Appendix G. Service Implementation Model Schema

The S-RAMP Service Implementation Model Schema XSD file is also provided at:

http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0/

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:tns="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0" xmlns:s-
ramp="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
elementFormDefault="qualified" version="1.0">
  <!--
      (c) 2010 Hewlett-Packard Company (HP), International Business Machines
      Corporation (IBM), Software AG (SAG) and TIBCO Software Inc. All
      rights reserved. Permission to copy and display the SOA Repository
      Artifact Model and Protocol (the “Specification”), in any medium
      without fee or royalty is hereby granted by Hewlett-Packard Company
      (HP), International Business Machines Corporation (IBM), Software AG
      (SAG) and TIBCO Software Inc. (collectively, the "Authors"), provided
      that you include the following on ALL copies of this document or
      portions thereof, that you make:

      1. A link or URL to this document at this location:
      http://s-ramp.org/2010/s-ramp/specification/documents/{this document
      name}
      2. The copyright notice as shown in the Specification.

      The Authors each agree to grant you a royalty-free license, under
      reasonable, non-discriminatory terms and conditions to their
      respective patents that they deem necessary to implement the "SOA
      Repository Artifact Model and Protocol” Specification, including all
      its constituent documents. THIS DOCUMENT IS PROVIDED "AS IS," AND THE
      AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED,
      INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS
      FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE
      CONTENTS OF THIS DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE
      IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY
      PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. THE AUTHORS WILL NOT
      BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR
      CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR
      DISTRIBUTION OF THIS DOCUMENT.
  -->

  <xsd:include schemaLocation="coremodel.xsd" />
  <xsd:include schemaLocation="soamodel.xsd" />

  <xsd:simpleType name="serviceImplementationModelEnum">
    <xsd:restriction base="s-ramp:baseArtifactEnum">
      <xsd:enumeration value="ServiceInstance" />
      <xsd:enumeration value="ServiceOperation" />
      <xsd:enumeration value="ServiceEndpoint" />
      <xsd:enumeration value="ServiceOperation" />
    </xsd:restriction>
```



```
    </xsd:simpleType>

    <xsd:simpleType name="serviceInstanceEnum">
      <xsd:restriction base="s-ramp:serviceImplementationModelEnum">
        <xsd:enumeration value="ServiceInstance" />
      </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="serviceOperationEnum">
      <xsd:restriction base="s-ramp:serviceImplementationModelEnum">
        <xsd:enumeration value="ServiceOperation" />
      </xsd:restriction>
    </xsd:simpleType>

    <!-- All Service Implementation Model artifacts inherit from
  ServiceImplementationModelType.
         Service Implementation Model Artifacts can have associated documents.

         Service Implementation Model Artifacts can have dependencies upon other
  Service
         Implementation Model Artifacts. -->

    <!-- Service Model Artifact Type -->
    <xsd:complexType name="ServiceImplementationModelType" abstract="true">
      <xsd:complexContent>
        <xsd:extension base="s-ramp:BaseArtifactType">
          <xsd:sequence>
            <!-- Modeled Relationship(s) to documentArtifactType: -->
            <xsd:element name="documentation" type="s-ramp:documentArtifactTarget"
  minOccurs="0" maxOccurs="unbounded" />
          </xsd:sequence>
          <xsd:anyAttribute namespace="##any" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <!-- Organization artifact Type. Used primarily in Service Model -->
    <xsd:complexType name="Organization">
      <xsd:complexContent>
        <xsd:extension base="s-ramp:Actor">
          <xsd:sequence>
            <!-- Modeled Relationship(s) to ServiceImplementationModelType: -->
            <xsd:element name="provides" type="s-ramp:serviceImplementationModelTarget"
  minOccurs="0" maxOccurs="unbounded" />
            <xsd:element name="extension" type="tns:ExtensionType" minOccurs="0" />
            <xsd:element name="end" type="xsd:string" minOccurs="1" maxOccurs="1" />
            <xsd:any namespace="##other" processContents="lax" minOccurs="0"
  maxOccurs="unbounded" />
          </xsd:sequence>
          <xsd:anyAttribute namespace="##any" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="ServiceInstance">
      <xsd:complexContent>
```



```
      <xsd:extension base="tns:ServiceImplementationModelType">
        <xsd:sequence>
          <!-- Modeled Relationship(s) to ServiceInstance(s): -->
          <xsd:element name="uses" type="s-ramp:serviceInstanceTarget" minOccurs="0"
maxOccurs="unbounded" />
          <!-- Modeled Relationship(s) to ServiceEndpoint(s): -->
          <xsd:element name="describedBy" type="s-ramp:serviceInstanceTarget"
minOccurs="0" maxOccurs="unbounded" />
          <xsd:element name="extension" type="tns:ExtensionType" minOccurs="0" />
          <xsd:element name="end" type="xsd:string" minOccurs="1" maxOccurs="1" />
          <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:anyAttribute namespace="##any" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="ServiceOperation">
    <xsd:complexContent>
      <xsd:extension base="tns:ServiceImplementationModelType">
        <xsd:sequence>
          <!-- Modeled Relationship to the DerivedArtifactType that defines this
ServiceOperation: -->
          <xsd:element name="operationDefinedBy" type="s-ramp:derivedArtifactTarget"
minOccurs="0" maxOccurs="1" />
          <xsd:element name="extension" type="tns:ExtensionType" minOccurs="0" />
          <xsd:element name="end" type="xsd:string" minOccurs="1" maxOccurs="1" />
          <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:anyAttribute namespace="##any" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="ServiceEndpoint">
    <xsd:complexContent>
      <xsd:extension base="tns:ServiceImplementationModelType">
        <xsd:sequence>
          <!-- Modeled Relationship with DerivedArtifactType which defines this
ServiceEndpoint: -->
          <xsd:element name="endpointDefinedBy" type="s-ramp:derivedArtifactTarget"
minOccurs="0" maxOccurs="1" />
          <xsd:element name="extension" type="tns:ExtensionType" minOccurs="0" />
          <xsd:element name="end" type="xsd:string" minOccurs="1" maxOccurs="1" />
          <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="url" type="xsd:anyURI" />
        <xsd:anyAttribute namespace="##any" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- Generic Extension Type for all un-used element extension points in tns -->
```



```
  <xsd:complexType name="ExtensionType">
    <xsd:sequence>
      <xsd:any processContents="Lax" minOccurs="1" maxOccurs="unbounded"
namespace="##targetNamespace" />
    </xsd:sequence>
    <xsd:anyAttribute namespace="##any" />
  </xsd:complexType>

  <!-- Relationship referencing the artifact's UUID, to reference a
ServiceImplementationModel. -->
  <xsd:complexType name="serviceImplementationModelTarget">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:target">
        <xsd:attribute name="artifactType" type="s-
ramp:serviceImplementationModelEnum" use="required" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- Relationship referencing the artifact's UUID, to reference a ServiceInstance.
-->
  <xsd:complexType name="serviceInstanceTarget">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:target">
        <xsd:attribute name="artifactType" type="s-ramp:serviceInstanceEnum"
use="required" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- Relationship referencing the artifact's UUID, to reference a ServiceOperation.
-->
  <xsd:complexType name="serviceOperationTarget">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:target">
        <xsd:attribute name="artifactType" type="s-ramp:serviceOperationEnum"
use="required" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```


Appendix H. XSD Model Schema

The S-RAMP XSD Model Schema XSD file is also provided at:

http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0/

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:tns="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0" xmlns:s-
ramp="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
elementFormDefault="qualified" version="1.0">
  <!--
      (c) 2010 Hewlett-Packard Company (HP), International Business Machines
      Corporation (IBM), Software AG (SAG) and TIBCO Software Inc. All
      rights reserved. Permission to copy and display the SOA Repository
      Artifact Model and Protocol (the “Specification”), in any medium
      without fee or royalty is hereby granted by Hewlett-Packard Company
      (HP), International Business Machines Corporation (IBM), Software AG
      (SAG) and TIBCO Software Inc. (collectively, the "Authors"), provided
      that you include the following on ALL copies of this document or
      portions thereof, that you make:

      1. A link or URL to this document at this location:
      http://s-ramp.org/2010/s-ramp/specification/documents/{this document
      name}
      2. The copyright notice as shown in the Specification.

      The Authors each agree to grant you a royalty-free license, under
      reasonable, non-discriminatory terms and conditions to their
      respective patents that they deem necessary to implement the "SOA
      Repository Artifact Model and Protocol” Specification, including all
      its constituent documents. THIS DOCUMENT IS PROVIDED "AS IS," AND THE
      AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED,
      INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS
      FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE
      CONTENTS OF THIS DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE
      IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY
      PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. THE AUTHORS WILL NOT
      BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR
      CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR
      DISTRIBUTION OF THIS DOCUMENT.
  -->

  <xsd:include schemaLocation="coremodel.xsd" />

  <xsd:simpleType name="xsdDocumentEnum">
    <xsd:restriction base="s-ramp:baseArtifactEnum">
      <xsd:enumeration value="XsdDocument" />
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:complexType name="XsdDocument">
    <xsd:complexContent>
```



```
        <xsd:extension base="s-ramp:XmlDocument">
          <xsd:sequence>
            <!-- Derived Relationships with (other)XmlDocument artifact(s): -->
            <xsd:element name="importedXsds" type="s-ramp:xsdDocumentTarget"
minOccurs="0" maxOccurs="unbounded" />
            <xsd:element name="includedXsds" type="s-ramp:xsdDocumentTarget"
minOccurs="0" maxOccurs="unbounded" />
            <xsd:element name="redefinedXsds" type="s-ramp:xsdDocumentTarget"
minOccurs="0" maxOccurs="unbounded" />
          </xsd:sequence>
          <xsd:attribute name="targetNamespace" type="xsd:anyURI" use="optional" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="AttributeDeclaration">
      <xsd:complexContent>
        <xsd:extension base="s-ramp:DerivedArtifactType">
          <xsd:attribute name="NCName" type="xsd:NCName" />
          <xsd:attribute name="namespace" type="xsd:anyURI" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="ElementDeclaration">
      <xsd:complexContent>
        <xsd:extension base="s-ramp:DerivedArtifactType">
          <xsd:attribute name="NCName" type="xsd:NCName" />
          <xsd:attribute name="namespace" type="xsd:anyURI" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="XsdType" abstract="true">
      <xsd:complexContent>
        <xsd:extension base="s-ramp:DerivedArtifactType" />
      </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="ComplexTypeDeclaration">
      <xsd:complexContent>
        <xsd:extension base="tns:XsdType">
          <xsd:attribute name="NCName" type="xsd:NCName" />
          <xsd:attribute name="namespace" type="xsd:anyURI" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="SimpleTypeDeclaration">
      <xsd:complexContent>
        <xsd:extension base="tns:XsdType">
          <xsd:attribute name="NCName" type="xsd:NCName" />
          <xsd:attribute name="namespace" type="xsd:anyURI" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <!-- Relationship referencing the artifact's UUID, to reference an XsdDocument. -->
    <xsd:complexType name="xsdDocumentTarget">
      <xsd:complexContent>
        <xsd:extension base="s-ramp:target">
```



```
2133            <xsd:attribute name="artifactType" type="s-ramp:xsdDocumentEnum"
2134    use="required" />
2135          </xsd:extension>
2136        </xsd:complexContent>
2137      </xsd:complexType>
2138    </xsd:schema>
```


Appendix I. WSDL Model Schema

The S-RAMP WSDL Model Schema XSD file is also provided at:

http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0/

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:tns="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0" xmlns:s-
ramp="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
elementFormDefault="qualified" version="1.0">
  <!--
      (c) 2010 Hewlett-Packard Company (HP), International Business Machines
      Corporation (IBM), Software AG (SAG) and TIBCO Software Inc. All
      rights reserved. Permission to copy and display the SOA Repository
      Artifact Model and Protocol (the “Specification”), in any medium
      without fee or royalty is hereby granted by Hewlett-Packard Company
      (HP), International Business Machines Corporation (IBM), Software AG
      (SAG) and TIBCO Software Inc. (collectively, the "Authors"), provided
      that you include the following on ALL copies of this document or
      portions thereof, that you make:

      1. A link or URL to this document at this location:
      http://s-ramp.org/2010/s-ramp/specification/documents/{this document
      name}
      2. The copyright notice as shown in the Specification.

      The Authors each agree to grant you a royalty-free license, under
      reasonable, non-discriminatory terms and conditions to their
      respective patents that they deem necessary to implement the "SOA
      Repository Artifact Model and Protocol” Specification, including all
      its constituent documents. THIS DOCUMENT IS PROVIDED "AS IS," AND THE
      AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED,
      INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS
      FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE
      CONTENTS OF THIS DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE
      IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY
      PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. THE AUTHORS WILL NOT
      BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR
      CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR
      DISTRIBUTION OF THIS DOCUMENT.
  -->

  <xsd:include schemaLocation="coremodel.xsd" />
  <xsd:include schemaLocation="xsdmodel.xsd" />
  <xsd:include schemaLocation="policymodel.xsd" />

  <xsd:simpleType name="wsdlDocumentEnum">
    <xsd:restriction base="s-ramp:documentArtifactEnum">
      <xsd:enumeration value="WsdlDocument" />
    </xsd:restriction>
  </xsd:simpleType>
```



```
    <xsd:simpleType name="wsdlExtensionEnum">
      <xsd:restriction base="s-ramp:derivedArtifactEnum">
        <xsd:enumeration value="WsdlExtension" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="portEnum">
      <xsd:restriction base="s-ramp:derivedArtifactEnum">
        <xsd:enumeration value="Port" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="bindingEnum">
      <xsd:restriction base="s-ramp:derivedArtifactEnum">
        <xsd:enumeration value="Binding" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="bindingOperationEnum">
      <xsd:restriction base="s-ramp:derivedArtifactEnum">
        <xsd:enumeration value="BindingOperation" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="portTypeEnum">
      <xsd:restriction base="s-ramp:derivedArtifactEnum">
        <xsd:enumeration value="PortType" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="operationEnum">
      <xsd:restriction base="s-ramp:derivedArtifactEnum">
        <xsd:enumeration value="Operation" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="bindingOperationFaultEnum">
      <xsd:restriction base="s-ramp:derivedArtifactEnum">
        <xsd:enumeration value="BindingOperationFault" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="bindingOperationInputEnum">
      <xsd:restriction base="s-ramp:derivedArtifactEnum">
        <xsd:enumeration value="BindingOperationInput" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="bindingOperationOutputEnum">
      <xsd:restriction base="s-ramp:derivedArtifactEnum">
        <xsd:enumeration value="BindingOperationOutput" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="operationInputEnum">
      <xsd:restriction base="s-ramp:derivedArtifactEnum">
        <xsd:enumeration value="OperationInput" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="operationOutputEnum">
      <xsd:restriction base="s-ramp:derivedArtifactEnum">
        <xsd:enumeration value="OperationOutput" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="faultEnum">
```



```
      <xsd:restriction base="s-ramp:derivedArtifactEnum">
        <xsd:enumeration value="Fault" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="messageEnum">
      <xsd:restriction base="s-ramp:derivedArtifactEnum">
        <xsd:enumeration value="Message" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="partEnum">
      <xsd:restriction base="s-ramp:derivedArtifactEnum">
        <xsd:enumeration value="Part" />
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="xsdTypeEnum">
      <xsd:restriction base="s-ramp:derivedArtifactEnum">
        <xsd:enumeration value="XsdType" />
      </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="WsdlDocument">
      <xsd:complexContent>
        <xsd:extension base="s-ramp:XmlDocument">
          <xsd:sequence>
            <!-- Derived Relationships: -->
            <xsd:element name="importedXsds" type="s-ramp:xsdDocumentTarget"
minOccurs="0" maxOccurs="unbounded" />
            <xsd:element name="includedXsds" type="s-ramp:xsdDocumentTarget"
minOccurs="0" maxOccurs="unbounded" />
            <xsd:element name="redefinedXsds" type="s-ramp:xsdDocumentTarget"
minOccurs="0" maxOccurs="unbounded" />
            <xsd:element name="importedWsdls" type="s-ramp:wsdlDocumentTarget"
minOccurs="0" maxOccurs="unbounded" />
          </xsd:sequence>
          <xsd:attribute name="targetNamespace" type="xsd:anyURI" use="optional" />
          <xsd:attribute name="xsdTargetNamespaces" type="xsd:anyURI" use="optional" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="WsdlDerivedArtifactType" abstract="true">
      <xsd:complexContent>
        <xsd:extension base="s-ramp:DerivedArtifactType">
          <xsd:sequence>
            <!-- Modeled "extension" relationship to any wsdlExtension artifact(s) -->
            <xsd:element name="extension" type="s-ramp:wsdlExtensionTarget"
minOccurs="0" maxOccurs="unbounded" />
          </xsd:sequence>
          <xsd:attribute name="namespace" type="xsd:anyURI" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="WsdlService">
      <xsd:complexContent>
        <xsd:extension base="tns:NamedWsdlDerivedArtifactType">
          <xsd:sequence>
            <!-- Derived Relationship to Port(s) this Service has: -->
```



```
2302            <xsd:element name="port" type="s-ramp:portTarget" minOccurs="1"
2303    maxOccurs="unbounded" />
2304          </xsd:sequence>
2305        </xsd:extension>
2306      </xsd:complexContent>
2307    </xsd:complexType>
2308    <xsd:complexType name="Port">
2309      <xsd:complexContent>
2310        <xsd:extension base="tns:NamedWsdlDerivedArtifactType">
2311          <xsd:sequence>
2312            <!-- Derived Relationships with Binding artifact: -->
2313            <xsd:element name="Binding" type="s-ramp:bindingTarget" />
2314          </xsd:sequence>
2315        </xsd:extension>
2316      </xsd:complexContent>
2317    </xsd:complexType>
2318    <xsd:complexType name="Binding">
2319      <xsd:complexContent>
2320        <xsd:extension base="tns:NamedWsdlDerivedArtifactType">
2321          <xsd:sequence>
2322            <!-- Derived Relationship to BindingOperation(s): -->
2323            <xsd:element name="bindingOperation" type="s-ramp:bindingOperationTarget"
2324    minOccurs="0" maxOccurs="unbounded" />
2325            <!-- Derived Relationship with this Binding's PortType: -->
2326            <xsd:element name="portType" type="s-ramp:portTypeTarget" />
2327          </xsd:sequence>
2328        </xsd:extension>
2329      </xsd:complexContent>
2330    </xsd:complexType>
2331    <xsd:complexType name="PortType">
2332      <xsd:complexContent>
2333        <xsd:extension base="tns:NamedWsdlDerivedArtifactType">
2334          <xsd:sequence>
2335            <!-- Derived Relationship to this PortType's Operation(s): -->
2336            <xsd:element name="operation" type="s-ramp:operationTarget" minOccurs="0"
2337    maxOccurs="unbounded" />
2338          </xsd:sequence>
2339        </xsd:extension>
2340      </xsd:complexContent>
2341    </xsd:complexType>
2342    <xsd:complexType name="BindingOperation">
2343      <xsd:complexContent>
2344        <xsd:extension base="tns:NamedWsdlDerivedArtifactType">
2345          <xsd:sequence>
2346            <!-- Derived Relationship to BindingOperationFault(s) for this
2347    BindingOperation: -->
2348            <xsd:element name="fault" type="s-ramp:bindingOperationFaultTarget"
2349    minOccurs="0" maxOccurs="unbounded" />
2350            <!-- Derived Relationship to BindingOperationInput for this
2351    BindingOperation: -->
2352            <xsd:element name="input" type="s-ramp:bindingOperationInputTarget"
2353    minOccurs="0" maxOccurs="1" />
2354            <!-- Derived Relationship to BindingOperationOutput for this
2355    BindingOperation: -->
2356            <xsd:element name="output" type="s-ramp:bindingOperationOutputTarget"
2357    minOccurs="0" maxOccurs="1" />
```



```
            <!-- Derived Relationship with Operation artifact: -->
            <xsd:element name="operation" type="s-ramp:operationTarget" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="BindingOperationInput">
      <xsd:complexContent>
        <xsd:extension base="tns:NamedWsdlDerivedArtifactType">
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="BindingOperationFault">
      <xsd:complexContent>
        <xsd:extension base="tns:NamedWsdlDerivedArtifactType" />
      </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="Operation">
      <xsd:complexContent>
        <xsd:extension base="tns:NamedWsdlDerivedArtifactType">
          <xsd:sequence>
            <!-- Derived Relationships to OperationInput and OperationOutput for this
Operation: -->
            <xsd:element name="input" type="s-ramp:operationInputTarget" />
            <xsd:element name="output" type="s-ramp:operationOutputTarget" />
            <!-- Derived Relationship for fault(s) associated with this Operation: -->
            <xsd:element name="fault" type="s-ramp:faultTarget" minOccurs="0"
maxOccurs="unbounded" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
      <!-- Only request / response is modeled -->
    </xsd:complexType>
    <xsd:complexType name="OperationInput">
      <xsd:complexContent>
        <xsd:extension base="tns:NamedWsdlDerivedArtifactType">
          <xsd:sequence>
            <!-- Derived Relationship with Message artifact: -->
            <xsd:element name="message" type="s-ramp:messageTarget" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="Fault">
      <xsd:complexContent>
        <xsd:extension base="tns:NamedWsdlDerivedArtifactType">
          <xsd:sequence>
            <!-- Derived Relationship with Message artifact: -->
            <xsd:element name="message" type="s-ramp:messageTarget" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="Message">
      <xsd:complexContent>
        <xsd:extension base="tns:NamedWsdlDerivedArtifactType">
```



```
2414          <xsd:sequence>
2415            <!-- Derived Relationship to Part(s) of this Message: -->
2416            <xsd:element name="part" type="s-ramp:partTarget" minOccurs="0"
2417    maxOccurs="unbounded" />
2418          </xsd:sequence>
2419        </xsd:extension>
2420      </xsd:complexContent>
2421    </xsd:complexType>
2422    <xsd:complexType name="Part">
2423      <xsd:complexContent>
2424        <xsd:extension base="tns:NamedWsdlDerivedArtifactType">
2425          <xsd:sequence>
2426            <!-- Derived Relationships with ElementDeclaraion and XSDType artifacts: --
2427    >
2428            <xsd:element name="type" type="s-ramp:xsdTypeTarget" minOccurs="0"
2429    maxOccurs="1" />
2430            <xsd:element name="element" type="s-ramp:elementTarget" minOccurs="0"
2431    maxOccurs="1" />
2432          </xsd:sequence>
2433        </xsd:extension>
2434      </xsd:complexContent>
2435    </xsd:complexType>
2436    <xsd:complexType name="BindingOperationOutput">
2437      <xsd:complexContent>
2438        <xsd:extension base="tns:NamedWsdlDerivedArtifactType">
2439        </xsd:extension>
2440      </xsd:complexContent>
2441    </xsd:complexType>
2442    <xsd:complexType name="OperationOutput">
2443      <xsd:complexContent>
2444        <xsd:extension base="tns:NamedWsdlDerivedArtifactType">
2445          <xsd:sequence>
2446            <!-- Derived Relationship with Message for this OperationOutput: -->
2447            <xsd:element name="message" type="s-ramp:messageTarget" />
2448          </xsd:sequence>
2449        </xsd:extension>
2450      </xsd:complexContent>
2451    </xsd:complexType>
2452    <xsd:complexType name="NamedWsdlDerivedArtifactType" abstract="true">
2453      <xsd:complexContent>
2454        <xsd:extension base="tns:WsdlDerivedArtifactType">
2455          <xsd:attribute name="NCName" type="xsd:NCName" use="optional" />
2456        </xsd:extension>
2457      </xsd:complexContent>
2458    </xsd:complexType>
2459    <xsd:complexType name="WsdlExtension">
2460      <xsd:complexContent>
2461        <xsd:extension base="s-ramp:DerivedArtifactType">
2462          <xsd:attribute name="NCName" type="xsd:NCName" use="optional" />
2463          <xsd:attribute name="namespace" type="xsd:anyURI" />
2464        </xsd:extension>
2465      </xsd:complexContent>
2466    </xsd:complexType>
2467
2468    <!-- Relationship referencing the artifact's UUID, to reference a WSDLDocument. -->
2469    <xsd:complexType name="wsdlDocumentTarget">
```



```
      <xsd:complexContent>
        <xsd:extension base="s-ramp:target">
          <xsd:attribute name="artifactType" type="s-ramp:wsdlDocumentEnum"
use="required" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <!-- Relationship referencing the artifact's UUID, to reference a WSDLExtention. --
>
    <xsd:complexType name="wsdlExtensionTarget">
      <xsd:complexContent>
        <xsd:extension base="s-ramp:target">
          <xsd:attribute name="artifactType" type="s-ramp:wsdlExtensionEnum"
use="required" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <!-- Relationship referencing the artifact's UUID, to reference a Port. -->
    <xsd:complexType name="portTarget">
      <xsd:complexContent>
        <xsd:extension base="s-ramp:target">
          <xsd:attribute name="artifactType" type="s-ramp:portEnum" use="required" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <!-- Relationship referencing the artifact's UUID, to reference a Binding. -->
    <xsd:complexType name="bindingTarget">
      <xsd:complexContent>
        <xsd:extension base="s-ramp:target">
          <xsd:attribute name="artifactType" type="s-ramp:bindingEnum" use="required"
/>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <!-- Relationship referencing the artifact's UUID, to reference a BindingOperation.
-->
    <xsd:complexType name="bindingOperationTarget">
      <xsd:complexContent>
        <xsd:extension base="s-ramp:target">
          <xsd:attribute name="artifactType" type="s-ramp:bindingOperationEnum"
use="required" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <!-- Relationship referencing the artifact's UUID, to reference a PortType. -->
    <xsd:complexType name="portTypeTarget">
      <xsd:complexContent>
        <xsd:extension base="s-ramp:target">
          <xsd:attribute name="artifactType" type="s-ramp:portTypeEnum" use="required"
/>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <!-- Relationship referencing the artifact's UUID, to reference an Operation. -->
    <xsd:complexType name="operationTarget">
      <xsd:complexContent>
```



```
      <xsd:extension base="s-ramp:target">
        <xsd:attribute name="artifactType" type="s-ramp:operationEnum" use="required"
/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <!-- Relationship referencing the artifact's UUID, to reference a
BindingOperationFault. -->
  <xsd:complexType name="bindingOperationFaultTarget">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:target">
        <xsd:attribute name="artifactType" type="s-ramp:bindingOperationFaultEnum"
use="required" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <!-- Relationship referencing the artifact's UUID, to reference a
BindingOperationInputTarget. -->
  <xsd:complexType name="bindingOperationInputTarget">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:target">
        <xsd:attribute name="artifactType" type="s-ramp:bindingOperationInputEnum"
use="required" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <!-- Relationship referencing the artifact's UUID, to reference a
BindingOperationOutput. -->
  <xsd:complexType name="bindingOperationOutputTarget">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:target">
        <xsd:attribute name="artifactType" type="s-ramp:bindingOperationOutputEnum"
use="required" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <!-- Relationship referencing the artifact's UUID, to reference an OperationInput.
-->
  <xsd:complexType name="operationInputTarget">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:target">
        <xsd:attribute name="artifactType" type="s-ramp:operationInputEnum"
use="required" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <!-- Relationship referencing the artifact's UUID, to reference an OperationOutput.
-->
  <xsd:complexType name="operationOutputTarget">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:target">
        <xsd:attribute name="artifactType" type="s-ramp:operationOutputEnum"
use="required" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
```



```
  <!-- Relationship referencing the artifact's UUID, to reference a Fault. -->
  <xsd:complexType name="faultTarget">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:target">
        <xsd:attribute name="artifactType" type="s-ramp:faultEnum" use="required" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <!-- Relationship referencing the artifact's UUID, to reference a Message. -->
  <xsd:complexType name="messageTarget">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:target">
        <xsd:attribute name="artifactType" type="s-ramp:messageEnum" use="required"
/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <!-- Relationship referencing the artifact's UUID, to reference a Part. -->
  <xsd:complexType name="partTarget">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:target">
        <xsd:attribute name="artifactType" type="s-ramp:partEnum" use="required" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <!-- Relationship referencing the artifact's UUID, to reference any XsdType. -->
  <xsd:complexType name="xsdTypeTarget">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:target">
        <xsd:attribute name="artifactType" type="s-ramp:xsdTypeEnum" use="required"
/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```


Appendix J. SOAP WSDL Model Schema

The S-RAMP SOAP WSDL Model Schema XSD file is also provided at:

http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0/

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:tns="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0" xmlns:s-
ramp="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
elementFormDefault="qualified" version="1.0">
  <!--
      (c) 2010 Hewlett-Packard Company (HP), International Business Machines
      Corporation (IBM), Software AG (SAG) and TIBCO Software Inc. All
      rights reserved. Permission to copy and display the SOA Repository
      Artifact Model and Protocol (the “Specification”), in any medium
      without fee or royalty is hereby granted by Hewlett-Packard Company
      (HP), International Business Machines Corporation (IBM), Software AG
      (SAG) and TIBCO Software Inc. (collectively, the "Authors"), provided
      that you include the following on ALL copies of this document or
      portions thereof, that you make:

      1. A link or URL to this document at this location:
      http://s-ramp.org/2010/s-ramp/specification/documents/{this document
      name}
      2. The copyright notice as shown in the Specification.

      The Authors each agree to grant you a royalty-free license, under
      reasonable, non-discriminatory terms and conditions to their
      respective patents that they deem necessary to implement the "SOA
      Repository Artifact Model and Protocol” Specification, including all
      its constituent documents. THIS DOCUMENT IS PROVIDED "AS IS," AND THE
      AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED,
      INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS
      FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE
      CONTENTS OF THIS DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE
      IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY
      PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. THE AUTHORS WILL NOT
      BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR
      CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR
      DISTRIBUTION OF THIS DOCUMENT.
  -->

  <xsd:include schemaLocation="coremodel.xsd" />
  <xsd:include schemaLocation="wsdlmodel.xsd" />

  <xsd:complexType name="SoapAddress">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:WsdlExtension">
        <xsd:attribute name="soapLocation" type="xsd:anyURI" />
      </xsd:extension>
    </xsd:complexContent>
```



```
    </xsd:complexType>

    <xsd:complexType name="SoapBinding">
      <xsd:complexContent>
        <xsd:extension base="s-ramp:WsdlExtension">
          <xsd:attribute name="style" type="xsd:string" />
          <xsd:attribute name="transport" type="xsd:string" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:schema>
```


Appendix K. Policy Model Schema

The S-RAMP Policy Model Schema XSD file is also provided at:

http://s-ramp.org/2010/specification/schemas/

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:tns="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0" xmlns:s-
ramp="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
elementFormDefault="qualified" version="1.0">
  <!--
      (c) 2010 Hewlett-Packard Company (HP), International Business Machines
      Corporation (IBM), Software AG (SAG) and TIBCO Software Inc. All
      rights reserved. Permission to copy and display the SOA Repository
      Artifact Model and Protocol (the “Specification”), in any medium
      without fee or royalty is hereby granted by Hewlett-Packard Company
      (HP), International Business Machines Corporation (IBM), Software AG
      (SAG) and TIBCO Software Inc. (collectively, the "Authors"), provided
      that you include the following on ALL copies of this document or
      portions thereof, that you make:

      1. A link or URL to this document at this location:
      http://s-ramp.org/2010/s-ramp/specification/documents/{this document
      name}
      2. The copyright notice as shown in the Specification.

      The Authors each agree to grant you a royalty-free license, under
      reasonable, non-discriminatory terms and conditions to their
      respective patents that they deem necessary to implement the "SOA
      Repository Artifact Model and Protocol” Specification, including all
      its constituent documents. THIS DOCUMENT IS PROVIDED "AS IS," AND THE
      AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED,
      INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS
      FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE
      CONTENTS OF THIS DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE
      IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY
      PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. THE AUTHORS WILL NOT
      BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR
      CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR
      DISTRIBUTION OF THIS DOCUMENT.
  -->

  <xsd:include schemaLocation="coremodel.xsd" />
  <xsd:include schemaLocation="wsdlmodel.xsd" />

  <xsd:complexType name="SoapAddress">
    <xsd:complexContent>
      <xsd:extension base="s-ramp:WsdlExtension">
        <xsd:attribute name="soapLocation" type="xsd:anyURI" />
      </xsd:extension>
    </xsd:complexContent>
```



```
    </xsd:complexType>

    <xsd:complexType name="SoapBinding">
      <xsd:complexContent>
        <xsd:extension base="s-ramp:WsdlExtension">
          <xsd:attribute name="style" type="xsd:string" />
          <xsd:attribute name="transport" type="xsd:string" />
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:schema>
```

