



S-RAMP Version 1.0. Part 1: Foundation

Committee Specification Draft 01

20 March 2013

Specification URIs

This version:

<http://docs.oasis-open.org/s-ramp/s-ramp/v1.0/csd01/part1-foundation/s-ramp-v1.0-csd01-part1-foundation.doc> (Authoritative)
<http://docs.oasis-open.org/s-ramp/s-ramp/v1.0/csd01/part1-foundation/s-ramp-v1.0-csd01-part1-foundation.html>
<http://docs.oasis-open.org/s-ramp/s-ramp/v1.0/csd01/part1-foundation/s-ramp-v1.0-csd01-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 also includes:

- XML schemas: <http://docs.oasis-open.org/s-ramp/s-ramp/v1.0/csd01/schemas/>
- *S-RAMP Version 1.0. Part 1: Foundation*. (this document)
<http://docs.oasis-open.org/s-ramp/s-ramp/v1.0/csd01/part1-foundation/s-ramp-v1.0-csd01-part1-foundation.html>
- *S-RAMP Version 1.0. Part 2: Atom Binding*.
<http://docs.oasis-open.org/s-ramp/s-ramp/v1.0/csd01/part2-atom-binding/s-ramp-v1.0-csd01-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. 20 March 2013. OASIS Committee Specification Draft 01. <http://docs.oasis-open.org/s-ramp/s-ramp/v1.0/csd01/part1-foundation/s-ramp-v1.0-csd01-part1-foundation.html>.

Notices

Copyright © OASIS Open 2013. All Rights Reserved.

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.....	6
1.1	Terminology.....	6
1.2	Diagrams used in this document.....	6
1.2.1	Attributes and elements.....	6
1.2.2	Element structure.....	7
1.2.3	Cardinality.....	7
1.3	Normative References.....	7
1.4	Non-Normative References.....	8
1.5	Problem Statement and Objectives.....	8
1.6	Use Case Scenarios.....	8
1.7	Design Principles.....	10
1.8	S-RAMP Schemas.....	10
1.9	XML Namespaces.....	10
2	Artifact Type Model.....	12
2.1	Artifact Type Models.....	12
2.1.1	Artifact Metadata.....	13
2.1.2	Artifact Relationships.....	13
2.2	The Core Model.....	14
2.2.1	Base Artifact Type.....	16
2.2.2	Document Artifact Types.....	17
2.2.3	Miscellaneous Types.....	17
2.3	Modeling SOA Concepts.....	17
2.3.1	The SOA Model.....	18
2.3.2	The Service Implementation Model.....	21
2.4	Derived Models.....	23
2.4.1	The XSD Model.....	23
2.4.2	The WSDL Model.....	25
2.4.3	The SOAPWSDL Model.....	27
2.4.4	The Policy Model.....	27
2.5	Referencing S-RAMP Artifacts.....	28
2.5.1	Notional Syntax.....	28
3	Classification Systems in S-RAMP.....	31
4	S-RAMP Query Model.....	34
4.1	Query Dialect (XPath2) Context.....	34
4.2	Query Expression Predicates.....	35
4.3	Query Functions.....	37
4.4	Query Grammar.....	39
4.5	Stored Queries.....	41
5	Conformance.....	43
5.1	Introduction.....	43
5.2	Data Model.....	43
5.3	Classification Systems.....	45
5.4	Query Model.....	45

Appendix A.	Acknowledgments	47
Appendix B.	Non-Normative Text	48
Appendix C.	Revision History	49
Appendix D.	Glossary	50
Appendix E.	Core Model Schema	51
Appendix F.	SOA Model Schema.....	58
Appendix G.	Service Implementation Model Schema	67
Appendix H.	XSD Model Schema.....	71
Appendix I.	WSDL Model Schema.....	74
Appendix J.	SOAP WSDL Model Schema.....	83
Appendix K.	Policy Model Schema.....	85

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.

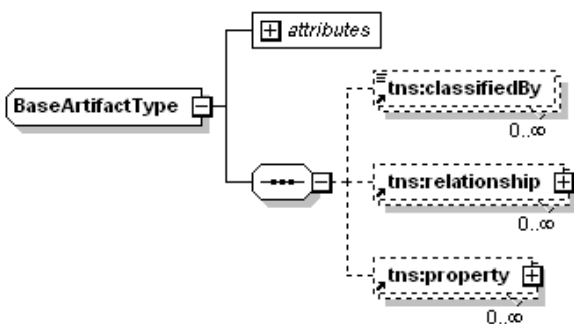
1.2 Diagrams used in this document

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 diagrams¹ 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.

33 1.2.2 Element structure

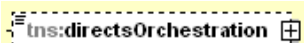


34 The octagonal symbol with the horizontal "dotted" line indicates "sequence of." This diagram says the
35 type *BaseArtifactType* consists of elements *classifiedBy*, *relationship* and *properties*. All three elements
36 are defined in the namespace whose prefix is "s-ramp".

37 The fact that *relationship* and *properties* have a box with a "+" in it at their right-hand end indicates that
38 there is more structure to them than is shown in the diagram.

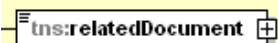
40 1.2.3 Cardinality

41 1.2.3.1 Optional, one



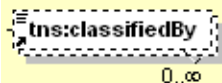
42 The dashed line indicates that the element *directsOrchestration* is optional. The fact that it is not adorned
43 with some other cardinality indicator (see below) says there can be at most one of them.

45 1.2.3.2 Mandatory, one



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

48 1.2.3.3 Optional, repeating



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

53 1.2.3.4 Mandatory, repeating



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

57 1.3 Normative References

58 [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP
59 14, RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>.

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

62

63 1.4 Non-Normative References

64	[XML]	<i>Extensible Markup Language (XML) 1.0 Specification (Fifth Edition)</i> . November 2008. W3C Recommendation. http://www.w3.org/TR/2008/REC-xml-20081126/
65		
66	[XMLNS]	<i>Namespaces in XML 1.0 (Second Edition)</i> . August 2006. W3C Recommendation. http://www.w3.org/TR/2006/REC-xml-names-20060816/
67		
68	[XSD]	<i>XML Schema Part 1: Structures Second Edition, version 1.0</i> . October 2004. W3C Recommendation. http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/
69		
70		
71	[XPath]	<i>XML Path Language (XPath) 2.0 (Second Edition)</i> . December 2010. W3C Recommendation. http://www.w3.org/TR/2010/REC-xpath20-20101214/
72		
73	[RDF]	<i>RDF Primer</i> . February 2004. W3C Recommendation. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/
74		
75	[OWL]	<i>OWL Web Ontology Language Guide</i> . February 2004. W3C Recommendation. http://www.w3.org/TR/2004/REC-owl-guide-20040210/
76		
77	[WSDL]	<i>Web Services Description Language (WSDL), Version 1.1</i> . March 2001. W3C Note. http://www.w3.org/TR/2001/NOTE-wsdl-20010315
78		
79	[ISO6392]	<i>Codes for the Representation of Names and Languages – Part 2</i> . 1998. ISO 639-2. http://www.loc.gov/standards/iso639-2/normtext.html
80		
81	[WSFWK]	<i>Web Services Policy 1.5 – Framework</i> . September 2007. W3C Recommendation. http://www.w3.org/TR/2007/REC-ws-policy-20070904
82		
83	[WSATTCH]	<i>Web Services Policy 1.5 – Attachment</i> . September 2007. W3C Recommendation. http://www.w3.org/TR/2007/REC-ws-policy-attach-20070904
84		
85	[UUID]	P. Leach, M. Mealling, and R. Salz, <i>A Universally Unique Identifier (UUID) URN Namespace</i> . July 2005. IETF RFC 4122. http://www.ietf.org/rfc/rfc4122.txt
86		
87	[QUERYOPS]	<i>XQuery 1.0 and XPath 2.0 Functions and Operators (Second Edition)</i> . December 2010. W3C Recommendation. http://www.w3.org/TR/2010/REC-xpath-functions-20101214/
88		
89		

90 1.5 Problem Statement and Objectives

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

102 1.6 Use Case Scenarios

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

107

108 *Table 1: Design Time Tool Repository Interaction Use Cases*

Tool Category	Activities	S-RAMP Feature	Examples
Integrated Development Environment (IDE)	<ol style="list-style-type: none"> 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 	WsdIDocument XsdDocument OWL classifications PortType	WID RSA Together VisualStudio Oracle JDeveloper
Business Process Modeling Tools	<ol style="list-style-type: none"> 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

109

110 *Table 2: Run Time Tool Repository Interaction Use Cases*

Tool Category	Activities	S-RAMP Feature	Examples
Testing	<ol style="list-style-type: none"> 1. Search to find a WSDL 2. Understand policies associated with WSDL 3. Understand relationships between SOA components 4. Notification when WSDL changes 	WsdIDocument Service PolicyAttachment Policy	HP Service Test Manager Rational Testing Tools Itko Lisa
ESB	<ol style="list-style-type: none"> 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	<ol style="list-style-type: none"> 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

111

112

113

114 Table 3: Monitoring Tool Repository Interaction Use Cases

Tool Category	Activities	S-RAMP Feature	Examples
Service Monitoring	<ol style="list-style-type: none"> 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 Publishing Protocol), ASF (Atom Syndication Format), etc.).
- 119
- 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

148 2 Artifact Type Model

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

154 2.1 Artifact Type Models

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

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

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

185

186 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.

187 2.1.1 Artifact Metadata

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

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

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

201 2.1.2 Artifact Relationships

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

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

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

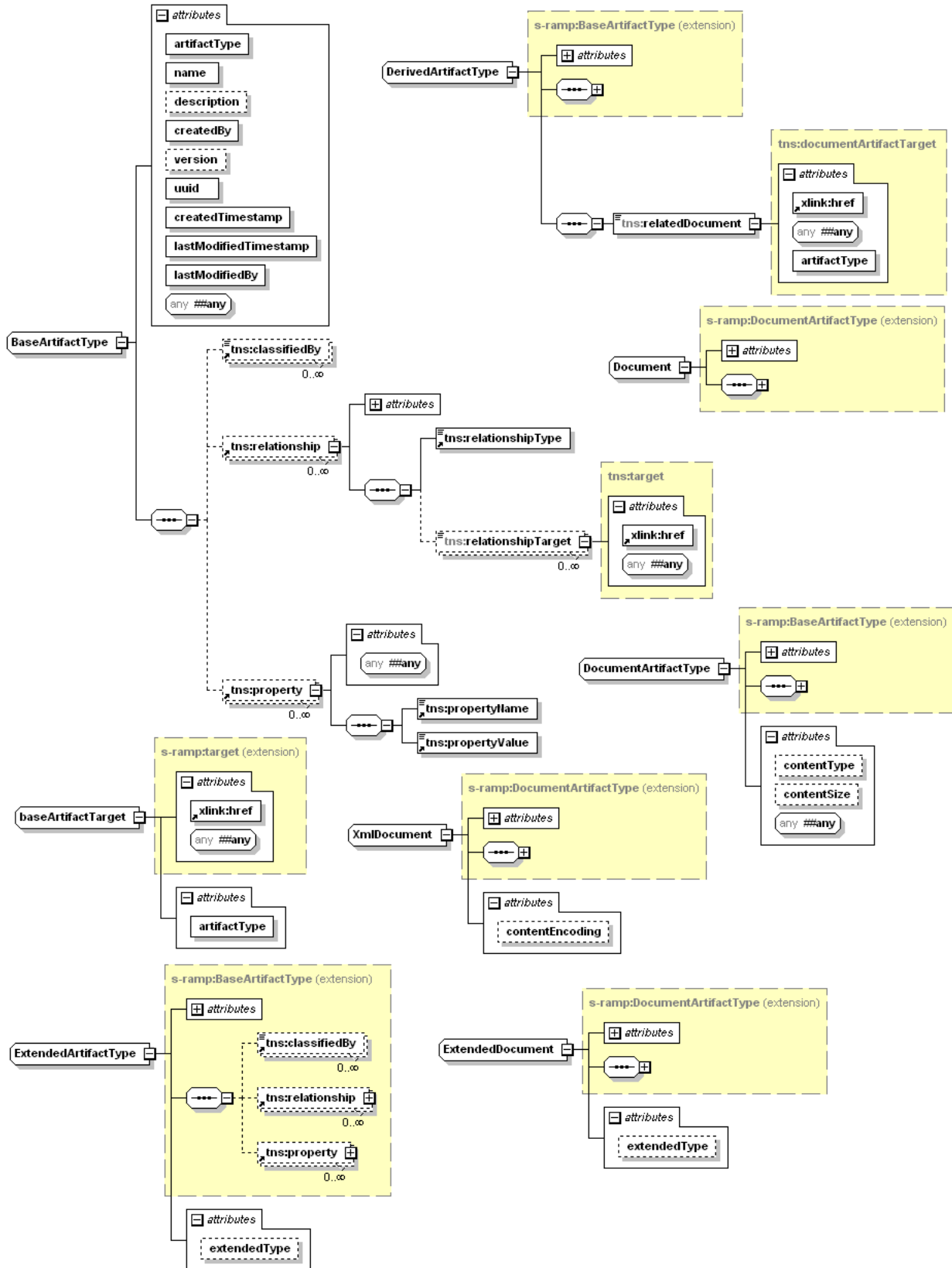
215 There are 4 types of relationships supported in S-RAMP. Refer to the table in Appendix **Error!**
216 **Reference source not found.** for a complete list of pre-defined relationships, an indication of whether the
217 relationship is derived, and the model in which it occurs.

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

- 229 elements defined in the schema within the applicable Artifact Type. There are several
230 considerations related to target cardinality of Modeled Relationships:
- 231 • **Modeled Relationships with Minimum Cardinality = 0:**
 - 232 • Instances of the Source Artifact can be created independently of the Target Artifact.
 - 233 • **Modeled Relationships with Minimum Cardinality > 0:**
 - 234 • Instances of the Source Artifact cannot be created without the appropriate Target
 - 235 Artifact(s) based upon the required minimum cardinality of the relationship.
 - 236 • This can be accomplished by publishing the relevant artifacts at the same time, or by
 - 237 publishing the target artifact(s) first.
 - 238 • Actions that result in the deletion of a relationship instance are not permitted if that would
 - 239 result in a violation of the minimum cardinality.
 - 240 • **Modeled Relationships with Maximum Cardinality < unbounded:**
 - 241 • Relationship instances cannot be created if that would result in a violation of the
 - 242 maximum cardinality limit for the Modeled Relationship.
- 243 Note that in cases where the minimum cardinality equals the maximum cardinality, such a
244 relationship must be created or updated in a single step to avoid intermediate states that would
245 violate these requirements.
- 246 3. **Generic Relationships.** These are user-defined ad-hoc directed relationship instances between
247 any two artifacts in S-RAMP. They always have a minimum cardinality of 0 and an unbounded
248 maximum cardinality. The Relationship Type value of a Generic Relationship instance is chosen
249 by the client, but it MUST NOT match any pre-defined Relationship Type values already defined
250 by the S-RAMP Modeled and Derived relationships (see Appendix **Error! Reference source not**
251 **found.**). The S-RAMP serialization of a Generic Relationship uses the *relationship* structure
252 defined in the Core Model.
 - 253 4. **Extended Artifact Modeled Relationships.** Users may define their own extended artifact
254 models, which may include modeled relationships. How and whether such models are supported
255 is beyond the scope of this specification. Such models are called “Extended Models”. Since pre-
256 defined relationships in a model are termed “Modeled”, then in this context they are called
257 “Extended Modeled Relationships”. The S-RAMP serialization of an Extended Modeled
258 Relationship uses the S-RAMP *relationship* structure defined in the Core Model.

259 2.2 The Core Model

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



265
 266
 267

Figure 1: Conceptualized Model of Core Model Artifacts

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

272 2.2.1 Base Artifact Type

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

276 2.2.1.1 Built-in Properties:

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

301 2.2.1.2 Generic Properties

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

305 2.2.1.3 Generic Relationships

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

309 2.2.1.4 Classifications

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

312 class from a classification system defined to the repository. For more on OWL classification
313 systems in S-RAMP, see Section **Error! Reference source not found.**
314

315 2.2.2 Document Artifact Types

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

318 **contentType:**

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

321 **contentSize:**

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

324 **contentHash:**

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

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

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

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

333 2.2.3 Miscellaneous Types

334 There are a few miscellaneous classes in the Core Model:

335 **StoredQuery:**

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

339 **ExtendedArtifactType:**

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

343 **ExtendedDocument:**

- 344 • The *ExtendedDocument* allows clients to create their own extended artifact model when it is not
345 pre-defined by the S-RAMP specification. The *extendedType* property is intended to provide an
346 indication of the artifact type. This differs from the *ExtendedArtifactType* in that it extends
347 *DocumentArtifactType*, thereby inheriting its properties. This type may be used by clients when
348 adding extended artifact model artifacts that contain document content.
349

350 2.3 Modeling SOA Concepts

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

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

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

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

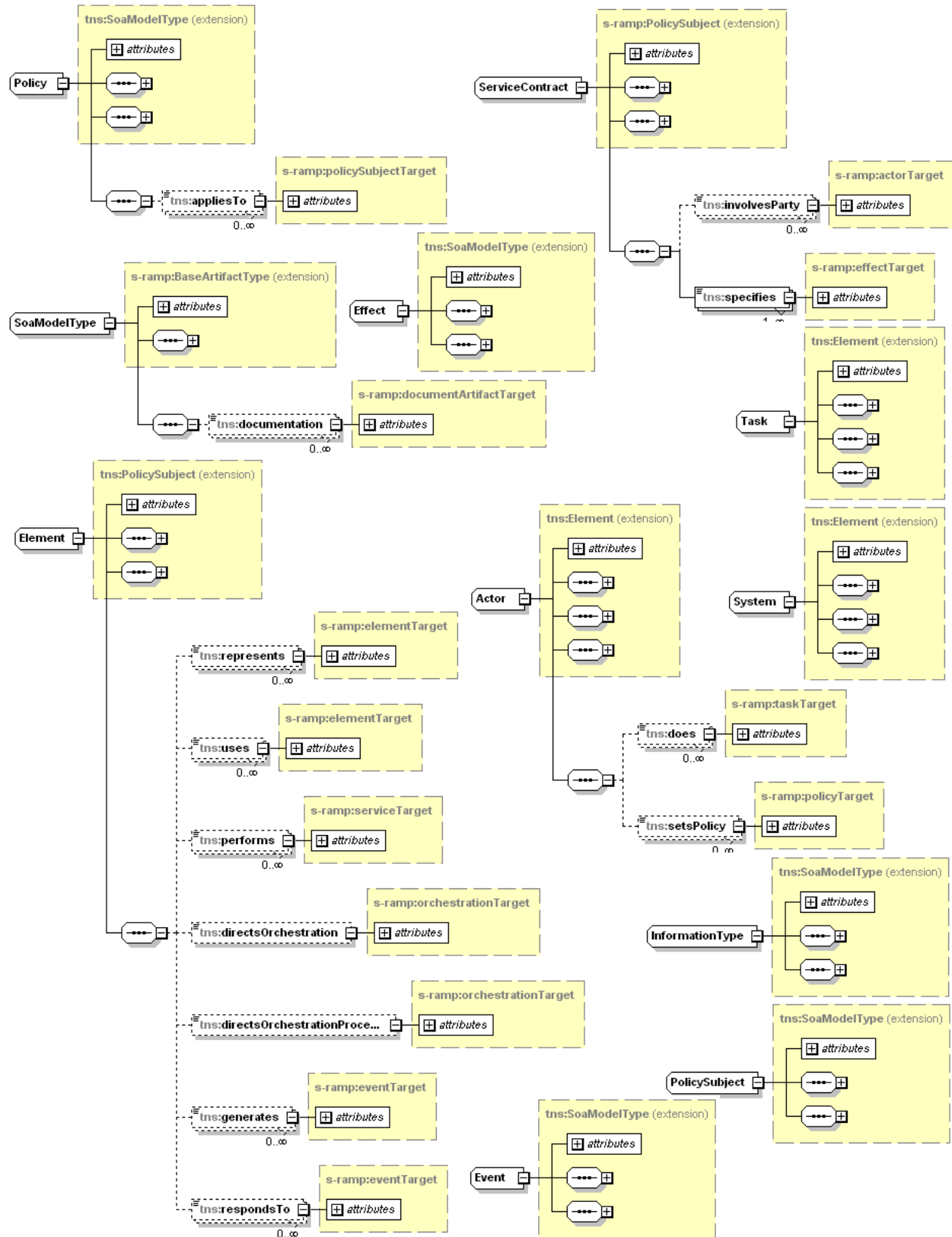
362 **2.3.1 The SOA Model**

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

366

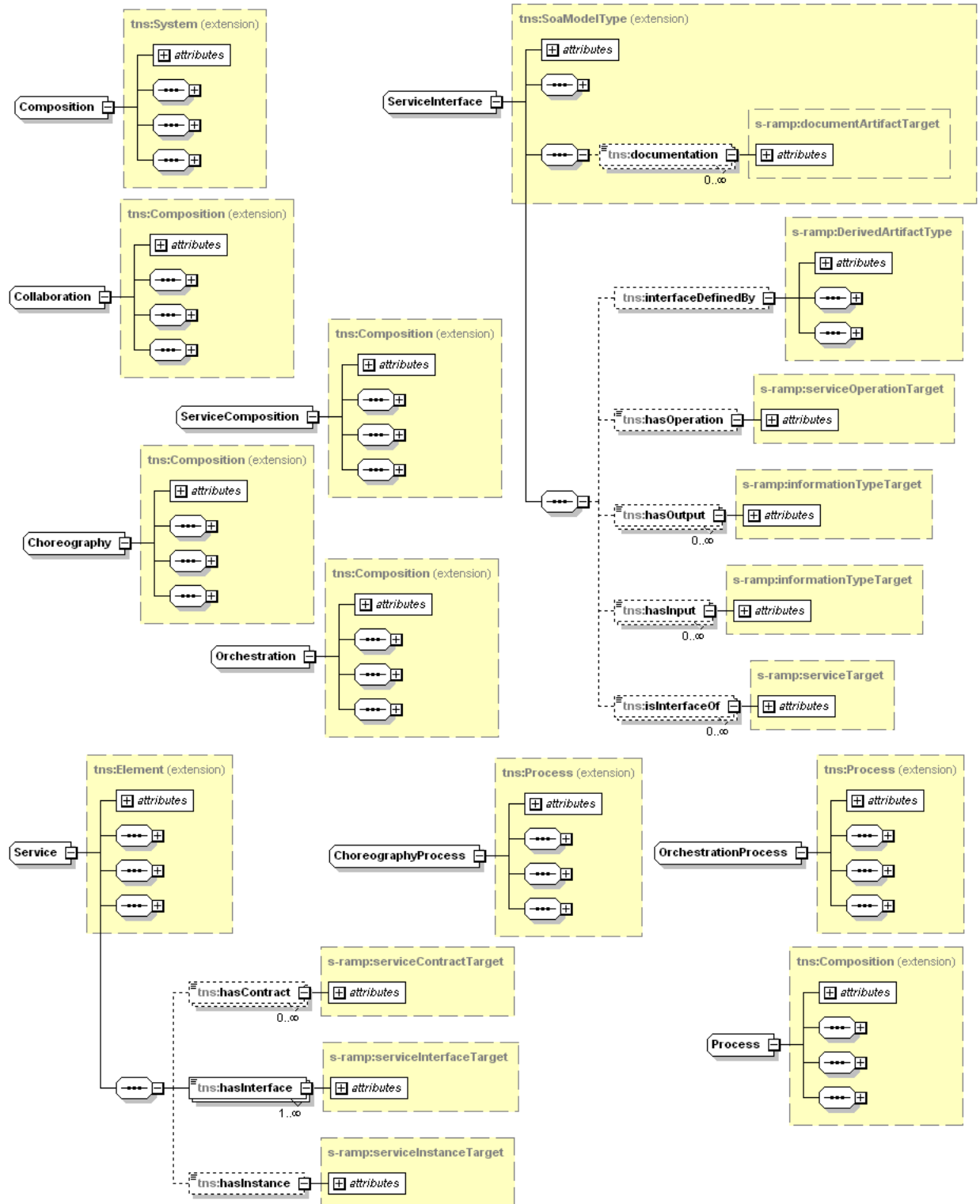
367

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



372
373

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



374

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

376

377 SOA Model artifacts and the SOA Ontology artifacts referenced by the SOA Model are designed to

378 provide clients the ability to create conceptual SOA representations. SOA Model Artifacts are all logical

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

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

384 **2.3.1.1 SOA Model Artifact Types and Relationships**

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

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

396

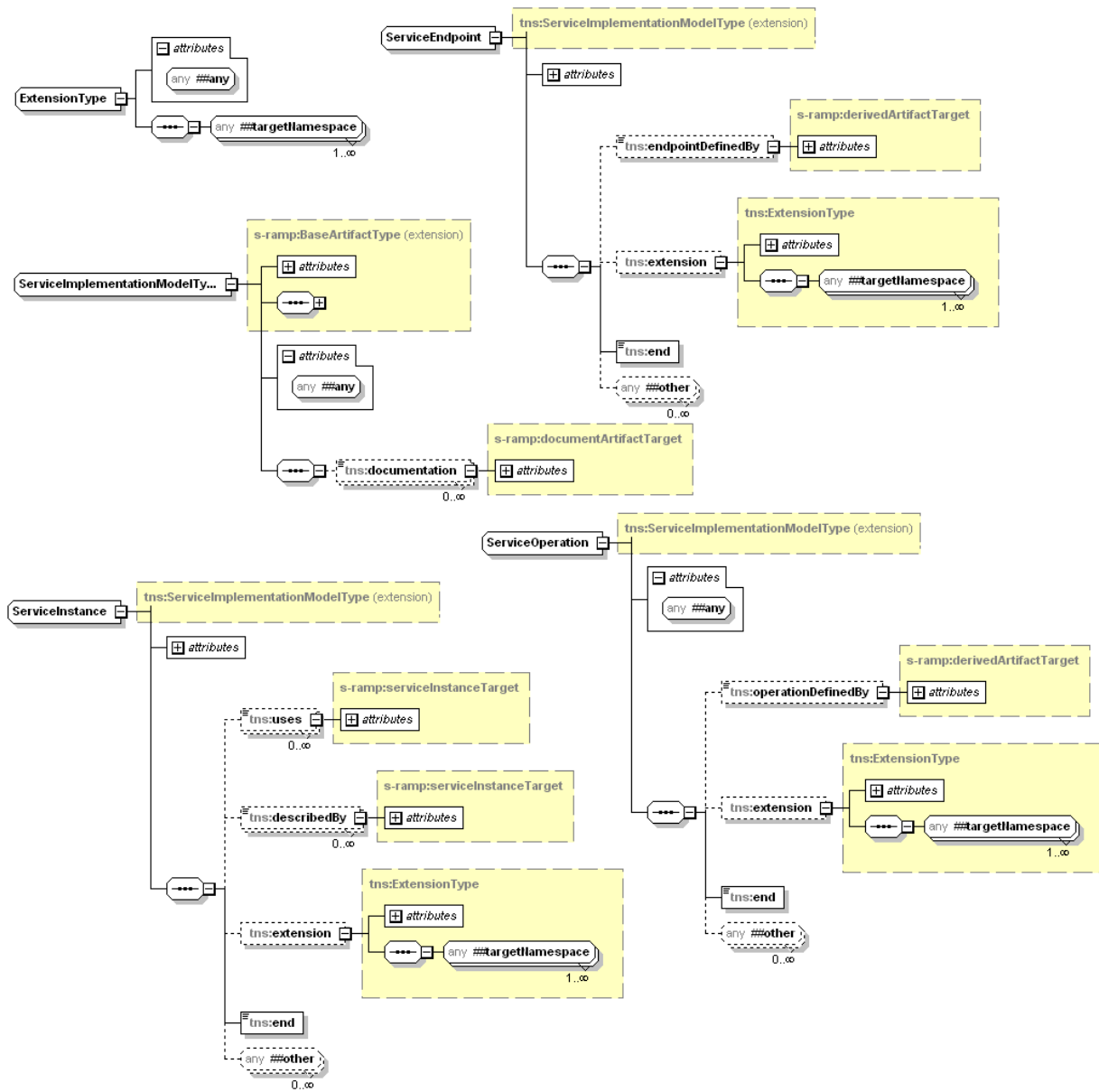
397 *Table 6: SOA Model Relationships*

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	<i>DerivedArtifactType</i>	This allows one to indicate the Derived Artifact instance which defines this service interface. For example, this could be <i>PortType</i> artifact instance from the WSDL Model.
policyDefinedBy	Policy	<i>DerivedArtifactType</i>	This allows one to link the SOA Model Policy artifact with a concrete S-RAMP derived artifact type (e.g., PolicyAttachment)
informationTypeDefinedBy	InformationType	<i>DerivedArtifactType</i>	This allows one to link the SOA Model InformationType artifact with a concrete S-RAMP derived artifact type (e.g., PolicyAttachment)

398

399 **2.3.2 The Service Implementation Model**

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



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.

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

425 The concrete Service Implementation Model Artifact Types are then:

426 **Organization**

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

431 **ServiceInstance**

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

435 **ServiceEndpoint**

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

440 **ServiceOperation**

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

445 **2.4 Derived Models**

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

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

457 **2.4.1 The XSD Model**

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

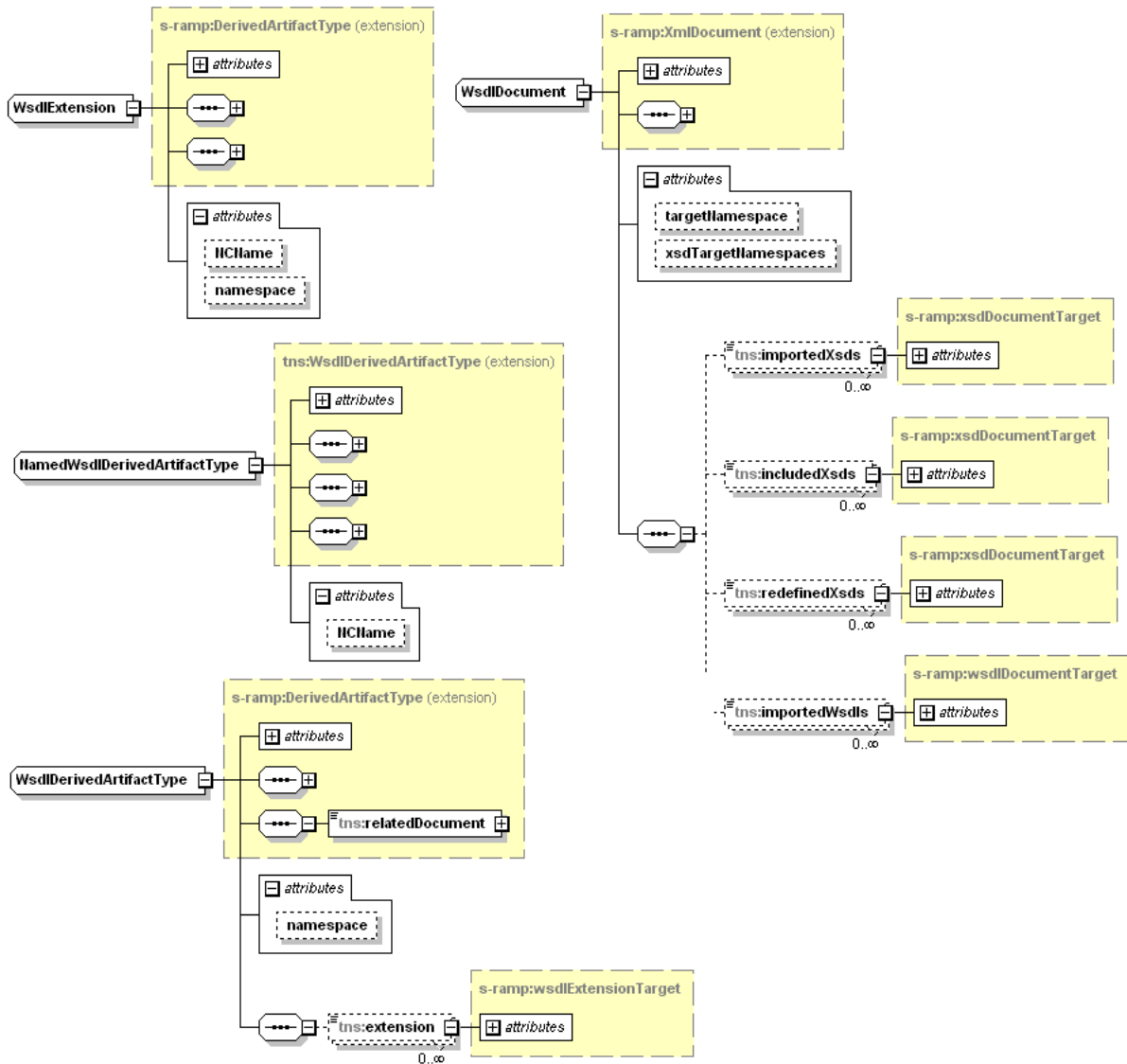
461

475 **2.4.2 The WSDL Model**

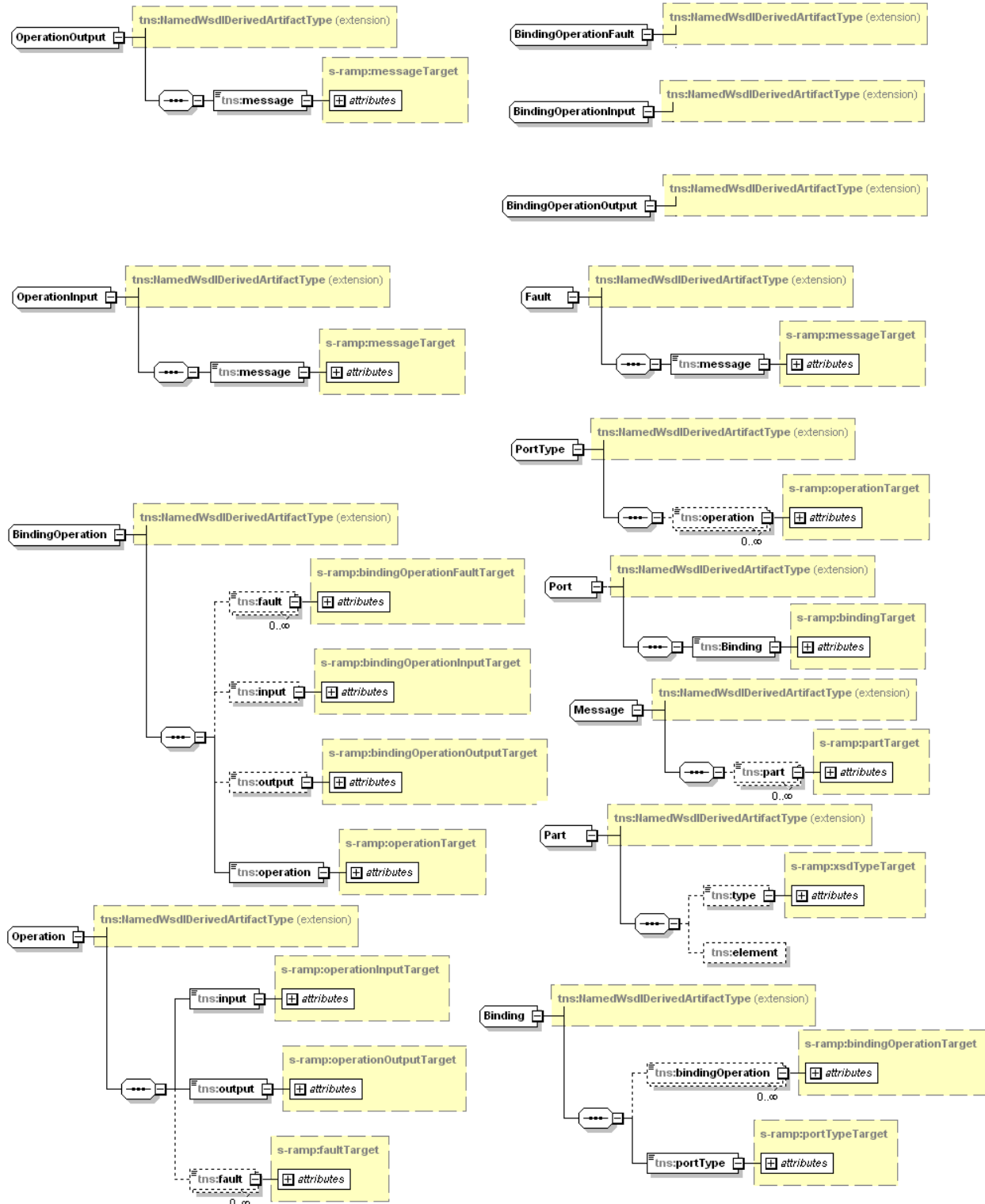
476 The WSDL Model is one of the most complex Derived Models, owing to the complexity of WSDL itself.
 477 The WSDL Model contains substantial richness because of the value in being able to perform highly
 478 refined queries using logical artifact representations of most of a WSDL document's constituent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

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

484



485
 486 *Figure 6: Conceptual Diagram of WSDL Model: Part 1*



487

488 *Figure 7: Conceptual Diagram of WSDL Model: Part 2*

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

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

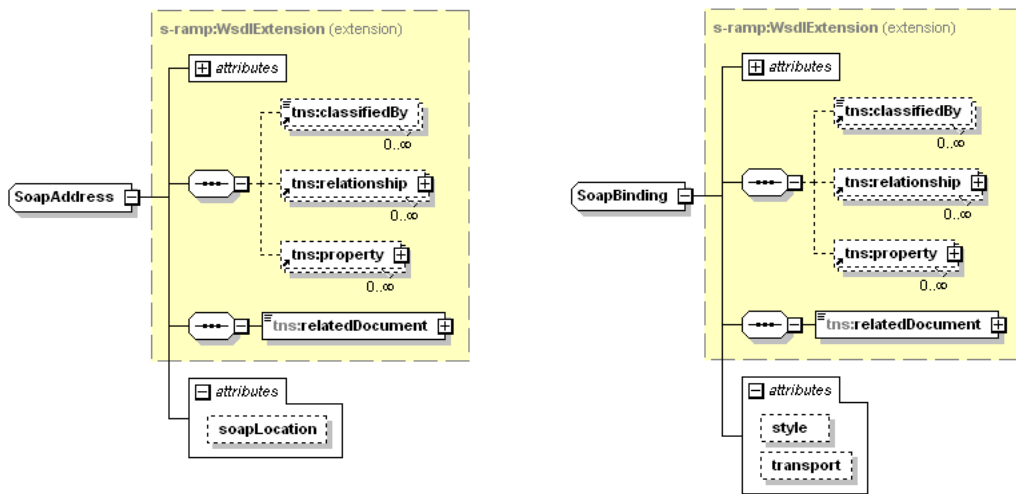
494 The WsdDerivedArtifactType is a modeling convenience from which all WSDL related metadata artifacts
 495 extend. It contains a *namespace* attribute that is the namespace of the WSDL document and it applies to
 496 every Derived Artifact instance for that Document. It also has an *extension* Derived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

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

501 **2.4.3 The SOAPWSDL Model**

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

505

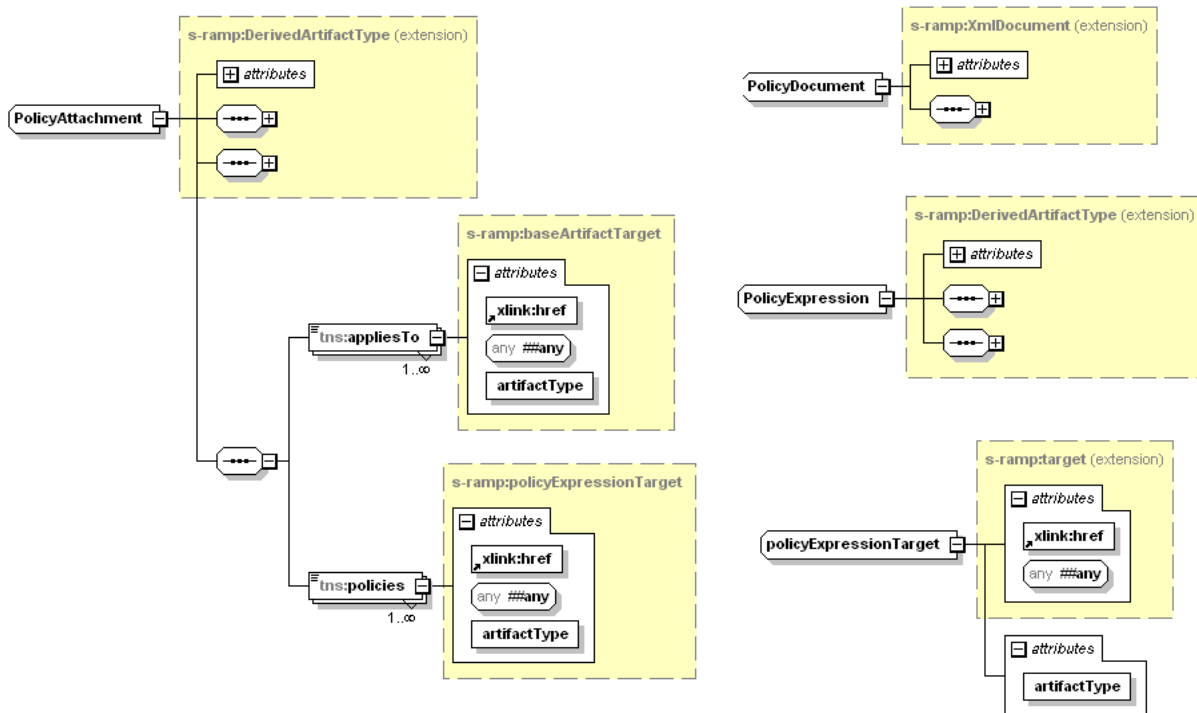


506

507 *Figure 8: Conceptualized Diagram of the SOAP WSDL Model*

508 **2.4.4 The Policy Model**

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



512
513 *Figure 9: Conceptualized Model of Policy Model Artifacts*

514
515 All *appliesTo* relationships are instantiated in the repository based on the content in the policy attachment
516 document. There is no restriction on the artifacts it can reference.

517 2.5 Referencing S-RAMP Artifacts

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

520 2.5.1 Notional Syntax

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

522
523 `/s-ramp/{ArtifactModel}/{ArtifactType}`

524
525 Note that each successive component of this syntax is optional. Specifically

- 526 • A reference of “/s-ramp” refers to all Artifact Types in all models
- 527 • A reference of “/s-ramp/{Artifact Model}” refers to all Artifact Types within the specified Artifact
528 Model.
- 529 • References of the form “://{ArtifactModel}” or “://{ArtifactType}” are also permitted.

530
531 Table 7: Artifact Models and Types below provides the pre-defined values for Artifact Model and Artifact
532 Types. Abstract types are not included since they cannot be instantiated.

533
534 *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

535

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

538

539 *Example 1: Artifact Model and Type References*

- 540 /s-ramp
- 541 o References all Artifact Types in the repository
- 542 /s-ramp/xsd or //xsd
- 543 o References all Artifact Types in the XSD Model (e.g., *XsdDocument*, *XsdType*, ...)
- 544 /s-ramp/xsd/XsdType or //XsdType
- 545 o References the *XsdType* Artifact Type in the XSD Model
- 546 /s-ramp/soa/Service
- 547 o References all *Service* Artifacts Types in the SOA Model (included by reference to
- 548 The Open Group's SOA Ontology)
- 549 /s-ramp/wsd/Port
- 550 o References the *Port* Artifact Type in the WSDL Model

551

552

3 Classification Systems in S-RAMP

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

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

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

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

577

578 ***rdf:ID***

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

585 ***rdf:about***

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

590 ***owl:Ontology***

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

594 ***owl:Imports***

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

599 ***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.

603 *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.

608 *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.

611 *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.

615

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

619 OWL files used in S-RAMP MUST conform to the following rules, which result in ontologies that are self-
620 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

623 The example ontology can be expressed in OWL as follows:

624

625 *Example 2: An OWL Ontology*

```
626 <?xml version="1.0" encoding="UTF-8"?>
627
628 <!DOCTYPE rdf:RDF [
629   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
630   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
631   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
632   <!ENTITY owl "http://www.w3.org/2002/07/owl#">
633   <!ENTITY ns_region "http://www.regions.com/geographicalregion">
634 ]>
635
636 <rdf:RDF
637   xmlns:xsd="&xsd;"
638   xmlns:rdf="&rdf;"
639   xmlns:rdfs="&rdfs;"
640   xmlns:owl="&owl;"
641   xmlns:ns_region="&ns_region;"
642   xml:base="&ns_region;"
643 >
644
645   <!-- ontology -->
646
647   <owl:Ontology rdf:ID="">
648     <rdfs:label>Geographical Regions</rdfs:label>
649     <rdfs:comment>An ontology used to provide classifications describing geographical
650 regions.</rdfs:comment>
651   </owl:Ontology>
652
653   <!-- root class -->
654
655   <owl:Class rdf:ID="World">
656     <rdfs:label>World</rdfs:label>
657     <rdfs:label xml:lang="en">World</rdfs:label>
658     <rdfs:label xml:lang="fr">Monde</rdfs:label>
```



```

659     </owl:Class>
660
661     <!-- sub classes -->
662
663     <owl:Class rdf:ID="Asia">
664       <rdfs:subClassOf rdf:resource="&ns_region;#World"/>
665       <rdfs:label>Asia</rdfs:label>
666     </owl:Class>
667
668     <owl:Class rdf:ID="Europe">
669       <rdfs:subClassOf rdf:resource="&ns_region;#World"/>
670       <rdfs:label>Europe</rdfs:label>
671     </owl:Class>
672
673     <!-- sub sub classes -->
674
675     <owl:Class rdf:ID="Japan">
676       <rdfs:subClassOf rdf:resource="&ns_region;#Asia"/>
677       <rdfs:label>Japan</rdfs:label>
678     </owl:Class>
679
680     <owl:Class rdf:ID="China">
681       <rdfs:subClassOf rdf:resource="&ns_region;#Asia"/>
682       <rdfs:label>China</rdfs:label>
683     </owl:Class>
684
685     <owl:Class rdf:ID="UnitedKingdom">
686       <rdfs:subClassOf rdf:resource="&ns_region;#Europe"/>
687       <rdfs:label>United Kingdom</rdfs:label>
688     </owl:Class>
689
690     <owl:Class rdf:ID="Germany">
691       <rdfs:subClassOf rdf:resource="&ns_region;#Europe"/>
692       <rdfs:label>Germany</rdfs:label>
693     </owl:Class>
694
695 </rdf:RDF>

```

696

697 Some points of interest in the foregoing example:

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

714

4 S-RAMP Query Model

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

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

728 4.1 Query Dialect (XPath2) Context

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

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

733

734 *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	<u>none</u>
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

735

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

738

739 *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”

740

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

743 Additional features and limitations of S-RAMP query support:

- 744 • Uses the XPath2 default Axis of ancestor / child
- 745 • The XPath abbreviated paths listed below are not supported:
 - 746 ○ ‘..’
- 747 • XPath2 Node types are not supported
 - 748 ○ Comments, text and so on are not relevant in an S-RAMP query

749 4.2 Query Expression Predicates

750 The default namespace assumed in an S-RAMP query is [http://docs.oasis-open.org/s-ramp/ns/s-ramp-](http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0)
751 [v1.0](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
752 form:

753

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

755

756 where:

757

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

759

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

762
763 {predicate}
764

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

766
767 When evaluated, the predicate returns true or false, and thus filters the set of artifacts returned from the
768 query. Predicates can use combinations of artifact properties, relationships and classifications. The
769 predicate sets the context for evaluating the query, making it wider or narrower. If a query predicate uses
770 a property which is not defined on an artifact instance whose scope is within the path-expression, then
771 the predicate is false for that artifact and it is not returned from the query. Several examples follow:

772

773 *Example 3: Query Expressions Using Properties*

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

775 o Returns *XsdDocument* artifact instances which have a generic property named
776 "someProperty".

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

778 o Returns *XsdDocument* artifact instances that have an artifact name whose value is
779 "bob".

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

781 o Returns *ServiceInstance* artifact instances having a generic property called
782 *someProperty* whose value equals 'high'.

783 o If the requested property is not defined on the artifact within the set of
784 artifacts selected for search, then the predicate is false and that artifact is
785 therefore not returned.

786

787 *Example 4: Query Expression Using Relationships*

788 `/s-ramp/wsd1/Wsd1Document[includedXsds]`

789 o Returns all the *Wsd1Document* artifact instance(s) representing WSDL documents for
790 which the query predicate evaluates to a non-empty context. The context here
791 consists of the targets of the *includedXsds* relationship; therefore any
792 *Wsd1Document* artifact that has at least one *includedXsds* relationship is returned.

793 o If the relationship in the predicate is not defined on a given artifact within the
794 set of artifacts selected for search, then the predicate expression is false and
795 that artifact is not returned.

796

797 *Example 5: Query Expressions Using Relationships and Properties*

798 `/s-ramp/wsd1/Wsd1Document[includedXsds[@someProperty='true']]`

799 o Only *Wsd1Document* artifact instances corresponding to WSDL documents which have at
800 least one *includedXsds* relationship instance, that has a property name of
801 "someProperty", whose value is "true" will be returned.

802 o If the relationship in the predicate is not defined on a given artifact whose
803 scope is within the path expression, then the predicate evaluates to false and
804 that artifact is not returned.

805

806 *Example 6: Extended Artifacts*

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

808 ○ Returns only BpmnDocument extended artifact instances named 'LoanApproval'.

809

810

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

812 `/s-ramp/wsd1/Message[@name='PurchaseRequestMessage']/part`

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

814 4.3 Query Functions

815 S-RAMP defines a number of its own query functions in addition to using some already defined in XPath
816 2.0 [XPATH]. This includes functions that provide several useful simplifications associated with how an
817 artifact is classified. These query functions use the following syntax:

818

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

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

- 823 • class color
 - 824 ○ class red
 - 825 ○ class white
- 826 • class taste
 - 827 ○ class sour
 - 828 ○ class sweet
 - 829 ○ class spicy

830

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

- 832 • name = "Bread" classifiedBy = ["white", "sweet"]
- 833 • name = "Wine" classifiedBy = ["red", "sweet"]
- 834 • name = "Chili" classifiedBy = ["red", "spicy"]

835

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

838 *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: <code>classifiedByAnyOf({artifact}, {classifiedBy_1}, {classifiedBy_2},...)</code>

	<p>Using the classification example setup preceding this table, this example returns all 3 <i>Wsd/Service</i> artifact instances:</p> <pre>/s-ramp/wsdl/WsdService[classifiedByAnyOf(., 'taste')]</pre> <p>This example returns the Wine and Chili <i>Wsd/Service</i> artifact instances:</p> <pre>/s-ramp/wsdl/WsdService[classifiedByAnyOf(., 'red', 'spicy')]</pre>
s-ramp:classifiedByAllOf	<p>Returns all artifact instances classified by every one of the specified OWL URIs or their subtypes.</p> <p>Syntax: <code>classifiedByAllOf({artifact}, {classifiedBy_1}, {classifiedBy_2},...)</code></p> <p>This example returns all 3 <i>Wsd/Service</i> artifact instances:</p> <pre>/s-ramp/wsdl/WsdService[classifiedByAllOf(., 'taste')]</pre> <p>This example returns only the Wine <i>Wsd/Service</i> artifact instance:</p> <pre>/s-ramp/wsdl/WsdService[classifiedByAllOf(., 'red', 'sweet')]</pre>
s-ramp:exactlyClassifiedByAnyOf	<p>Returns all artifact instances classified by at least one of the specified OWL URIs. Subtypes are not considered:</p> <p>Syntax: <code>exactlyClassifiedByAnyOf({artifact}, {classifiedBy_1}, {classifiedBy_2},...)</code></p> <p>This example returns none of the <i>Wsd/Service</i> artifact instances:</p> <pre>/s-ramp/wsdl/WsdService[exactlyClassifiedByAnyOf(., 'taste')]</pre> <p>This example returns only the Wine and Chili <i>Wsd/Service</i> artifact instances:</p> <pre>/s-ramp/wsdl/WsdService[exactlyClassifiedByAnyOf(., 'red', 'taste')]</pre>
s-ramp:exactlyClassifiedByAllOf	<p>Returns all artifact instances classified by every one of the specified OWL URIs. Subtypes are not considered:</p> <p>Syntax: <code>exactlyClassifiedByAllOf({artifact}, {classifiedBy_1}, {classifiedBy_2},...)</code></p> <p>This example returns none of the <i>Wsd/Service</i> artifact instances:</p> <pre>/s-ramp/wsdl/WsdService[exactlyClassifiedByAllOf(., 'taste')]</pre> <p>This example returns only the Wine <i>Wsd/Service</i> artifact instance:</p> <pre>/s-ramp/wsdl/WsdService[exactlyClassifiedByAllOf(., 'red', 'sweet')]</pre>

xp2:matches	<p>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.</p> <p style="text-align: center;">Syntax: matches({value}, {expression})</p> <p>The expression may use wildcards, but only in the form ‘.*’</p> <p>This example returns all artifact instances (of any Artifact Type) from the Service Implementation Model, whose artifact <i>name</i> matches “.*account.*”, and a user-defined <i>version</i> property whose value is ‘1’.</p> <pre>/s-ramp/serviceImplementation[xp2:matches(@name, ‘.*account.* ‘ and @version = ‘1’]</pre>
xp2:not	<p>This is the XPath 2.0 “not” (negation) function, which accepts an xsd:boolean as its input and returns the inversion.</p> <p style="text-align: center;">Syntax: not({boolean-expression})</p> <p>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.</p> <pre>/s-ramp/core/Document[xp2:not (@user-property)] /s-ramp/wsd1/Part[xp2:not (element)]</pre> <p>Note: to invert a property value, simply use the ‘!=’ operator rather than the ‘not’ function.</p>

839 4.4 Query Grammar

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

842

843 **QName ::=**

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

845

846 **s-ramp query ::=**

847 artifact -set

848 | artifact -set ‘[’ predicate ‘]’ |

849 | artifact -set ‘[’ predicate ‘]’ ‘/’ subartifact-set

850

851 **artifact- set ::=**

852 location-path

853

854 **location-path::=**
855 /s-ramp
856 |/s-ramp/<artifact-model>
857 | /s-ramp/<artifact-model>/<artifact-type>
858 |//<artifact-model>
859 |//<artifact-type>
860 **subartifact-set::=**
861 relationship-path
862 | relationship-path '[' predicate ']
863 | relationship-path '[' predicate ']' '/' subartifact-set
864 | FunctionCall
865
866 **relationship-path::=**
867 any-outgoing-relationship
868 | <s-ramp-relationship-type>
869
870 **any-outgoing-relationship::=**
871 *outgoing*
872
873 **Predicate::=**
874 Expr
875
876 **Expr::=**
877 AndExpr
878
879 **AndExpr::=**
880 OrExpr
881 | AndExpr 'and' OrExpr
882
883 **OrExpr::=**
884 EqualityExpr
885 | OrExpr 'or' EqualityExpr
886
887 **EqualityExpr::=**
888 subartifact-set
889 |ForwardPropertyStep
890 | ForwardPropertyStep '=' PrimaryExpr
891 | ForwardPropertyStep '!=' PrimaryExpr
892 | ForwardPropertyStep '<' PrimaryExpr
893 | ForwardPropertyStep '>' PrimaryExpr
894 | ForwardPropertyStep '<=' PrimaryExpr
895 | ForwardPropertyStep '>=' PrimaryExpr
896 | '(' Expr ')'
897

898 **PropertyQName ::=**
899 QName
900

901 **PrimaryExpr ::=**
902 Literal
903 | Number
904 | '\$' <PropertyQName>
905

906 **ForwardPropertyStep ::=**
907 Subartifact-set '/' '@<PropertyQName>|@<PropertyQName>
908

909 **FunctionCall ::=**
910 FunctionName '(' (Argument (',' Argument) *)? ')'
911

912 **Argument ::=**
913 PrimaryExpr
914 | Expr
915

916 **Literal ::=**
917 ''' [^']* ''' | "" [^"]* ""
918

919 **Number ::=**
920 Digits ('.' Digits)? | '.' Digits
921

922 **Digits ::=**
923 [0-9]+
924

925 **FunctionName ::=**
926 QName – NodeType
927

928 **NodeType ::=**
929 'comment'
930 | 'text'
931 | 'processing-instruction'
932 | 'node'

933 **4.5 Stored Queries**

934 S-RAMP provides support for storing queries in the repository using the *StoredQuery* Artifact Type. This
935 can be convenient because it allows quick execution of a frequently performed query. The syntax
936 associated with creation, retrieval, update and deletion of a Stored Query is binding specific. Refer to the
937 appropriate binding document of this specification for details.
938

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.

949 5 Conformance

950 5.1 Introduction

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

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

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

957

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

959

960 This specification addresses the following aspects of conformance:

- 961 • Data Model
- 962 • Classification Systems
- 963 • Query Model

964 5.2 Data Model

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

967

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

969

970 *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: WsdIDocument	M
WSDL Model: WsdService	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

972 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.

977 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

986 Appendix A. Acknowledgments

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

989 **Participants:**

990 Joel Fleck II, Hewlett-Packard
991 Jishnu Mukerji, Hewlett-Packard
992 Radek Pospisil, Hewlett-Packard
993 Vincent Brunssen, IBM
994 John Colgrave, IBM
995 Diane Jordan, IBM
996 Bernard Kuflik, IBM
997 Kelvin Lawrence, IBM
998 Martin Smithson, IBM
999 Gershon Janssen, Individual
1000 Carl Mattocks, Individual
1001 Rex Brooks, Network Centric Operations Industry Consortium
1002 jian zhang, Primeton Technologies, Inc.
1003 Randall Hauch, Red Hat
1004 Kurt Stam, Red Hat
1005 Eric Wittmann, Red Hat
1006 Steve Fanshier, Software AG, Inc.
1007 Gary Woods, Software AG, Inc.
1008 Prasad Yendluri, Software AG, Inc.
1009 Eric Johnson, TIBCO Software Inc.
1010 Senaka Fernando, WSO2
1011 Paul Fremantle, WSO2
1012 Jonathan Marsh, WSO2

1013 **Appendix B. Non-Normative Text**

1014 This specification provides no additional non-normative information at this time. It is expected that best
1015 practices for S-RAMP Repositories will emerge over time as this specification is adopted by vendors and
1016 users.

1017

Appendix C. Revision History

1018

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

1019

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.

Appendix E. Core Model Schema

1022

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

1024

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

1025

1026

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

1027

```
<xsd:schema targetNamespace="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
```

1028

```
version="1.0" xmlns:tns="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0" xmlns:s-
```

1029

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

1030

```
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

1031

```
xmlns:xlink="http://www.w3.org/1999/xlink" elementFormDefault="qualified">
```

1032

```
<!--
```

1033

```
(c) 2010 Hewlett-Packard Company (HP), International Business Machines
```

1034

```
Corporation (IBM), Software AG (SAG) and TIBCO Software Inc. All
```

1035

```
rights reserved. Permission to copy and display the SOA Repository
```

1036

```
Artifact Model and Protocol (the "Specification"), in any medium
```

1037

```
without fee or royalty is hereby granted by Hewlett-Packard Company
```

1038

```
(HP), International Business Machines Corporation (IBM), Software AG
```

1039

```
(SAG) and TIBCO Software Inc. (collectively, the "Authors"), provided
```

1040

```
that you include the following on ALL copies of this document or
```

1041

```
portions thereof, that you make:
```

1042

```
1. A link or URL to this document at this location:
```

1043

```
http://s-ramp.org/2010/s-ramp/specification/documents/{this document
```

1044

```
name}
```

1045

```
2. The copyright notice as shown in the Specification.
```

1046

1047

```
The Authors each agree to grant you a royalty-free license, under
```

1048

```
reasonable, non-discriminatory terms and conditions to their
```

1049

```
respective patents that they deem necessary to implement the "SOA
```

1050

```
Repository Artifact Model and Protocol" Specification, including all
```

1051

```
its constituent documents. THIS DOCUMENT IS PROVIDED "AS IS," AND THE
```

1052

```
AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED,
```

1053

```
INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS
```

1054

```
FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE
```

1055

```
CONTENTS OF THIS DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE
```

1056

```
IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY
```

1057

```
PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. THE AUTHORS WILL NOT
```

1058

```
BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR
```

1059

```
CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR
```

1060

```
DISTRIBUTION OF THIS DOCUMENT.
```

1061

```
-->
```

1062

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

1063

```
<!-- Core data types: -->
```

1064

```
<xsd:simpleType name="baseArtifactEnum">
```

1065

```
<xsd:restriction base="xsd:string">
```

1066

```
<xsd:enumeration value="BaseArtifactType" />
```

1067

```
<xsd:enumeration value="DocumentArtifactType" />
```

1068

```
<xsd:enumeration value="Document" />
```

1069

```
<xsd:enumeration value="XMLDocument" />
```

1070

```
<!-- xsd model -->
```

1071

1072

1073

```

1074 <xsd:enumeration value="XsdDocument" />
1075 <!-- wsdl model -->
1076 <xsd:enumeration value="WsdDocument" />
1077 <!-- policy model -->
1078 <xsd:enumeration value="PolicyDocument" />
1079 <!-- derived artifacts -->
1080 <xsd:enumeration value="DerivedArtifactType" />
1081 <!-- policy model -->
1082 <xsd:enumeration value="PolicyAttachment" />
1083 <xsd:enumeration value="PolicyExpression" />
1084 <!-- xsd model -->
1085 <xsd:enumeration value="AttributeDeclaration" />
1086 <xsd:enumeration value="ElementDeclaration" />
1087 <xsd:enumeration value="XsdType" />
1088 <xsd:enumeration value="ComplexTypeDeclaration" />
1089 <xsd:enumeration value="SimpleTypeDeclaration" />
1090 <!-- wsdl model -->
1091 <xsd:enumeration value="WsdDerivedArtifactType" />
1092 <xsd:enumeration value="NamedWsdDerivedArtifactType" />
1093 <xsd:enumeration value="WsdService" />
1094 <xsd:enumeration value="Port" />
1095 <xsd:enumeration value="Binding" />
1096 <xsd:enumeration value="PortType" />
1097 <xsd:enumeration value="BindingOperation" />
1098 <xsd:enumeration value="BindingOperationInput" />
1099 <xsd:enumeration value="BindingOperationFault" />
1100 <xsd:enumeration value="Operation" />
1101 <xsd:enumeration value="OperationInput" />
1102 <xsd:enumeration value="Fault" />
1103 <xsd:enumeration value="Message" />
1104 <xsd:enumeration value="Part" />
1105 <xsd:enumeration value="BindingOperationOutput" />
1106 <xsd:enumeration value="OperationOutput" />
1107 <xsd:enumeration value="WsdExtension" />
1108 <!-- soapwsdl model -->
1109 <xsd:enumeration value="SoapAddress" />
1110 <xsd:enumeration value="SoapBinding" />
1111 <!-- extended artifacts -->
1112 <xsd:enumeration value="ExtendedArtifactType" />
1113 <xsd:enumeration value="ExtendedDocument" />
1114 <!-- soa model -->
1115 <xsd:enumeration value="SoaModelType" />
1116 <xsd:enumeration value="ServiceInterface" />
1117 <xsd:enumeration value="Effect" />
1118 <xsd:enumeration value="Event" />
1119 <xsd:enumeration value="InformationType" />
1120 <xsd:enumeration value="Policy" />
1121 <xsd:enumeration value="PolicySubject" />
1122 <xsd:enumeration value="ServiceContract" />
1123 <xsd:enumeration value="Element" />
1124 <xsd:enumeration value="Service" />
1125 <xsd:enumeration value="Actor" />
1126 <xsd:enumeration value="Organization" />
1127 <xsd:enumeration value="System" />
1128 <xsd:enumeration value="Task" />
1129 <xsd:enumeration value="Composition" />

```

```

1130     <xsd:enumeration value="ServiceComposition" />
1131     <xsd:enumeration value="Choreography" />
1132     <xsd:enumeration value="Collaboration" />
1133     <xsd:enumeration value="Orchestration" />
1134     <xsd:enumeration value="Process" />
1135     <xsd:enumeration value="ChoreographyProcess" />
1136     <xsd:enumeration value="CollaborationProcess" />
1137     <xsd:enumeration value="OrchestrationProcess" />
1138     <!-- serviceimplementation model -->
1139     <xsd:enumeration value="ServiceImplementationModelType" />
1140     <xsd:enumeration value="ServiceInstance" />
1141     <xsd:enumeration value="ServiceOperation" />
1142     <xsd:enumeration value="ServiceEndpoint" />
1143     <xsd:enumeration value="ServiceOperation" />
1144 </xsd:restriction>
1145 </xsd:simpleType>
1146
1147 <xsd:simpleType name="derivedArtifactEnum">
1148   <xsd:restriction base="s-ramp:baseArtifactEnum">
1149     <xsd:enumeration value="DerivedArtifactType" />
1150     <!-- policy model -->
1151     <xsd:enumeration value="PolicyAttachment" />
1152     <xsd:enumeration value="PolicyExpression" />
1153     <!-- xsd model -->
1154     <xsd:enumeration value="AttributeDeclaration" />
1155     <xsd:enumeration value="ElementDeclaration" />
1156     <xsd:enumeration value="XsdType" />
1157     <xsd:enumeration value="ComplexTypeDeclaration" />
1158     <xsd:enumeration value="SimpleTypeDeclaration" />
1159     <!-- wsdl model -->
1160     <xsd:enumeration value="WsdLDerivedArtifactType" />
1161     <xsd:enumeration value="NamedWsdLDerivedArtifactType" />
1162     <xsd:enumeration value="WsdLService" />
1163     <xsd:enumeration value="Port" />
1164     <xsd:enumeration value="Binding" />
1165     <xsd:enumeration value="PortType" />
1166     <xsd:enumeration value="BindingOperation" />
1167     <xsd:enumeration value="BindingOperationInput" />
1168     <xsd:enumeration value="BindingOperationFault" />
1169     <xsd:enumeration value="Operation" />
1170     <xsd:enumeration value="OperationInput" />
1171     <xsd:enumeration value="Fault" />
1172     <xsd:enumeration value="Message" />
1173     <xsd:enumeration value="Part" />
1174     <xsd:enumeration value="BindingOperationOutput" />
1175     <xsd:enumeration value="OperationOutput" />
1176     <xsd:enumeration value="WsdLExtension" />
1177     <!-- soapwsdl model -->
1178     <xsd:enumeration value="SoapAddress" />
1179     <xsd:enumeration value="SoapBinding" />
1180   </xsd:restriction>
1181 </xsd:simpleType>
1182
1183 <xsd:simpleType name="documentArtifactEnum">
1184   <xsd:restriction base="s-ramp:baseArtifactEnum">
1185     <xsd:enumeration value="DocumentArtifactType" />

```

```

1186     <xsd:enumeration value="Document" />
1187     <xsd:enumeration value="XmlDocument" />
1188     <xsd:enumeration value="ExtendedDocument" />
1189     <!-- xsd model -->
1190     <xsd:enumeration value="XsdDocument" />
1191     <!-- wsdl model -->
1192     <xsd:enumeration value="WsdlDocument" />
1193     <!-- policy model -->
1194     <xsd:enumeration value="PolicyDocument" />
1195 </xsd:restriction>
1196 </xsd:simpleType>
1197
1198 <!-- Base type for almost all Artifacts in S-RAMP. Most other types in S-RAMP
1199 extend this one. Extensions of BaseArtifactType are limited to attributes. -->
1200 <xsd:complexType abstract="true" name="BaseArtifactType">
1201     <xsd:sequence>
1202         <xsd:element ref="tns:classifiedBy" minOccurs="0" maxOccurs="unbounded" />
1203         <xsd:element ref="tns:relationship" minOccurs="0" maxOccurs="unbounded" />
1204         <xsd:element ref="tns:property" minOccurs="0" maxOccurs="unbounded" />
1205     </xsd:sequence>
1206     <xsd:attribute name="artifactType" type="s-ramp:baseArtifactEnum" use="required"
1207 />
1208     <xsd:attribute name="name" type="xsd:string" use="required" />
1209     <xsd:attribute name="description" type="xsd:string" use="optional" />
1210     <xsd:attribute name="createdBy" type="xsd:string" use="required" />
1211     <xsd:attribute name="version" type="xsd:string" use="optional" />
1212     <xsd:attribute name="uuid" type="xsd:string" use="required" />
1213     <xsd:attribute name="createdTimestamp" type="xsd:dateTime" use="required" />
1214     <xsd:attribute name="lastModifiedTimestamp" type="xsd:dateTime" use="required" />
1215     <xsd:attribute name="lastModifiedBy" type="xsd:string" use="required" />
1216     <xsd:anyAttribute namespace="##any" />
1217 </xsd:complexType>
1218
1219 <!-- Base type for all Derived Artifacts in S-RAMP -->
1220 <xsd:complexType abstract="true" name="DerivedArtifactType">
1221     <xsd:complexContent>
1222         <xsd:extension base="s-ramp:BaseArtifactType">
1223             <xsd:sequence>
1224                 <xsd:element name="relatedDocument" type="tns:documentArtifactTarget"
1225 minOccurs="1" maxOccurs="1" />
1226             </xsd:sequence>
1227         </xsd:extension>
1228     </xsd:complexContent>
1229 </xsd:complexType>
1230 <xsd:element name="DerivedArtifactType" type="tns:DerivedArtifactType" />
1231
1232 <!-- Base type for all Documents in S-RAMP -->
1233 <xsd:complexType abstract="true" name="DocumentArtifactType">
1234     <xsd:complexContent>
1235         <xsd:extension base="s-ramp:BaseArtifactType">
1236             <xsd:attribute name="contentType" type="xsd:string" />
1237             <xsd:attribute name="contentSize" type="xsd:Long" />
1238             <xsd:attribute name="contentHash" type="xsd:string" />
1239             <xsd:anyAttribute namespace="##any" />
1240         </xsd:extension>
1241     </xsd:complexContent>

```



```

1242 </xsd:complexType>
1243
1244 <!-- Document type implements DocumentArtifactType -->
1245 <xsd:complexType name="Document">
1246   <xsd:complexContent>
1247     <xsd:extension base="s-ramp:DocumentArtifactType" />
1248   </xsd:complexContent>
1249 </xsd:complexType>
1250
1251 <!-- Base Type for all XML documents. Specific document types extend XmlDocument --
1252 >
1253 <xsd:complexType name="XmlDocument">
1254   <xsd:complexContent>
1255     <xsd:extension base="s-ramp:DocumentArtifactType">
1256       <xsd:attribute name="contentEncoding" type="xsd:string" />
1257     </xsd:extension>
1258   </xsd:complexContent>
1259 </xsd:complexType>
1260
1261 <!-- Types used for Extended Artifact Models in S-RAMP -->
1262 <xsd:complexType name="ExtendedArtifactType">
1263   <xsd:complexContent>
1264     <xsd:extension base="s-ramp:BaseArtifactType">
1265       <xsd:attribute name="extendedType" type="xsd:string" />
1266     </xsd:extension>
1267   </xsd:complexContent>
1268 </xsd:complexType>
1269 <xsd:complexType name="ExtendedDocument">
1270   <xsd:complexContent>
1271     <xsd:extension base="s-ramp:DocumentArtifactType">
1272       <xsd:attribute name="extendedType" type="xsd:string" />
1273     </xsd:extension>
1274   </xsd:complexContent>
1275 </xsd:complexType>
1276
1277 <!-- Relationship target artifact's UUID. Used by all types of relationships -->
1278 <xsd:complexType name="target">
1279   <xsd:simpleContent>
1280     <xsd:extension base="xsd:string">
1281       <xsd:attribute ref="xLink:href" use="required" />
1282       <xsd:anyAttribute namespace="##any" />
1283     </xsd:extension>
1284   </xsd:simpleContent>
1285 </xsd:complexType>
1286
1287 <!-- Relationship referencing the artifact's UUID, to reference any BaseArtifact. -
1288 ->
1289 <xsd:complexType name="baseArtifactTarget">
1290   <xsd:complexContent>
1291     <xsd:extension base="s-ramp:target">
1292       <xsd:attribute name="artifactType" type="s-ramp:baseArtifactEnum"
1293 use="required" />
1294     </xsd:extension>
1295   </xsd:complexContent>
1296 </xsd:complexType>
1297

```

```

1298     <!-- Relationship referencing the artifact's UUID, to reference any
1299     DerivedArtifact. -->
1300     <xsd:complexType name="derivedArtifactTarget">
1301         <xsd:complexContent>
1302             <xsd:extension base="s-ramp:target">
1303                 <xsd:attribute name="artifactType" type="s-ramp:derivedArtifactEnum"
1304 use="required" />
1305             </xsd:extension>
1306         </xsd:complexContent>
1307     </xsd:complexType>
1308
1309     <!-- Relationship referencing the artifact's UUID, to reference any
1310     DocumentArtifact. -->
1311     <xsd:complexType name="documentArtifactTarget">
1312         <xsd:complexContent>
1313             <xsd:extension base="s-ramp:target">
1314                 <xsd:attribute name="artifactType" type="s-ramp:documentArtifactEnum"
1315 use="required" />
1316             </xsd:extension>
1317         </xsd:complexContent>
1318     </xsd:complexType>
1319
1320     <!-- Global Element Declarations -->
1321
1322     <!-- Relationship element used for all GENERIC (user-defined) Relationships -->
1323     <xsd:element name="relationship">
1324         <xsd:complexType>
1325             <xsd:sequence>
1326                 <xsd:element ref="tns:relationshipType" minOccurs="1" maxOccurs="1" />
1327                 <xsd:element name="relationshipTarget" type="tns:target" minOccurs="0"
1328 maxOccurs="unbounded" />
1329             </xsd:sequence>
1330             <xsd:anyAttribute namespace="##any" />
1331         </xsd:complexType>
1332     </xsd:element>
1333
1334     <!-- Stored Queries Artifact element -->
1335     <xsd:element name="StoredQuery">
1336         <xsd:complexType>
1337             <xsd:sequence>
1338                 <xsd:element ref="tns:propertyName" minOccurs="0" maxOccurs="unbounded" />
1339                 <xsd:element ref="tns:queryExpression" />
1340             </xsd:sequence>
1341             <xsd:attribute name="queryName" type="xsd:string" />
1342             <xsd:anyAttribute namespace="##any" />
1343         </xsd:complexType>
1344     </xsd:element>
1345
1346     <!-- Property -->
1347     <xsd:element name="property">
1348         <xsd:complexType>
1349             <xsd:sequence>
1350                 <xsd:element ref="tns:propertyName" minOccurs="1" maxOccurs="1" />
1351                 <xsd:element ref="tns:propertyValue" minOccurs="1" maxOccurs="1" />
1352             </xsd:sequence>
1353             <xsd:anyAttribute namespace="##any" />

```



```
1354     </xsd:complexType>
1355 </xsd:element>
1356
1357 <xsd:element name="relationshipType" type="xsd:string" />
1358 <xsd:element name="classifiedBy" type="xsd:anyURI" />
1359 <xsd:element name="propertyName" type="xsd:string" />
1360 <xsd:element name="propertyValue" type="xsd:string" />
1361 <xsd:element name="queryExpression" type="xsd:string" />
1362 <!-- The sourceId and targetId contain the UUID's corresponding to a relationship's
1363 source and target -->
1364 <xsd:element name="sourceId" type="xsd:string" />
1365 <xsd:element name="targetId" type="xsd:string" />
1366
1367 </xsd:schema>
1368
```

1369

Appendix F. SOA Model Schema

1370 The S-RAMP SOA Model Schema XSD file is also provided at:

1371 <http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0/>

1372

```
1373 <?xml version="1.0" encoding="UTF-8"?>
1374 <xsd:schema xmlns:tns="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0" xmlns:s-
1375 ramp="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
1376 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1377 targetNamespace="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
1378 elementFormDefault="qualified" version="1.0">
1379 <!--
```

```
1380 (c) 2010 Hewlett-Packard Company (HP), International Business Machines
1381 Corporation (IBM), Software AG (SAG) and TIBCO Software Inc. All
1382 rights reserved. Permission to copy and display the SOA Repository
1383 Artifact Model and Protocol (the "Specification"), in any medium
1384 without fee or royalty is hereby granted by Hewlett-Packard Company
1385 (HP), International Business Machines Corporation (IBM), Software AG
1386 (SAG) and TIBCO Software Inc. (collectively, the "Authors"), provided
1387 that you include the following on ALL copies of this document or
1388 portions thereof, that you make:
```

- ```
1389
1390 1. A link or URL to this document at this location:
1391 http://s-ramp.org/2010/s-ramp/specification/documents/{this document
1392 name}
1393 2. The copyright notice as shown in the Specification.
```

```
1394
1395 The Authors each agree to grant you a royalty-free license, under
1396 reasonable, non-discriminatory terms and conditions to their
1397 respective patents that they deem necessary to implement the "SOA
1398 Repository Artifact Model and Protocol" Specification, including all
1399 its constituent documents. THIS DOCUMENT IS PROVIDED "AS IS," AND THE
1400 AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED,
1401 INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS
1402 FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE
1403 CONTENTS OF THIS DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE
1404 IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY
1405 PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. THE AUTHORS WILL NOT
1406 BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR
1407 CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR
1408 DISTRIBUTION OF THIS DOCUMENT.
```

```
1409 -->
```

```
1410
1411 <xsd:include schemaLocation="coremodel.xsd" />
1412 <xsd:include schemaLocation="serviceimplementationmodel.xsd" />
1413
1414 <xsd:simpleType name="informationTypeEnum">
1415 <xsd:restriction base="s-ramp:baseArtifactEnum">
1416 <xsd:enumeration value="InformationType" />
1417 </xsd:restriction>
1418 </xsd:simpleType>
1419 <xsd:simpleType name="serviceEnum">
1420 <xsd:restriction base="s-ramp:baseArtifactEnum">
```

```

1421 <xsd:enumeration value="Service" />
1422 </xsd:restriction>
1423 </xsd:simpleType>
1424 <xsd:simpleType name="serviceContractEnum">
1425 <xsd:restriction base="s-ramp:baseArtifactEnum">
1426 <xsd:enumeration value="ServiceContract" />
1427 </xsd:restriction>
1428 </xsd:simpleType>
1429 <xsd:simpleType name="serviceInterfaceEnum">
1430 <xsd:restriction base="s-ramp:baseArtifactEnum">
1431 <xsd:enumeration value="ServiceInterface" />
1432 </xsd:restriction>
1433 </xsd:simpleType>
1434 <xsd:simpleType name="policySubjectEnum">
1435 <xsd:restriction base="s-ramp:baseArtifactEnum">
1436 <xsd:enumeration value="PolicySubject" />
1437 </xsd:restriction>
1438 </xsd:simpleType>
1439 <xsd:simpleType name="taskEnum">
1440 <xsd:restriction base="s-ramp:baseArtifactEnum">
1441 <xsd:enumeration value="Task" />
1442 </xsd:restriction>
1443 </xsd:simpleType>
1444 <xsd:simpleType name="policyEnum">
1445 <xsd:restriction base="s-ramp:baseArtifactEnum">
1446 <xsd:enumeration value="Policy" />
1447 </xsd:restriction>
1448 </xsd:simpleType>
1449 <xsd:simpleType name="elementEnum">
1450 <xsd:restriction base="s-ramp:baseArtifactEnum">
1451 <xsd:enumeration value="Element" />
1452 </xsd:restriction>
1453 </xsd:simpleType>
1454 <xsd:simpleType name="orchestrationEnum">
1455 <xsd:restriction base="s-ramp:baseArtifactEnum">
1456 <xsd:enumeration value="Orchestration" />
1457 </xsd:restriction>
1458 </xsd:simpleType>
1459 <xsd:simpleType name="orchestrationProcessEnum">
1460 <xsd:restriction base="s-ramp:baseArtifactEnum">
1461 <xsd:enumeration value="OrchestrationProcess" />
1462 </xsd:restriction>
1463 </xsd:simpleType>
1464 <xsd:simpleType name="eventEnum">
1465 <xsd:restriction base="s-ramp:baseArtifactEnum">
1466 <xsd:enumeration value="Event" />
1467 </xsd:restriction>
1468 </xsd:simpleType>
1469 <xsd:simpleType name="actorEnum">
1470 <xsd:restriction base="s-ramp:baseArtifactEnum">
1471 <xsd:enumeration value="Actor" />
1472 </xsd:restriction>
1473 </xsd:simpleType>
1474 <xsd:simpleType name="effectEnum">
1475 <xsd:restriction base="s-ramp:baseArtifactEnum">
1476 <xsd:enumeration value="Effect" />

```

```

1477 </xsd:restriction>
1478 </xsd:simpleType>
1479
1480 <xsd:complexType abstract="true" name="SoaModelType">
1481 <xsd:complexContent>
1482 <xsd:extension base="s-ramp:BaseArtifactType">
1483 <xsd:sequence>
1484 <!-- Modeled Relationship to abstract DocumentArtifactType -->
1485 <xsd:element minOccurs="0" name="documentation" type="s-
1486 ramp:documentArtifactTarget" maxOccurs="unbounded" />
1487 </xsd:sequence>
1488 </xsd:extension>
1489 </xsd:complexContent>
1490 </xsd:complexType>
1491
1492 <xsd:complexType name="ServiceInterface">
1493 <xsd:complexContent>
1494 <xsd:extension base="tns:SoaModelType">
1495 <xsd:sequence>
1496 <!-- Modeled Relationship to abstract DerivedArtifactType: -->
1497 <xsd:element minOccurs="0" maxOccurs="1" name="interfaceDefinedBy" type="s-
1498 ramp:DerivedArtifactType" />
1499 <!-- Modeled Relationship to ServiceOperation: -->
1500 <xsd:element minOccurs="0" name="hasOperation" type="s-
1501 ramp:serviceOperationTarget" />
1502 <!-- Modeled Relationship to InformationType: -->
1503 <xsd:element minOccurs="0" name="hasOutput" type="s-
1504 ramp:informationTypeTarget" maxOccurs="unbounded" />
1505 <!-- Modeled Relationship to InformationType: -->
1506 <xsd:element minOccurs="0" name="hasInput" type="s-
1507 ramp:informationTypeTarget" maxOccurs="unbounded" />
1508 <!-- Modeled Relationship to Service: -->
1509 <xsd:element minOccurs="0" name="isInterfaceOf" type="s-ramp:serviceTarget"
1510 maxOccurs="unbounded" />
1511 </xsd:sequence>
1512 </xsd:extension>
1513 </xsd:complexContent>
1514 </xsd:complexType>
1515
1516 <xsd:complexType name="Service">
1517 <xsd:complexContent>
1518 <xsd:extension base="tns:Element">
1519 <xsd:sequence>
1520 <!-- Modeled Relationship to ServiceContract: -->
1521 <xsd:element minOccurs="0" name="hasContract" type="s-
1522 ramp:serviceContractTarget" maxOccurs="unbounded" />
1523 <!-- Modeled Relationship to ServiceInterface: -->
1524 <xsd:element minOccurs="1" name="hasInterface" type="s-
1525 ramp:serviceInterfaceTarget" maxOccurs="unbounded" />
1526 <!-- Modeled Relationship to ServiceInstance: -->
1527 <xsd:element minOccurs="0" maxOccurs="1" name="hasInstance" type="s-
1528 ramp:serviceInstanceTarget" />
1529 </xsd:sequence>
1530 </xsd:extension>
1531 </xsd:complexContent>
1532 </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" />

```

```

1645 </xsd:complexContent>
1646 </xsd:complexType>
1647
1648 <xsd:complexType name="Collaboration">
1649 <xsd:complexContent>
1650 <xsd:extension base="tns:Composition" />
1651 </xsd:complexContent>
1652 </xsd:complexType>
1653
1654 <xsd:complexType name="Orchestration">
1655 <xsd:complexContent>
1656 <xsd:extension base="tns:Composition" />
1657 </xsd:complexContent>
1658 </xsd:complexType>
1659
1660 <xsd:complexType name="Process">
1661 <xsd:complexContent>
1662 <xsd:extension base="tns:Composition" />
1663 </xsd:complexContent>
1664 </xsd:complexType>
1665
1666 <xsd:complexType name="ChoreographyProcess">
1667 <xsd:complexContent>
1668 <xsd:extension base="tns:Process" />
1669 </xsd:complexContent>
1670 </xsd:complexType>
1671
1672 <xsd:complexType name="CollaborationProcess">
1673 <xsd:complexContent>
1674 <xsd:extension base="tns:Process" />
1675 </xsd:complexContent>
1676 </xsd:complexType>
1677
1678 <xsd:complexType name="OrchestrationProcess">
1679 <xsd:complexContent>
1680 <xsd:extension base="tns:Process" />
1681 </xsd:complexContent>
1682 </xsd:complexType>
1683
1684 <xsd:complexType name="Task">
1685 <xsd:complexContent>
1686 <xsd:extension base="tns:Element" />
1687 </xsd:complexContent>
1688 </xsd:complexType>
1689
1690 <xsd:complexType name="ServiceComposition">
1691 <xsd:complexContent>
1692 <xsd:extension base="tns:Composition" />
1693 </xsd:complexContent>
1694 </xsd:complexType>
1695
1696 <!-- Relationship referencing the artifact's UUID, to reference an InformationType.
1697 -->
1698 <xsd:complexType name="informationTypeTarget">
1699 <xsd:complexContent>
1700 <xsd:extension base="s-ramp:target">

```



```

1701 <xsd:attribute name="artifactType" type="s-ramp:informationTypeEnum"
1702 use="required" />
1703 </xsd:extension>
1704 </xsd:complexContent>
1705 </xsd:complexType>
1706 <!-- Relationship referencing the artifact's UUID, to reference a Service. -->
1707 <xsd:complexType name="serviceTarget">
1708 <xsd:complexContent>
1709 <xsd:extension base="s-ramp:target">
1710 <xsd:attribute name="artifactType" type="s-ramp:serviceEnum" use="required"
1711 />
1712 </xsd:extension>
1713 </xsd:complexContent>
1714 </xsd:complexType>
1715 <!-- Relationship referencing the artifact's UUID, to reference a ServiceContract.
1716 -->
1717 <xsd:complexType name="serviceContractTarget">
1718 <xsd:complexContent>
1719 <xsd:extension base="s-ramp:target">
1720 <xsd:attribute name="artifactType" type="s-ramp:serviceContractEnum"
1721 use="required" />
1722 </xsd:extension>
1723 </xsd:complexContent>
1724 </xsd:complexType>
1725 <!-- Relationship referencing the artifact's UUID, to reference a ServiceInterface.
1726 -->
1727 <xsd:complexType name="serviceInterfaceTarget">
1728 <xsd:complexContent>
1729 <xsd:extension base="s-ramp:target">
1730 <xsd:attribute name="artifactType" type="s-ramp:serviceInterfaceEnum"
1731 use="required" />
1732 </xsd:extension>
1733 </xsd:complexContent>
1734 </xsd:complexType>
1735 <!-- Relationship referencing the artifact's UUID, to reference a PolicySubject. --
1736 >
1737 <xsd:complexType name="policySubjectTarget">
1738 <xsd:complexContent>
1739 <xsd:extension base="s-ramp:target">
1740 <xsd:attribute name="artifactType" type="s-ramp:policySubjectEnum"
1741 use="required" />
1742 </xsd:extension>
1743 </xsd:complexContent>
1744 </xsd:complexType>
1745 <!-- Relationship referencing the artifact's UUID, to reference a Task. -->
1746 <xsd:complexType name="taskTarget">
1747 <xsd:complexContent>
1748 <xsd:extension base="s-ramp:target">
1749 <xsd:attribute name="artifactType" type="s-ramp:taskEnum" use="required" />
1750 </xsd:extension>
1751 </xsd:complexContent>
1752 </xsd:complexType>
1753 <!-- Relationship referencing the artifact's UUID, to reference a Policy. -->
1754 <xsd:complexType name="policyTarget">
1755 <xsd:complexContent>
1756 <xsd:extension base="s-ramp:target">

```



```

1757 <xsd:attribute name="artifactType" type="s-ramp:policyEnum" use="required" />
1758 </xsd:extension>
1759 </xsd:complexContent>
1760 </xsd:complexType>
1761 <!-- Relationship referencing the artifact's UUID, to reference an Element. -->
1762 <xsd:complexType name="elementTarget">
1763 <xsd:complexContent>
1764 <xsd:extension base="s-ramp:target">
1765 <xsd:attribute name="artifactType" type="s-ramp:elementEnum" use="required"
1766 />
1767 </xsd:extension>
1768 </xsd:complexContent>
1769 </xsd:complexType>
1770 <!-- Relationship referencing the artifact's UUID, to reference an Orchestration. -
1771 ->
1772 <xsd:complexType name="orchestrationTarget">
1773 <xsd:complexContent>
1774 <xsd:extension base="s-ramp:target">
1775 <xsd:attribute name="artifactType" type="s-ramp:orchestrationEnum"
1776 use="required" />
1777 </xsd:extension>
1778 </xsd:complexContent>
1779 </xsd:complexType>
1780 <!-- Relationship referencing the artifact's UUID, to reference an
1781 OrchestrationProcess. -->
1782 <xsd:complexType name="orchestrationProcessTarget">
1783 <xsd:complexContent>
1784 <xsd:extension base="s-ramp:target">
1785 <xsd:attribute name="artifactType" type="s-ramp:orchestrationProcessEnum"
1786 use="required" />
1787 </xsd:extension>
1788 </xsd:complexContent>
1789 </xsd:complexType>
1790 <!-- Relationship referencing the artifact's UUID, to reference an Event. -->
1791 <xsd:complexType name="eventTarget">
1792 <xsd:complexContent>
1793 <xsd:extension base="s-ramp:target">
1794 <xsd:attribute name="artifactType" type="s-ramp:eventEnum" use="required" />
1795 </xsd:extension>
1796 </xsd:complexContent>
1797 </xsd:complexType>
1798 <!-- Relationship referencing the artifact's UUID, to reference an Actor. -->
1799 <xsd:complexType name="actorTarget">
1800 <xsd:complexContent>
1801 <xsd:extension base="s-ramp:target">
1802 <xsd:attribute name="artifactType" type="s-ramp:actorEnum" use="required" />
1803 </xsd:extension>
1804 </xsd:complexContent>
1805 </xsd:complexType>
1806 <!-- Relationship referencing the artifact's UUID, to reference an Effect. -->
1807 <xsd:complexType name="effectTarget">
1808 <xsd:complexContent>
1809 <xsd:extension base="s-ramp:target">
1810 <xsd:attribute name="artifactType" type="s-ramp:effectEnum" use="required" />
1811 </xsd:extension>
1812 </xsd:complexContent>

```

```
1813 </xsd:complexType>
1814 </xsd:schema>
1815
1816
1817
1818
```

---

## 1819 Appendix G. Service Implementation Model Schema

1820 The S-RAMP Service Implementation Model Schema XSD file is also provided at:

1821 <http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0/>

```
1822
1823 <?xml version="1.0" encoding="UTF-8"?>
1824 <xsd:schema xmlns:tns="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0" xmlns:s-
1825 ramp="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
1826 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1827 targetNamespace="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
1828 elementFormDefault="qualified" version="1.0">
1829 <!--
1830 (c) 2010 Hewlett-Packard Company (HP), International Business Machines
1831 Corporation (IBM), Software AG (SAG) and TIBCO Software Inc. All
1832 rights reserved. Permission to copy and display the SOA Repository
1833 Artifact Model and Protocol (the "Specification"), in any medium
1834 without fee or royalty is hereby granted by Hewlett-Packard Company
1835 (HP), International Business Machines Corporation (IBM), Software AG
1836 (SAG) and TIBCO Software Inc. (collectively, the "Authors"), provided
1837 that you include the following on ALL copies of this document or
1838 portions thereof, that you make:
1839
1840 1. A link or URL to this document at this location:
1841 http://s-ramp.org/2010/s-ramp/specification/documents/{this document
1842 name}
1843 2. The copyright notice as shown in the Specification.
1844
1845 The Authors each agree to grant you a royalty-free license, under
1846 reasonable, non-discriminatory terms and conditions to their
1847 respective patents that they deem necessary to implement the "SOA
1848 Repository Artifact Model and Protocol" Specification, including all
1849 its constituent documents. THIS DOCUMENT IS PROVIDED "AS IS," AND THE
1850 AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED,
1851 INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS
1852 FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE
1853 CONTENTS OF THIS DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE
1854 IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY
1855 PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. THE AUTHORS WILL NOT
1856 BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR
1857 CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR
1858 DISTRIBUTION OF THIS DOCUMENT.
1859 -->
1860
1861 <xsd:include schemaLocation="coremodel.xsd" />
1862 <xsd:include schemaLocation="soamodel.xsd" />
1863
1864 <xsd:simpleType name="serviceImplementationModelEnum">
1865 <xsd:restriction base="s-ramp:baseArtifactEnum">
1866 <xsd:enumeration value="ServiceInstance" />
1867 <xsd:enumeration value="ServiceOperation" />
1868 <xsd:enumeration value="ServiceEndpoint" />
1869 <xsd:enumeration value="ServiceOperation" />
1870 </xsd:restriction>
```

```

1871 </xsd:simpleType>
1872
1873 <xsd:simpleType name="serviceInstanceEnum">
1874 <xsd:restriction base="s-ramp:serviceImplementationModelEnum">
1875 <xsd:enumeration value="ServiceInstance" />
1876 </xsd:restriction>
1877 </xsd:simpleType>
1878
1879 <xsd:simpleType name="serviceOperationEnum">
1880 <xsd:restriction base="s-ramp:serviceImplementationModelEnum">
1881 <xsd:enumeration value="ServiceOperation" />
1882 </xsd:restriction>
1883 </xsd:simpleType>
1884
1885 <!-- All Service Implementation Model artifacts inherit from
1886 ServiceImplementationModelType.
1887 Service Implementation Model Artifacts can have associated documents.
1888
1889 Service Implementation Model Artifacts can have dependencies upon other
1890 Service
1891 Implementation Model Artifacts. -->
1892
1893 <!-- Service Model Artifact Type -->
1894 <xsd:complexType name="ServiceImplementationModelType" abstract="true">
1895 <xsd:complexContent>
1896 <xsd:extension base="s-ramp:BaseArtifactType">
1897 <xsd:sequence>
1898 <!-- Modeled Relationship(s) to documentArtifactType: -->
1899 <xsd:element name="documentation" type="s-ramp:documentArtifactTarget"
1900 minOccurs="0" maxOccurs="unbounded" />
1901 </xsd:sequence>
1902 <xsd:anyAttribute namespace="##any" />
1903 </xsd:extension>
1904 </xsd:complexContent>
1905 </xsd:complexType>
1906
1907 <!-- Organization artifact Type. Used primarily in Service Model -->
1908 <xsd:complexType name="Organization">
1909 <xsd:complexContent>
1910 <xsd:extension base="s-ramp:Actor">
1911 <xsd:sequence>
1912 <!-- Modeled Relationship(s) to ServiceImplementationModelType: -->
1913 <xsd:element name="provides" type="s-ramp:serviceImplementationModelTarget"
1914 minOccurs="0" maxOccurs="unbounded" />
1915 <xsd:element name="extension" type="tns:ExtensionType" minOccurs="0" />
1916 <xsd:element name="end" type="xsd:string" minOccurs="1" maxOccurs="1" />
1917 <xsd:any namespace="##other" processContents="Lax" minOccurs="0"
1918 maxOccurs="unbounded" />
1919 </xsd:sequence>
1920 <xsd:anyAttribute namespace="##any" />
1921 </xsd:extension>
1922 </xsd:complexContent>
1923 </xsd:complexType>
1924
1925 <xsd:complexType name="ServiceInstance">
1926 <xsd:complexContent>

```

```

1927 <xsd:extension base="tns:ServiceImplementationModelType">
1928 <xsd:sequence>
1929 <!-- Modeled Relationship(s) to ServiceInstance(s): -->
1930 <xsd:element name="uses" type="s-ramp:serviceInstanceTarget" minOccurs="0"
1931 maxOccurs="unbounded" />
1932 <!-- Modeled Relationship(s) to ServiceEndpoint(s): -->
1933 <xsd:element name="describedBy" type="s-ramp:serviceInstanceTarget"
1934 minOccurs="0" maxOccurs="unbounded" />
1935 <xsd:element name="extension" type="tns:ExtensionType" minOccurs="0" />
1936 <xsd:element name="end" type="xsd:string" minOccurs="1" maxOccurs="1" />
1937 <xsd:any namespace="##other" processContents="lax" minOccurs="0"
1938 maxOccurs="unbounded" />
1939 </xsd:sequence>
1940 <xsd:anyAttribute namespace="##any" />
1941 </xsd:extension>
1942 </xsd:complexContent>
1943 </xsd:complexType>
1944
1945 <xsd:complexType name="ServiceOperation">
1946 <xsd:complexContent>
1947 <xsd:extension base="tns:ServiceImplementationModelType">
1948 <xsd:sequence>
1949 <!-- Modeled Relationship to the DerivedArtifactType that defines this
1950 ServiceOperation: -->
1951 <xsd:element name="operationDefinedBy" type="s-ramp:derivedArtifactTarget"
1952 minOccurs="0" maxOccurs="1" />
1953 <xsd:element name="extension" type="tns:ExtensionType" minOccurs="0" />
1954 <xsd:element name="end" type="xsd:string" minOccurs="1" maxOccurs="1" />
1955 <xsd:any namespace="##other" processContents="lax" minOccurs="0"
1956 maxOccurs="unbounded" />
1957 </xsd:sequence>
1958 <xsd:anyAttribute namespace="##any" />
1959 </xsd:extension>
1960 </xsd:complexContent>
1961 </xsd:complexType>
1962
1963 <xsd:complexType name="ServiceEndpoint">
1964 <xsd:complexContent>
1965 <xsd:extension base="tns:ServiceImplementationModelType">
1966 <xsd:sequence>
1967 <!-- Modeled Relationship with DerivedArtifactType which defines this
1968 ServiceEndpoint: -->
1969 <xsd:element name="endpointDefinedBy" type="s-ramp:derivedArtifactTarget"
1970 minOccurs="0" maxOccurs="1" />
1971 <xsd:element name="extension" type="tns:ExtensionType" minOccurs="0" />
1972 <xsd:element name="end" type="xsd:string" minOccurs="1" maxOccurs="1" />
1973 <xsd:any namespace="##other" processContents="lax" minOccurs="0"
1974 maxOccurs="unbounded" />
1975 </xsd:sequence>
1976 <xsd:attribute name="url" type="xsd:anyURI" />
1977 <xsd:anyAttribute namespace="##any" />
1978 </xsd:extension>
1979 </xsd:complexContent>
1980 </xsd:complexType>
1981
1982 <!-- Generic Extension Type for all un-used element extension points in tns -->

```

```

1983 <xsd:complexType name="ExtensionType">
1984 <xsd:sequence>
1985 <xsd:any processContents="Lax" minOccurs="1" maxOccurs="unbounded"
1986 namespace="##targetNamespace" />
1987 </xsd:sequence>
1988 <xsd:anyAttribute namespace="##any" />
1989 </xsd:complexType>
1990
1991 <!-- Relationship referencing the artifact's UUID, to reference a
1992 ServiceImplementationModel. -->
1993 <xsd:complexType name="serviceImplementationModelTarget">
1994 <xsd:complexContent>
1995 <xsd:extension base="s-ramp:target">
1996 <xsd:attribute name="artifactType" type="s-
1997 ramp:serviceImplementationModelEnum" use="required" />
1998 </xsd:extension>
1999 </xsd:complexContent>
2000 </xsd:complexType>
2001
2002 <!-- Relationship referencing the artifact's UUID, to reference a ServiceInstance.
2003 -->
2004 <xsd:complexType name="serviceInstanceTarget">
2005 <xsd:complexContent>
2006 <xsd:extension base="s-ramp:target">
2007 <xsd:attribute name="artifactType" type="s-ramp:serviceInstanceEnum"
2008 use="required" />
2009 </xsd:extension>
2010 </xsd:complexContent>
2011 </xsd:complexType>
2012
2013 <!-- Relationship referencing the artifact's UUID, to reference a ServiceOperation.
2014 -->
2015 <xsd:complexType name="serviceOperationTarget">
2016 <xsd:complexContent>
2017 <xsd:extension base="s-ramp:target">
2018 <xsd:attribute name="artifactType" type="s-ramp:serviceOperationEnum"
2019 use="required" />
2020 </xsd:extension>
2021 </xsd:complexContent>
2022 </xsd:complexType>
2023 </xsd:schema>
2024

```

2025

## Appendix H. XSD Model Schema

2026 The S-RAMP XSD Model Schema XSD file is also provided at:

2027 <http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0/>

2028

2029 `<?xml version="1.0" encoding="UTF-8"?>`

2030 `<xsd:schema xmlns:tns="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0" xmlns:s-`

2031 `ramp="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"`

2032 `xmlns:xsd="http://www.w3.org/2001/XMLSchema"`

2033 `targetNamespace="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"`

2034 `elementFormDefault="qualified" version="1.0">`

2035 `<!--`

2036 `(c) 2010 Hewlett-Packard Company (HP), International Business Machines`

2037 `Corporation (IBM), Software AG (SAG) and TIBCO Software Inc. All`

2038 `rights reserved. Permission to copy and display the SOA Repository`

2039 `Artifact Model and Protocol (the "Specification"), in any medium`

2040 `without fee or royalty is hereby granted by Hewlett-Packard Company`

2041 `(HP), International Business Machines Corporation (IBM), Software AG`

2042 `(SAG) and TIBCO Software Inc. (collectively, the "Authors"), provided`

2043 `that you include the following on ALL copies of this document or`

2044 `portions thereof, that you make:`

2045

2046 `1. A link or URL to this document at this location:`

2047 `http://s-ramp.org/2010/s-ramp/specification/documents/{this document`

2048 `name}`

2049 `2. The copyright notice as shown in the Specification.`

2050

2051 `The Authors each agree to grant you a royalty-free license, under`

2052 `reasonable, non-discriminatory terms and conditions to their`

2053 `respective patents that they deem necessary to implement the "SOA`

2054 `Repository Artifact Model and Protocol" Specification, including all`

2055 `its constituent documents. THIS DOCUMENT IS PROVIDED "AS IS," AND THE`

2056 `AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED,`

2057 `INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS`

2058 `FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE`

2059 `CONTENTS OF THIS DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE`

2060 `IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY`

2061 `PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. THE AUTHORS WILL NOT`

2062 `BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR`

2063 `CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR`

2064 `DISTRIBUTION OF THIS DOCUMENT.`

2065 `-->`

2066 `<xsd:include schemaLocation="coremodel.xsd" />`

2067

2068 `<xsd:simpleType name="xsdDocumentEnum">`

2069 `<xsd:restriction base="s-ramp:baseArtifactEnum">`

2070 `<xsd:enumeration value="XsdDocument" />`

2071 `</xsd:restriction>`

2072 `</xsd:simpleType>`

2073

2074 `<xsd:complexType name="XsdDocument">`

2075 `<xsd:complexContent>`

2076



```

2077 <xsd:extension base="s-ramp:XmlDocument">
2078 <xsd:sequence>
2079 <!-- Derived Relationships with (other)XmlDocument artifact(s): -->
2080 <xsd:element name="importedXsds" type="s-ramp:xsdDocumentTarget"
2081 minOccurs="0" maxOccurs="unbounded" />
2082 <xsd:element name="includedXsds" type="s-ramp:xsdDocumentTarget"
2083 minOccurs="0" maxOccurs="unbounded" />
2084 <xsd:element name="redefinedXsds" type="s-ramp:xsdDocumentTarget"
2085 minOccurs="0" maxOccurs="unbounded" />
2086 </xsd:sequence>
2087 <xsd:attribute name="targetNamespace" type="xsd:anyURI" use="optional" />
2088 </xsd:extension>
2089 </xsd:complexContent>
2090 </xsd:complexType>
2091 <xsd:complexType name="AttributeDeclaration">
2092 <xsd:complexContent>
2093 <xsd:extension base="s-ramp:DerivedArtifactType">
2094 <xsd:attribute name="NCName" type="xsd:NCName" />
2095 <xsd:attribute name="namespace" type="xsd:anyURI" />
2096 </xsd:extension>
2097 </xsd:complexContent>
2098 </xsd:complexType>
2099 <xsd:complexType name="ElementDeclaration">
2100 <xsd:complexContent>
2101 <xsd:extension base="s-ramp:DerivedArtifactType">
2102 <xsd:attribute name="NCName" type="xsd:NCName" />
2103 <xsd:attribute name="namespace" type="xsd:anyURI" />
2104 </xsd:extension>
2105 </xsd:complexContent>
2106 </xsd:complexType>
2107 <xsd:complexType name="XsdType" abstract="true">
2108 <xsd:complexContent>
2109 <xsd:extension base="s-ramp:DerivedArtifactType" />
2110 </xsd:complexContent>
2111 </xsd:complexType>
2112 <xsd:complexType name="ComplexTypeDeclaration">
2113 <xsd:complexContent>
2114 <xsd:extension base="tns:XsdType">
2115 <xsd:attribute name="NCName" type="xsd:NCName" />
2116 <xsd:attribute name="namespace" type="xsd:anyURI" />
2117 </xsd:extension>
2118 </xsd:complexContent>
2119 </xsd:complexType>
2120 <xsd:complexType name="SimpleTypeDeclaration">
2121 <xsd:complexContent>
2122 <xsd:extension base="tns:XsdType">
2123 <xsd:attribute name="NCName" type="xsd:NCName" />
2124 <xsd:attribute name="namespace" type="xsd:anyURI" />
2125 </xsd:extension>
2126 </xsd:complexContent>
2127 </xsd:complexType>
2128
2129 <!-- Relationship referencing the artifact's UUID, to reference an XsdDocument. -->
2130 <xsd:complexType name="xsdDocumentTarget">
2131 <xsd:complexContent>
2132 <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>
```

2139

## Appendix I. WSDL Model Schema

2140 The S-RAMP WSDL Model Schema XSD file is also provided at:

2141 <http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0/>

2142

2143 `<?xml version="1.0" encoding="UTF-8"?>`

2144 `<xsd:schema xmlns:tns="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0" xmlns:s-`

2145 `ramp="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"`

2146 `xmlns:xsd="http://www.w3.org/2001/XMLSchema"`

2147 `targetNamespace="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"`

2148 `elementFormDefault="qualified" version="1.0">`

2149 `<!--`

2150 `(c) 2010 Hewlett-Packard Company (HP), International Business Machines`

2151 `Corporation (IBM), Software AG (SAG) and TIBCO Software Inc. All`

2152 `rights reserved. Permission to copy and display the SOA Repository`

2153 `Artifact Model and Protocol (the "Specification"), in any medium`

2154 `without fee or royalty is hereby granted by Hewlett-Packard Company`

2155 `(HP), International Business Machines Corporation (IBM), Software AG`

2156 `(SAG) and TIBCO Software Inc. (collectively, the "Authors"), provided`

2157 `that you include the following on ALL copies of this document or`

2158 `portions thereof, that you make:`

2159

2160 `1. A link or URL to this document at this location:`

2161 `http://s-ramp.org/2010/s-ramp/specification/documents/{this document`

2162 `name}`

2163 `2. The copyright notice as shown in the Specification.`

2164

2165 `The Authors each agree to grant you a royalty-free license, under`

2166 `reasonable, non-discriminatory terms and conditions to their`

2167 `respective patents that they deem necessary to implement the "SOA`

2168 `Repository Artifact Model and Protocol" Specification, including all`

2169 `its constituent documents. THIS DOCUMENT IS PROVIDED "AS IS," AND THE`

2170 `AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED,`

2171 `INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS`

2172 `FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE`

2173 `CONTENTS OF THIS DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE`

2174 `IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY`

2175 `PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. THE AUTHORS WILL NOT`

2176 `BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR`

2177 `CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR`

2178 `DISTRIBUTION OF THIS DOCUMENT.`

2179 `-->`

2180 `<xsd:include schemaLocation="coremodel.xsd" />`

2181 `<xsd:include schemaLocation="xsdmodel.xsd" />`

2182 `<xsd:include schemaLocation="policymodel.xsd" />`

2183

2184 `<xsd:simpleType name="wsdlDocumentEnum">`

2185 `<xsd:restriction base="s-ramp:documentArtifactEnum">`

2186 `<xsd:enumeration value="wsdlDocument" />`

2187 `</xsd:restriction>`

2188 `</xsd:simpleType>`

2189

```

2190 <xsd:simpleType name="wsdlExtensionEnum">
2191 <xsd:restriction base="s-ramp:derivedArtifactEnum">
2192 <xsd:enumeration value="WsdlExtension" />
2193 </xsd:restriction>
2194 </xsd:simpleType>
2195 <xsd:simpleType name="portEnum">
2196 <xsd:restriction base="s-ramp:derivedArtifactEnum">
2197 <xsd:enumeration value="Port" />
2198 </xsd:restriction>
2199 </xsd:simpleType>
2200 <xsd:simpleType name="bindingEnum">
2201 <xsd:restriction base="s-ramp:derivedArtifactEnum">
2202 <xsd:enumeration value="Binding" />
2203 </xsd:restriction>
2204 </xsd:simpleType>
2205 <xsd:simpleType name="bindingOperationEnum">
2206 <xsd:restriction base="s-ramp:derivedArtifactEnum">
2207 <xsd:enumeration value="BindingOperation" />
2208 </xsd:restriction>
2209 </xsd:simpleType>
2210 <xsd:simpleType name="portTypeEnum">
2211 <xsd:restriction base="s-ramp:derivedArtifactEnum">
2212 <xsd:enumeration value="PortType" />
2213 </xsd:restriction>
2214 </xsd:simpleType>
2215 <xsd:simpleType name="operationEnum">
2216 <xsd:restriction base="s-ramp:derivedArtifactEnum">
2217 <xsd:enumeration value="Operation" />
2218 </xsd:restriction>
2219 </xsd:simpleType>
2220 <xsd:simpleType name="bindingOperationFaultEnum">
2221 <xsd:restriction base="s-ramp:derivedArtifactEnum">
2222 <xsd:enumeration value="BindingOperationFault" />
2223 </xsd:restriction>
2224 </xsd:simpleType>
2225 <xsd:simpleType name="bindingOperationInputEnum">
2226 <xsd:restriction base="s-ramp:derivedArtifactEnum">
2227 <xsd:enumeration value="BindingOperationInput" />
2228 </xsd:restriction>
2229 </xsd:simpleType>
2230 <xsd:simpleType name="bindingOperationOutputEnum">
2231 <xsd:restriction base="s-ramp:derivedArtifactEnum">
2232 <xsd:enumeration value="BindingOperationOutput" />
2233 </xsd:restriction>
2234 </xsd:simpleType>
2235 <xsd:simpleType name="operationInputEnum">
2236 <xsd:restriction base="s-ramp:derivedArtifactEnum">
2237 <xsd:enumeration value="OperationInput" />
2238 </xsd:restriction>
2239 </xsd:simpleType>
2240 <xsd:simpleType name="operationOutputEnum">
2241 <xsd:restriction base="s-ramp:derivedArtifactEnum">
2242 <xsd:enumeration value="OperationOutput" />
2243 </xsd:restriction>
2244 </xsd:simpleType>
2245 <xsd:simpleType name="faultEnum">

```

```

2246 <xsd:restriction base="s-ramp:derivedArtifactEnum">
2247 <xsd:enumeration value="Fault" />
2248 </xsd:restriction>
2249 </xsd:simpleType>
2250 <xsd:simpleType name="messageEnum">
2251 <xsd:restriction base="s-ramp:derivedArtifactEnum">
2252 <xsd:enumeration value="Message" />
2253 </xsd:restriction>
2254 </xsd:simpleType>
2255 <xsd:simpleType name="partEnum">
2256 <xsd:restriction base="s-ramp:derivedArtifactEnum">
2257 <xsd:enumeration value="Part" />
2258 </xsd:restriction>
2259 </xsd:simpleType>
2260 <xsd:simpleType name="xsdTypeEnum">
2261 <xsd:restriction base="s-ramp:derivedArtifactEnum">
2262 <xsd:enumeration value="XsdType" />
2263 </xsd:restriction>
2264 </xsd:simpleType>
2265
2266 <xsd:complexType name="WsdLDocument">
2267 <xsd:complexContent>
2268 <xsd:extension base="s-ramp:XMLDocument">
2269 <xsd:sequence>
2270 <!-- Derived Relationships: -->
2271 <xsd:element name="importedXsds" type="s-ramp:xsdDocumentTarget"
2272 minOccurs="0" maxOccurs="unbounded" />
2273 <xsd:element name="includedXsds" type="s-ramp:xsdDocumentTarget"
2274 minOccurs="0" maxOccurs="unbounded" />
2275 <xsd:element name="redefinedXsds" type="s-ramp:xsdDocumentTarget"
2276 minOccurs="0" maxOccurs="unbounded" />
2277 <xsd:element name="importedWsdls" type="s-ramp:wsdlDocumentTarget"
2278 minOccurs="0" maxOccurs="unbounded" />
2279 </xsd:sequence>
2280 <xsd:attribute name="targetNamespace" type="xsd:anyURI" use="optional" />
2281 <xsd:attribute name="xsdTargetNamespaces" type="xsd:anyURI" use="optional" />
2282 </xsd:extension>
2283 </xsd:complexContent>
2284 </xsd:complexType>
2285 <xsd:complexType name="WsdLDerivedArtifactType" abstract="true">
2286 <xsd:complexContent>
2287 <xsd:extension base="s-ramp:DerivedArtifactType">
2288 <xsd:sequence>
2289 <!-- Modeled "extension" relationship to any wsdlExtension artifact(s) -->
2290 <xsd:element name="extension" type="s-ramp:wsdlExtensionTarget"
2291 minOccurs="0" maxOccurs="unbounded" />
2292 </xsd:sequence>
2293 <xsd:attribute name="namespace" type="xsd:anyURI" />
2294 </xsd:extension>
2295 </xsd:complexContent>
2296 </xsd:complexType>
2297 <xsd:complexType name="WsdLService">
2298 <xsd:complexContent>
2299 <xsd:extension base="tns:NamedWsdLDerivedArtifactType">
2300 <xsd:sequence>
2301 <!-- 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" />

```

```

2358 <!-- Derived Relationship with Operation artifact: -->
2359 <xsd:element name="operation" type="s-ramp:operationTarget" />
2360 </xsd:sequence>
2361 </xsd:extension>
2362 </xsd:complexContent>
2363 </xsd:complexType>
2364 <xsd:complexType name="BindingOperationInput">
2365 <xsd:complexContent>
2366 <xsd:extension base="tns:NamedWsdLDerivedArtifactType">
2367 </xsd:extension>
2368 </xsd:complexContent>
2369 </xsd:complexType>
2370 <xsd:complexType name="BindingOperationFault">
2371 <xsd:complexContent>
2372 <xsd:extension base="tns:NamedWsdLDerivedArtifactType" />
2373 </xsd:complexContent>
2374 </xsd:complexType>
2375 <xsd:complexType name="Operation">
2376 <xsd:complexContent>
2377 <xsd:extension base="tns:NamedWsdLDerivedArtifactType">
2378 <xsd:sequence>
2379 <!-- Derived Relationships to OperationInput and OperationOutput for this
2380 Operation: -->
2381 <xsd:element name="input" type="s-ramp:operationInputTarget" />
2382 <xsd:element name="output" type="s-ramp:operationOutputTarget" />
2383 <!-- Derived Relationship for fault(s) associated with this Operation: -->
2384 <xsd:element name="fault" type="s-ramp:faultTarget" minOccurs="0"
2385 maxOccurs="unbounded" />
2386 </xsd:sequence>
2387 </xsd:extension>
2388 </xsd:complexContent>
2389 <!-- Only request / response is modeled -->
2390 </xsd:complexType>
2391 <xsd:complexType name="OperationInput">
2392 <xsd:complexContent>
2393 <xsd:extension base="tns:NamedWsdLDerivedArtifactType">
2394 <xsd:sequence>
2395 <!-- Derived Relationship with Message artifact: -->
2396 <xsd:element name="message" type="s-ramp:messageTarget" />
2397 </xsd:sequence>
2398 </xsd:extension>
2399 </xsd:complexContent>
2400 </xsd:complexType>
2401 <xsd:complexType name="Fault">
2402 <xsd:complexContent>
2403 <xsd:extension base="tns:NamedWsdLDerivedArtifactType">
2404 <xsd:sequence>
2405 <!-- Derived Relationship with Message artifact: -->
2406 <xsd:element name="message" type="s-ramp:messageTarget" />
2407 </xsd:sequence>
2408 </xsd:extension>
2409 </xsd:complexContent>
2410 </xsd:complexType>
2411 <xsd:complexType name="Message">
2412 <xsd:complexContent>
2413 <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">

```

```

2470 <xsd:complexContent>
2471 <xsd:extension base="s-ramp:target">
2472 <xsd:attribute name="artifactType" type="s-ramp:wSDLDocumentEnum"
2473 use="required" />
2474 </xsd:extension>
2475 </xsd:complexContent>
2476 </xsd:complexType>
2477 <!-- Relationship referencing the artifact's UUID, to reference a WSDLExtension. --
2478 >
2479 <xsd:complexType name="wSDLExtensionTarget">
2480 <xsd:complexContent>
2481 <xsd:extension base="s-ramp:target">
2482 <xsd:attribute name="artifactType" type="s-ramp:wSDLExtensionEnum"
2483 use="required" />
2484 </xsd:extension>
2485 </xsd:complexContent>
2486 </xsd:complexType>
2487 <!-- Relationship referencing the artifact's UUID, to reference a Port. -->
2488 <xsd:complexType name="portTarget">
2489 <xsd:complexContent>
2490 <xsd:extension base="s-ramp:target">
2491 <xsd:attribute name="artifactType" type="s-ramp:portEnum" use="required" />
2492 </xsd:extension>
2493 </xsd:complexContent>
2494 </xsd:complexType>
2495 <!-- Relationship referencing the artifact's UUID, to reference a Binding. -->
2496 <xsd:complexType name="bindingTarget">
2497 <xsd:complexContent>
2498 <xsd:extension base="s-ramp:target">
2499 <xsd:attribute name="artifactType" type="s-ramp:bindingEnum" use="required"
2500 />
2501 </xsd:extension>
2502 </xsd:complexContent>
2503 </xsd:complexType>
2504 <!-- Relationship referencing the artifact's UUID, to reference a BindingOperation.
2505 -->
2506 <xsd:complexType name="bindingOperationTarget">
2507 <xsd:complexContent>
2508 <xsd:extension base="s-ramp:target">
2509 <xsd:attribute name="artifactType" type="s-ramp:bindingOperationEnum"
2510 use="required" />
2511 </xsd:extension>
2512 </xsd:complexContent>
2513 </xsd:complexType>
2514 <!-- Relationship referencing the artifact's UUID, to reference a PortType. -->
2515 <xsd:complexType name="portTypeTarget">
2516 <xsd:complexContent>
2517 <xsd:extension base="s-ramp:target">
2518 <xsd:attribute name="artifactType" type="s-ramp:portTypeEnum" use="required"
2519 />
2520 </xsd:extension>
2521 </xsd:complexContent>
2522 </xsd:complexType>
2523 <!-- Relationship referencing the artifact's UUID, to reference an Operation. -->
2524 <xsd:complexType name="operationTarget">
2525 <xsd:complexContent>

```



```

2526 <xsd:extension base="s-ramp:target">
2527 <xsd:attribute name="artifactType" type="s-ramp:operationEnum" use="required"
2528 />
2529 </xsd:extension>
2530 </xsd:complexContent>
2531 </xsd:complexType>
2532 <!-- Relationship referencing the artifact's UUID, to reference a
2533 BindingOperationFault. -->
2534 <xsd:complexType name="bindingOperationFaultTarget">
2535 <xsd:complexContent>
2536 <xsd:extension base="s-ramp:target">
2537 <xsd:attribute name="artifactType" type="s-ramp:bindingOperationFaultEnum"
2538 use="required" />
2539 </xsd:extension>
2540 </xsd:complexContent>
2541 </xsd:complexType>
2542 <!-- Relationship referencing the artifact's UUID, to reference a
2543 BindingOperationInputTarget. -->
2544 <xsd:complexType name="bindingOperationInputTarget">
2545 <xsd:complexContent>
2546 <xsd:extension base="s-ramp:target">
2547 <xsd:attribute name="artifactType" type="s-ramp:bindingOperationInputEnum"
2548 use="required" />
2549 </xsd:extension>
2550 </xsd:complexContent>
2551 </xsd:complexType>
2552 <!-- Relationship referencing the artifact's UUID, to reference a
2553 BindingOperationOutput. -->
2554 <xsd:complexType name="bindingOperationOutputTarget">
2555 <xsd:complexContent>
2556 <xsd:extension base="s-ramp:target">
2557 <xsd:attribute name="artifactType" type="s-ramp:bindingOperationOutputEnum"
2558 use="required" />
2559 </xsd:extension>
2560 </xsd:complexContent>
2561 </xsd:complexType>
2562 <!-- Relationship referencing the artifact's UUID, to reference an OperationInput.
2563 -->
2564 <xsd:complexType name="operationInputTarget">
2565 <xsd:complexContent>
2566 <xsd:extension base="s-ramp:target">
2567 <xsd:attribute name="artifactType" type="s-ramp:operationInputEnum"
2568 use="required" />
2569 </xsd:extension>
2570 </xsd:complexContent>
2571 </xsd:complexType>
2572 <!-- Relationship referencing the artifact's UUID, to reference an OperationOutput.
2573 -->
2574 <xsd:complexType name="operationOutputTarget">
2575 <xsd:complexContent>
2576 <xsd:extension base="s-ramp:target">
2577 <xsd:attribute name="artifactType" type="s-ramp:operationOutputEnum"
2578 use="required" />
2579 </xsd:extension>
2580 </xsd:complexContent>
2581 </xsd:complexType>

```

```

2582 <!-- Relationship referencing the artifact's UUID, to reference a Fault. -->
2583 <xsd:complexType name="faultTarget">
2584 <xsd:complexContent>
2585 <xsd:extension base="s-ramp:target">
2586 <xsd:attribute name="artifactType" type="s-ramp:faultEnum" use="required" />
2587 </xsd:extension>
2588 </xsd:complexContent>
2589 </xsd:complexType>
2590 <!-- Relationship referencing the artifact's UUID, to reference a Message. -->
2591 <xsd:complexType name="messageTarget">
2592 <xsd:complexContent>
2593 <xsd:extension base="s-ramp:target">
2594 <xsd:attribute name="artifactType" type="s-ramp:messageEnum" use="required"
2595 />
2596 </xsd:extension>
2597 </xsd:complexContent>
2598 </xsd:complexType>
2599 <!-- Relationship referencing the artifact's UUID, to reference a Part. -->
2600 <xsd:complexType name="partTarget">
2601 <xsd:complexContent>
2602 <xsd:extension base="s-ramp:target">
2603 <xsd:attribute name="artifactType" type="s-ramp:partEnum" use="required" />
2604 </xsd:extension>
2605 </xsd:complexContent>
2606 </xsd:complexType>
2607 <!-- Relationship referencing the artifact's UUID, to reference any XsdType. -->
2608 <xsd:complexType name="xsdTypeTarget">
2609 <xsd:complexContent>
2610 <xsd:extension base="s-ramp:target">
2611 <xsd:attribute name="artifactType" type="s-ramp:xsdTypeEnum" use="required"
2612 />
2613 </xsd:extension>
2614 </xsd:complexContent>
2615 </xsd:complexType>
2616 </xsd:schema>

```

---

## 2617 Appendix J. SOAP WSDL Model Schema

2618 The S-RAMP SOAP WSDL Model Schema XSD file is also provided at:

2619 <http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0/>

2620

```
2621 <?xml version="1.0" encoding="UTF-8"?>
2622 <xsd:schema xmlns:tns="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0" xmlns:s-
2623 ramp="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
2624 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
2625 targetNamespace="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
2626 elementFormDefault="qualified" version="1.0">
2627 <!--
```

```
2628 (c) 2010 Hewlett-Packard Company (HP), International Business Machines
2629 Corporation (IBM), Software AG (SAG) and TIBCO Software Inc. All
2630 rights reserved. Permission to copy and display the SOA Repository
2631 Artifact Model and Protocol (the "Specification"), in any medium
2632 without fee or royalty is hereby granted by Hewlett-Packard Company
2633 (HP), International Business Machines Corporation (IBM), Software AG
2634 (SAG) and TIBCO Software Inc. (collectively, the "Authors"), provided
2635 that you include the following on ALL copies of this document or
2636 portions thereof, that you make:
```

2637

```
2638 1. A link or URL to this document at this location:
```

```
2639 http://s-ramp.org/2010/s-ramp/specification/documents/{this document
2640 name}
```

```
2641 2. The copyright notice as shown in the Specification.
```

2642

```
2643 The Authors each agree to grant you a royalty-free license, under
2644 reasonable, non-discriminatory terms and conditions to their
2645 respective patents that they deem necessary to implement the "SOA
2646 Repository Artifact Model and Protocol" Specification, including all
2647 its constituent documents. THIS DOCUMENT IS PROVIDED "AS IS," AND THE
2648 AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED,
2649 INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS
2650 FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE
2651 CONTENTS OF THIS DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE
2652 IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY
2653 PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. THE AUTHORS WILL NOT
2654 BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR
2655 CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR
2656 DISTRIBUTION OF THIS DOCUMENT.
```

```
2657 -->
```

2658

```
2659 <xsd:include schemaLocation="coremodel.xsd" />
```

```
2660 <xsd:include schemaLocation="wsdlmodel.xsd" />
```

2661

```
2662 <xsd:complexType name="SoapAddress">
```

```
2663 <xsd:complexContent>
```

```
2664 <xsd:extension base="s-ramp:WsdLExtension">
```

```
2665 <xsd:attribute name="soapLocation" type="xsd:anyURI" />
```

```
2666 </xsd:extension>
```

```
2667 </xsd:complexContent>
```

```
2668 </xsd:complexType>
2669
2670 <xsd:complexType name="SoapBinding">
2671 <xsd:complexContent>
2672 <xsd:extension base="s-ramp:WsdExtension">
2673 <xsd:attribute name="style" type="xsd:string" />
2674 <xsd:attribute name="transport" type="xsd:string" />
2675 </xsd:extension>
2676 </xsd:complexContent>
2677 </xsd:complexType>
2678 </xsd:schema>
```

2679

## Appendix K. Policy Model Schema

2680 The S-RAMP Policy Model Schema XSD file is also provided at:

2681 <http://s-ramp.org/2010/specification/schemas/>

2682

2683 `<?xml version="1.0" encoding="UTF-8"?>`

2684 `<xsd:schema xmlns:tns="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0" xmlns:s-`

2685 `ramp="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"`

2686 `xmlns:xsd="http://www.w3.org/2001/XMLSchema"`

2687 `targetNamespace="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"`

2688 `elementFormDefault="qualified" version="1.0">`

2689 `<!--`

2690 `(c) 2010 Hewlett-Packard Company (HP), International Business Machines`

2691 `Corporation (IBM), Software AG (SAG) and TIBCO Software Inc. All`

2692 `rights reserved. Permission to copy and display the SOA Repository`

2693 `Artifact Model and Protocol (the "Specification"), in any medium`

2694 `without fee or royalty is hereby granted by Hewlett-Packard Company`

2695 `(HP), International Business Machines Corporation (IBM), Software AG`

2696 `(SAG) and TIBCO Software Inc. (collectively, the "Authors"), provided`

2697 `that you include the following on ALL copies of this document or`

2698 `portions thereof, that you make:`

2699

2700 `1. A link or URL to this document at this location:`

2701 `http://s-ramp.org/2010/s-ramp/specification/documents/{this document`

2702 `name}`

2703 `2. The copyright notice as shown in the Specification.`

2704

2705 `The Authors each agree to grant you a royalty-free license, under`

2706 `reasonable, non-discriminatory terms and conditions to their`

2707 `respective patents that they deem necessary to implement the "SOA`

2708 `Repository Artifact Model and Protocol" Specification, including all`

2709 `its constituent documents. THIS DOCUMENT IS PROVIDED "AS IS," AND THE`

2710 `AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED,`

2711 `INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS`

2712 `FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE`

2713 `CONTENTS OF THIS DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE`

2714 `IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY`

2715 `PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. THE AUTHORS WILL NOT`

2716 `BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR`

2717 `CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR`

2718 `DISTRIBUTION OF THIS DOCUMENT.`

2719

2720 `-->`

2721 `<xsd:include schemaLocation="coremodel.xsd" />`

2722 `<xsd:include schemaLocation="wsdlmodel.xsd" />`

2723

2724 `<xsd:complexType name="SoapAddress">`

2725 `<xsd:complexContent>`

2726 `<xsd:extension base="s-ramp:WsdLExtension">`

2727 `<xsd:attribute name="soapLocation" type="xsd:anyURI" />`

2728 `</xsd:extension>`

2729 `</xsd:complexContent>`

```
2730 </xsd:complexType>
2731
2732 <xsd:complexType name="SoapBinding">
2733 <xsd:complexContent>
2734 <xsd:extension base="s-ramp:WsdExtension">
2735 <xsd:attribute name="style" type="xsd:string" />
2736 <xsd:attribute name="transport" type="xsd:string" />
2737 </xsd:extension>
2738 </xsd:complexContent>
2739 </xsd:complexType>
2740 </xsd:schema>
2741
```