SOA – Repository Artifact Model and Protocol Specification (S-RAMP)

Atom Binding

International Business Machines Corporation
Hewlett-Packard Corporation
Software AG
TIBCO Software Inc

Revision 1.0
October 7, 2010

Preface

Authors (alphabetically)

Tom Bellwood, IBM
Vince Brunssen, IBM
John Colgrave, IBM
Martin Dvorak, HP
Dan Enache, TIBCO
Steve Fanshier, SAG
Eric Johnson, TIBCO
Diane Jordan, IBM
Bernard Kufluk, IBM
Miroslav Novak, HP
Christopher Peltz, HP
Radek Pospisil, HP
Albert Regner, HP
Martin Smithson, IBM
Gary Woods, SAG
Prasad Yendluri, SAG

Copyright Notice

Abstract

The Atom Publishing Protocol (APP) lends it self well to the basic interaction scenarios associated with use of a SOA repository. This document provides an Atom Binding for S-RAMP, describing the use of APP to interact with a S-RAMP compliant repository. It defines how the S-RAMP Artifact Model is mapped to APP elements and the use of foreign markup, provides instance message examples and describes the message syntax for publishing content in a S-RAMP compliant repository, as well as performing query and subscription operations.

TABLE OF CONTENTS

PREFACE....II

Authors (alphabetically)....ii

Copyright Notice....ii

Abstract....iii

LIST OF EXAMPLES....VI

LIST OF TABLES....VIII

CHAPTER 1 - INTRODUCTION....1

1.1Purpose and Scope....1

1.2References....1

1.3Document History....1

1.4Conventions and Notation....1
1.4.1XML Namespaces....1
1.4.2Conformance....2

1.5Terminology....3

1.6Abbreviations and Acronyms....4

2.S-RAMP ARTIFACT TO ATOM ENTRY MAPPING....5

2.1Overview....5

2.2Design Principles....5

2.3Coarse Grained View....7
2.3.1S-RAMP Atom Category Schemes & Terms....8
2.3.2Atom Link Relation Values....10
2.3.3Service Document....11
2.3.4Atom Entries in S-RAMP....11
2.3.5Publishing Artifacts in the Coarse Grained View....14
2.3.5.1Publishing an Artifact Entry....14
2.3.5.2Publishing Multiple Artifact Entries....19
2.3.5.2.1Using Batch POST....20
2.3.5.2.2ZIP Publishing Using POST of ZIP File....23
2.3.5.3Retrieving Repository Artifacts....26

2.3.5.3.1Resolving Internal References....27
2.3.5.4Editing an Artifact Entry....27
2.3.5.5Deleting an Artifact Entry....28

2.4Fine Grained Views....28
2.4.1S-RAMP Relationships....28
2.4.1.1Relationship Feeds....29
2.4.1.2Relationship Entry Documents....33
2.4.1.3Relationship Type Entry Documents....37
2.4.1.4Creating a Relationship Instance....40
2.4.1.5Retrieving a Relationship Instance....44
2.4.1.6Editing a Relationship Instance....45
2.4.1.7Deleting a Relationship....45
2.4.2S-RAMP Properties....47
2.4.2.1Property Entry Documents....47
2.4.2.2Creating Properties....50
2.4.2.3Retrieving Properties....51
2.4.2.4Editing Properties....52
2.4.2.5Deleting Properties....53
2.4.3S-RAMP Classifications....53
2.4.3.1The Classification Entry Document....54
2.4.3.2Creating Classifications....55
2.4.3.3Retrieving Classifications....57
2.4.3.4Editing Classifications....58
2.4.3.5Deleting Classifications....58

3.S-RAMP QUERY USING ATOM BINDING....59

3.1Searching Repository Artifacts....59

3.2Inline Queries....59

3.3Stored Queries....61
3.3.1Stored Query Entry Documents....61

4.SECURITY....64

AS-RAMP ATOM SERVICE DOCUMENT....64

BS-RAMP URI SPACE....80

CS-RAMP ATOM BINDING SCHEMA....80

List of Examples

Example 1 - Summary Artifact Entry....11
Example 2 Full Artifact Entry with s-ramp:artifact Section....12
Example 3 - Publishing a Document Without Atom Multi-part POST....15
Example 4 - Initial Media Link Entry Returned Following a POST....16
Example 5 - Updating an Initial Media Link Entry with Metadata....17
Example 6 - Combined Publishing using Atom Multi-Part POST....18
Example 7: Batch Post Construct Example....20
Example 8 – Successful Batch POST Response....21
Example 9 – Failed Batch POST Response – Complete Rollback....22
Example 10 – Failed Batch POST Response – Partial Create....22
Example 11 - Response from Publish using ZIP File Method....24
Example 12 - Various Requests to Retrieve Repository Artifact(s) or Content....26
Example 13 - Complex Relationship Scenario Summary Entry....30
Example 14 - Complex Relationship Scenario Relationships Feed....33
Example 15 – Backward Relationships Feed....36
Example 16 - Relationship Types Feed....38
Example 17 - Creating Generic Relationships - Before....40
Example 18 - Creating Generic Relationships - Adding the Relationship....41
Example 19 - Creating Generic Relationships - After....42
Example 20 - Adding a Relationship with No Targets....43
Example 21 - Retrieving a Relationship Entry Instance....44
Example 22 - Property Entry Feed....48
Example 23 - Creating a Property - Adding the Property....50
Example 24 - Creating a Property - After....50
Example 25 - Retrieving a Property Entry Document....51
Example 26 - Editing a Property Entry Document....52
Example 27 - Classification Entry Feed....54
Example 28 - Creating a Classification - Before....55
Example 29 - Creating a Classification - Adding the Classification Entry....56
Example 30 - Creating a Classification - After....56
Example 31 - Retrieving a Classification Entry Document....57
Example 32 - Ad-hoc Queries....59

Example 33 - Ad-hoc Query Response..... 60

EXAMPLE 34 - STORED QUERY ENTRY DOCUMENT.....62

List of Tables

Table 1 - XML Namespace Prefixes Used..... 2
Table 2 - Defined Terms.....3
Table 3 – Mapping of built-in S-RAMP Artifact Properties to Atom Elements in an Entry Document.....7
Table 4 - Category term Attributes for Entry Types.....8
Table 5: Link rel Attribute Values.....10

TABLE 6 - S-RAMP URI SPACE.....80

Chapter 1 - Introduction

1.1 Purpose and Scope

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 fine-grained query. It includes binding(s) which document the syntax for interaction with a compliant Repository for create, read, update, delete, query and subscription operations within the context of each binding.

The specification is organized into several documents. This document, the *SOA Repository Artifact Model and Protocol - Atom Binding* document, builds upon the *SOA Repository Artifact Model and Protocol – Foundation* document. It describes the interaction protocol and syntax associated with using Atom to interact with an S-RAMP compliant Repository. Any other bindings will be expressed in their own separate binding documents.

When there is a discrepancy between this Atom Binding document and the Foundation document, the Atom Binding document takes precedence but only within the context of Atom based interaction within an S-RAMP compliant repository.

1.2 References

This specification shall be used in conjunction with the following publications. When the publications are superseded by an approved revision, the revision shall apply, unless otherwise noted here.

IETF, URN Namespace Definition

IETF, The Atom Publishing Protocol

IETF, The Atom Syndication Format

ISO 639.2, Codes for the Representation of Names and Languages

W3C, Extensible Markup Language (XML) 1.0 Specification

W3C, Namespaces in XML 1.0 (Second Edition)

W3C, XML Schema Part 1: Structures Second Edition, version 1.0

W3C, XML Path Language (XPath), version 2.0

A Universally Unique IDentifier (UUID) URN Namespace Specification, version 4

1.3 Document History

This version 0.9 is the first version of this document.

1.4 Conventions and Notation

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC 2119].

This specification uses the following syntax to define normative outlines for messages:

The syntax appears as an XML instance, but values in italics indicate data types instead of values.

The character "|" is used to indicate a choice between alternatives.

The characters "{" and "}" are used to indicate that the contained item is a description of a value instead of the actual value itself.

The characters "…" indicate where the syntax may be extended.

XML namespace prefixes (see Table 3) are used to indicate the namespace of the element being defined.

1.4.1 XML Namespaces

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

http://s-ramp.org/xmlns/2010/s-ramp

The namespaces used in this document are provided in Table 1 below. The choice of any namespace prefix is arbitrary and not semantically significant.

Table 1 - XML Namespace Prefixes Used

Prefix	XML Namespace	Specification(s)
atom	http://www.w3.org/2005/Atom	Atom Syndication Format
app	http://www.w3.org/2007/app	Atom Publishing Protocol
fn	http://www.w3.org/2005/xpath-functions	XPath 2
s-ramp	http://s-ramp.org/xmlns/2010/s-ramp	S-RAMP Foundation
wsdl	http://schemas.xmlsoap.org/wsdl/	WSDL [WSDL 1.1]
wsp	http://schemas.xmlsoap.org/ws/2002/12/policy	WS-Policy [WS-Policy]
xsd	http://www.w3.org/2001/XMLSchema	XML Schema [Part 1, 2]

1.4.2 Conformance

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

Normative text within this specification takes precedence over normative outlines, which in turn take precedence over the XML Schema [XML Schema Part 1, Part 2].

1.5 Terminology

Table 2 contains the defined terms used throughout this document:

Table 2 - Defined Terms

Term	Definition
Artifact Type	The data type of an S-RAMP artifact
Artifact Feed	An Atom feed of S-RAMP Artifacts.
Artifact Type Model	The set of all Artifact Types used in the S-RAMP specification
Backward Relationship Feed	A feed whose members are Relationship Entry document(s), whose Target Entry link matches the Artifact Entry in which the feed is included
Service Implementation Model	Data types in the S-RAMP Artifact Model which describe business concepts and relationships.
Coarse Grained View	An Atom entry document which represents an S-RAMP artifact. In its full version, it can be used to modify any and all metadata for the artifact.
Core Model	Base Artifact Types used in the S-RAMP specification.
Derived Artifact	Any S-RAMP artifact which is part of a Derived Model (e.g., XSD Model).
Fine Grained View	An optional representation of one of three classes of S-RAMP metadata (relationships, properties or classifications), which permit updates to individual metadata items without changing the Artifact Entry document.
Policy Model	Policy document related Derived Artifact Types used in the S-RAMP specification.
Relationship Entry	An Atom entry document which represents a relationship, which consists of the triplet of the Relationship Type, a link to its Source Entry and a link to its Target Entry.
Relationship Type	A name which represents the type of the relationship (e.g., “includedXsds”). Multiple relationships can share the same Relationship Type.
Relationship Type Entry	An Atom entry document instance which represents the type of the relationship (e.g., “includedXsds”). There is one such document instance for each Relationship Type.
Source Entry	An Atom entry document representing the source side artifact of a directed S-RAMP relationship.
Stored Query	A query stored in the repository which may be executed on demand.
Stored Query Entry	An Atom entry document representing a Stored Query.
Target Entry	An Atom entry document representing the target side artifact of a directed S-RAMP relationship.
WSDL Model	WSDL document related Derived Artifact Types used in the S-RAMP specification.
XSD Model	XSD document related Derived Artifact Types used in the S-RAMP specification.

1.6 Abbreviations and Acronyms

APP	Atom Publishing Protocol
S-RAMP	SOA Repository Artifact Model and Protocol
XPath2	XML Path Language (XPath) 2.0

2. S-RAMP Artifact to Atom Entry Mapping

This chapter describes how S-RAMP artifacts are represented in Atom as well as how to perform create, retrieve, update, delete operations against the data in an S-RAMP compliant repository.

2.1 Overview

This specification defines a single URI space for S-RAMP. This URI space is organized according to the Artifact Type Model structure. This URI space is used to reference repository objects in all Atom requests and responses. See Appendix B for details.

It is useful to discuss several other basic characteristics of the Atom mapping for S-RAMP.

- Atom entry documents are used to represent all S-RAMP artifacts in the repository.
- S-RAMP makes use of the APP media resource and media link entry concepts.
- In its full representation, all Atom entry documents (including Media Link Entries) used in S-RAMP contain the metadata associated with the S-RAMP repository artifact which it represents.
- S-RAMP artifact metadata will be represented in the same way in an Atom entry document, regardless of what kind of S-RAMP artifact it represents.
- For the convenience of Atom clients that do not understand S-RAMP data structures, applicable components of S-RAMP specific artifact metadata have been mapped to specific Atom elements for use in messages. Complete S-RAMP instance metadata, however, SHALL be provided in the full representation of an Artifact Entry document using foreign markup and SHALL be compliant with the S-RAMP schemas described in the Appendices of the *SOA Repository Artifact Model and Protocol Specification – Foundation Book.* While this representation is duplicative, it provides the greatest value for both S-RAMP aware as well as simple Atom clients. In all cases, however, the foreign markup content of a message SHALL take precedence and be considered authoritative by all S-RAMP compliant repositories.
- The HTTP POST message SHALL be used for creation of all new artifacts, and HTTP PUT SHALL be used for updating existing S-RAMP artifacts (see exception noted in Section 2.3.5.2.2 concerning the use of ZIP files to publish S-RAMP content).
- All S-RAMP artifacts are represented by the Atom Binding schema in Appendix C.
- Text in this document takes precedence over the schema.

2.2 Design Principles

Metadata in the S-RAMP data model comes in several forms:

- Built-in and user-defined properties which describe a repository artifact
- User-defined classifications which describe a repository artifact
- Built-in and user-defined relationships between two repository artifacts

S-RAMP has adhered to several design criteria requirements which have guided mapping of the binding independent S-RAMP Artifact Type Model onto an Atom binding:

- The Atom entry document is the Atom representation of an S-RAMP artifact object. There are two possible Atom representations of an S-RAMP artifact object.
    1. A summary Atom entry that appears in an Atom feed document. Summary entries do not include the S-RAMP structured extension element (S-RAMP foreign markup).
    2. A complete Atom entry which does include the S-RAMP structured extension element (see Section 2.3.2).

- It MUST be possible to update all user-editable S-RAMP metadata with a single update to an Atom entry document.
    - Some of the S-RAMP Artifact Type Model built-in properties have been mapped to existing Atom elements present in any legal Atom entry, wherever there is an obvious and direct analog. This provides some basic value to any Atom client.
    - The full Atom representation of an S-RAMP object will also include foreign markup in the form of a structured extension element (s-ramp:artifact). The XML markup within this element is compliant with the binding independent S-RAMP schemas, and provides a complete instance document describing the artifact represented by the Atom entry document. While, this does create some degree of duplication with those few items which are directly mapped to Atom elements, it does permit single step operations on repository artifacts and it facilitates various processing optimizations.
- Given the richness and complexity of S-RAMP metadata, it is useful for Atom clients to have greater contextual value and more convenient mechanisms for reading and editing that metadata beyond what is otherwise only available as foreign markup (described above). S-RAMP defines normative features whose implementation is OPTIONAL that provide clients with the ability to manipulate individual pieces of S-RAMP metadata.
- Whenever a feed is returned as the response to a client request to an S-RAMP server, the server MAY return a partial list that includes the appropriate links to support client pagination of the feed using the "first", "last", "previous" and "next" link *rel* types defined in the Atom Publishing Protocol specification, Section 10.1 "Collection partial lists".

These design principles have given rise to two major approaches to representing S-RAMP metadata in Atom:

1. Coarse Grained View (normative). This is an Atom entry document (Artifact Entry) which represents an S-RAMP object. It contains:
    - A subset of S-RAMP built-in property values mapped to existing Atom elements, as described in Table 3 of Section 2.3, in both summary and complete representations.
    - Only the complete Atom entry representation also contains a foreign markup section describing a structured extension element called s-ramp:artifact. This extension contains a complete S-RAMP XSD schema compliant instance document representing the S-RAMP artifact object.
    - The value of each target of a relationship in s-ramp:artifact for the Atom Binding SHALL be the URI of the Target Entry.
2. Fine Grained Views (optional features which have a normative interface when implemented):
    - These provide a hierarchical representation for a given class of metadata (relationships, properties or classifications)
    - Fine Grained Views allow an S-RAMP client to navigate to and manipulate the applicable metadata without the need to retrieve the full Atom representation of the S-RAMP object, which includes a potentially large structured extension element (see Section 2.3.2). Detailed information concerning the various Fine Grained Views can be found in Section 2.4.
    - The Artifact Entry document MAY contain Atom link(s) to feed(s) which contain Fine Grained View(s) for any of the three major classes of metadata described above.
        - The presence of these feed link(s) in each case indicates whether fine-grained support is exposed by a given implementation, for the indicated class(es) of metadata.
        - Using these fine-grained feeds, one can publish, retrieve, edit and delete individual pieces of such metadata without having to edit the Artifact Entry document which includes those feeds. For relationship metadata, it is also possible to perform a bulk deletion of all relationships of a given Relationship Type in one operation.
        - When Fine Grained View(s) are supported, an S-RAMP server SHALL present the appropriate feed link(s) in an Atom entry document. Attempts by a client to alter any of these feed links are ignored by the server.

Other miscellaneous conventions which pertain to Atom entry documents used in S-RAMP:

- Artifact Entry atom:id elements SHALL be a UUID, which is mapped from the S-RAMP artifact UUID value, expressed as a legal URN (e.g., urn:uuid:{uuid value}). *atom:id* element values for all other entry documents in S-RAMP have no normative format and are server generated. They SHALL be unique values, but do not need to be parsed or understood. Examples presented throughout this document, however, will either contain a real URN compliant UUID value, or will use a symbolic short name representing a UUID value for a the artifact, such as `{uuid:source.xsd}`, `{uuid:target.xsd}` and so on, to simplify or add clarity as needed.
- Also note that servers SHALL retain atom:id values chosen by clients for Artifact Entry documents during initial publish.
- *href* values used in links and src values used in atom:content elements carry no special meaning. As above, examples presented throughout this specification MAY depart from this rule simply to add clarity.
- URN values are used in S-RAMP atom:category *scheme* and atom:link *rel* attribute values. By convention, S-RAMP assumes a format for these URN values which contains ":" separated terms. Multiple ":" separators in a URN are permitted. At the time of publication, S-RAMP URN values use the "x-" convention established by RFC 2611.
- For clarity, the sample URLs used throughout this document have generally not been URL encoded.
- URLs used throughout this document often omit the base URL component which precedes "/s-ramp/…". For example:

  "http://host:port/mysoa/s-ramp/..."

  is presented as simply:

  "/s-ramp/…"

2.3 Coarse Grained View

Table 3 below defines a mapping for some S-RAMP artifact data items from the S-RAMP schema to built-in Atom elements in an Atom entry document. Only a small subset of such data is directly mapped. More is discussed on the remainder in subsequent sections.

Table 3 – Mapping of built-in S-RAMP Artifact Properties to Atom Elements in an Entry Document

S-RAMP Type Element	S-RAMP Subtype	Atom Element Mapping
BaseArtifactType	createdBy	Corresponds to the atom:name element of the atomPersonConstruct of the atom:author element. No mapping is defined for other elements in this construct.
	uuid	atom:id
	createdTimestamp	atom:published
	lastModifiedTimestamp	atom:updated
	lastModifiedBy	atom:contributor element lastModifiedBy maps to the atom:name element of the atomPersonConstruct of the atom:contributor element. No mapping is defined for other

		elements in this construct. ▪ *Note: S-RAMP restricts the number of occurrences of atom:contributor in an entry document to one.*
	artifactName	atom:title
	artifactDescription	atom:summary
DocumentArtifactType	contentType	In the Media Link Entry: atom:content, with this attribute: • type (for contentType)
XmlDocument	contentEncoding	atom:content, *type* attribute ▪ Including the *type* attribute using a *charset* attribute on the MIME media type. ▪ Example: type="application/xml; charset=utf-8"

2.3.1 S-RAMP Atom Category Schemes & Terms

S-RAMP pre-defines several atom:category *scheme* attribute and corresponding *term* values for describing the data present in an Atom entry document. These *term* values are fixed.

- urn:x-s-ramp:2010:type
  - Indicates the type of S-RAMP artifact represented by the Entry.
  - Appears in all entry documents and media link entry documents.

Defined values for the *term* attribute are described in below.

Table 4 - Category *term* Attributes for Entry Types

Type of Entry	**Defined *term* value(s)**
Artifact Entry (non-document)	“HumanActor” “AttributeDeclaration” “Binding” “BindingOperation” “BindingOperationInput” “BindingOperationOutput” “BindingOperationFault” “Choreography” “ChoreographyProcess” “Collaboration” “CollaborationProcess” “ComplexTypeDeclaration” “Composition”

	“Effect” “Element” “ElementDeclaration” “Event” “Fault” “InformationType” “Message” “Operation” “OperationInput” “OperationOutput” “Orchestration” “OrchestrationProcess” “Organization” “Part” “Policy” “PolicyAttachment” “PolicyExpression” “PolicySubject” “Port” “PortType” “Process” “SoapAddress” “SoapBinding” “SimpleTypeDeclaration” “Service” “ServiceComposition” “ServiceContract” “ServiceEndpoint” “ServiceInstance” “ServiceInterface” “ServiceOperation” “WsdlExtension” “WsdlService” “XsdType” “{User Defined Artifact Type}”
Artifact (Media Link) Entry (corresponds to a document)	“Document” “PolicyDocument” “WsdlDocument” “XmlDocument” “XsdDocument”
Relationship Entry	“relationship”
Relationship Type Entry	“relationshipType”
Property Entry	“property”

Classification Entry	“classification”
Stored Query Entry	“query”

- urn:s-ramp.org:2010:kind
  - Indicates the kind of the entry
  - Occurs in Artifact Entry, Relationship Target Entry, Relationship Type Entry, and Property Entry documents, except as noted below.
  - Legal values for the *term* attribute are
    - “derived”
      - Indicates entry is part of a Derived Model
    - “modeled”
      - Indicates entry is pre-defined and is part of the SOA or Service Implementation Models or a user defined model
    - “generic”
      - Indicates entry is ad-hoc
      - Does not occur in Artifact Entry documents.

2.3.2 Atom Link Relation Values

summarizes all of the relation attribute (*rel*) values for links used in S-RAMP:

Table 5: Link *rel* Attribute Values

Attribute Value	Where Used
self	All entry documents
edit-media	All media link entry documents
edit	All entry documents which can be edited by the client
urn:x-s-ramp:2010:relationships	Source Entry documents: Link to a feed of all Relationship entry documents
urn:x-s-ramp:2010:relationshipTypes	Source Entry and Relationship Type Entry documents: Link to a feed of all Relationship Type entry documents
urn:x-s-ramp:2010:relationships: {Relationship Type}	Source Entry documents: Link to a feed of Relationship entry documents which share the specified Relationship Type
urn:x-s-ramp:2010:backwardRelationships	Target Entry documents: Link to a feed of backward relationships of a Target Entry document (applies ONLY to modeled and derived relationships)
urn:x-s-ramp:2010:backwardRelationships: {Relationship Type}	Target Entry documents: Link to a feed of backward relationships for which the subject Target Entry document is the target of a relationship (of a given Relationship Type) in a Source Entry document. This feature ONLY applies to modeled and derived relationships. Each Backward Relationship Entry will only have a single target that points to the appropriate

	Source Entry Document, because a forward relationship from the Source Entry document never has duplicate targets for a given Relationship Type.
urn:x-s-ramp:2010:relationship:source	Relationship Entry Documents. Used in the Atom link to the Source Entry of the relationship.
urn:x-s-ramp:2010:relationship:target	Relationship Entry Documents. Used in the Atom link to the Target Entry of the relationship.
urn:x-s-ramp:2010:relationshipType	Relationship Entry Documents. Used in the Atom link to the Relationship Type Entry which corresponds to the Relationship Type value for this Relationship Entry.
urn:x-s-ramp:2010:properties	Artifact Entry documents: Link to a feed of all Property Entry documents
urn:x-s-ramp:2010:classifications	Artifact Entry documents: Link to a feed of all Classification Entry documents
urn:x-s-ramp:2010:query:results	Stored Query Entry Documents. Used in the results feed associated with execution of a given Stored Query.

2.3.3 Service Document

This section describes how S-RAMP implementations publicize the top level collections defined by this specification in an Atom Publishing Protocol Service Document.

S-RAMP implementations SHALL return an Atom Publishing Protocol Service Document to clients who perform an HTTP GET on the following URL:

```
{base URL}/s-ramp/servicedocument
```

The content of the Service Document that is returned is defined as follows:

- MUST contain a workspace for each of the artifact models identified in Section 3 of the *SOA Repository Artifact Model & Protocol Specification – Foundation Document*.
- Each workspace MUST contain an app:collection element for each of the artifact types that are defined within the corresponding artifact model for that workspace.
- Each collection in a workspace MUST specify an atom:categories element that will define the categories that MUST be applied to the member resources of the collection as defined in Section 2.3.1.
- The workspace for the query artifact model MUST contain an app:collection element for each Stored Query that exists in the S-RAMP implementation.

The workspace for the SOA or Service Implementation Artifact Model MUST contain an app:collection element for each user defined type that has been registered in the S-RAMP implementation.

2.3.4 Atom Entries in S-RAMP

An example of an S-RAMP summary (media link) entry which corresponds to the accountingTypes.xsd resource is shown below. The mapping defined in Table 3 is illustrated in Example 1:

Example 1 - Summary Artifact Entry

```
<entry xmlns=”http://www.w3.org/2005/Atom”
       xmlns:s-ramp=”http://s-ramp.org/xmlns/2010/s-ramp”>
    <id>urn:uuid:aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a</id>
```

```
    <updated>2009-05-26T13:13:55.013+02:00</updated>
    <title type="text">accountingTypes.xsd</title>
    <published>2009-05-26T13:13:55.013+02:00</published>
    <author>
        <name>Bellwood</name>
    </author>
    <contributor>
        <name>Pospisil</name>
    </contributor>
    <summary type="text">Accounting types schema document</summary>
    <content
        type="application/xml"
        src="http://example.org/s-ramp/xsd/XsdDocument/
        aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/media"/>
    <link type="application/atom+xml;type=entry" rel="self"
        href="http://example.org/s-ramp/xsd/XsdDocument/
        aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a" />
    <link type="application/atom+xml;type=entry" rel="edit-media"
        href="http://example.org/s-ramp/xsd/XsdDocument/
        aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/media" />
    <link type="application/atom+xml;type=entry" rel="edit"
        href="http://example.org/s-ramp/xsd/XsdDocument/
        aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a" />

    <!—Links to of the optional feeds provided by server implementations which
        support the various the Fine Grained Views defined in S-RAMP omitted for brevity.

        See Section 2.4 for complete information on these feeds and the Fine Grained
        View.-->

    <!— S-RAMP defined categorizations identifying class of data represented by
        this entry -->
     <category term="XsdDocument" label="XML Schema"
        scheme="urn:x-s-ramp:2010:type" />
</entry>
```

As noted earlier, a full Atom entry representation of an S-RAMP object includes foreign markup in the form of a structured extension element called s-ramp:artifact. This contains an S-RAMP schema compliant XML instance fragment describing the complete S-RAMP artifact. A sample s-ramp:artifact section is shown below in the full version of the Artifact Entry described in the example in Section 2.3. The optional Fine Grained View feed links are again omitted here for brevity:

Example 2 Full Artifact Entry with s-ramp:artifact Section

```
<entry xmlns="http://www.w3.org/2005/Atom"
    xmlns:s-ramp="http://s-ramp.org/xmlns/2010/s-ramp"
    xmlns:xlink="http://www.w3.org/1999/xlink" >
    <id>urn:uuid:aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a</id>
    <updated>2009-05-26T13:13:55.013+02:00</updated>
    <title type="text">accountingTypes.xsd</title>
    <published>2009-05-26T13:13:55.013+02:00</published>
    <author>
        <name>Bellwood</name>
    </author>
    <contributor>
```

```
    <name>Pospisil</name>
</contributor>
<summary type="text">accountingTypes.xsd schema document</summary>
<content
    type="application/xml;charset=utf-8"
    src="http://example.org/s-ramp/xsd/XsdDocument/
    aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/media"/>
<link type="application/atom+xml;type=entry" rel="self"
    href="http://example.org/s-ramp/xsd/XsdDocument/
    aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a”/>
<link type="application/atom+xml;type=entry" rel="edit-media"
    href="http://example.org/s-ramp/xsd/XsdDocument/
    aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/media" />
<link type="application/atom+xml;type=entry" rel="edit"
    href="http://example.org/s-ramp/xsd/XsdDocument/
    aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a” />

<!— S-RAMP defined categorizations identifying class of data represented by
    this entry -->
<category term="XsdDocument" label="XML Schema"
    scheme="urn:x-s-ramp:2010:type” />
<s-ramp:artifact
    xsi:schemaLocation="http://s-ramp.org/xmlns/2010/s-ramp
        http://s-ramp.org/2010/specification/schemas/xsdmodel.xsd"
    xmlns:s-ramp="http://s-ramp.org/xmlns/2010/s-ramp"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <s-ramp:XsdDocument
        name=”accountingTypes.xsd”
        description=”accountingTypes.xsd”
        createdBy="Bellwood" version="1.0"
        uuid="aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a"
        createdTimestamp="2009-05-26T13:13:55.013+02:00"
        lastModifiedTimestamp="2009-06-26T13:13:55.013+02:00"
        lastModifiedBy="Pospisil"  contentEncoding="UTF-8"
        contentType=“application/xml”  contentSize="4096" >
        <s-ramp:classifiedBy>
            http://example.org/ontologies/accounting.owl/accounts
        </s-ramp:classifiedBy>
        <!-- Example of the “importedXsds” Derived relationship -->
        <s-ramp:importedXsds>
            <s-ramp:target xlink:href=”http://example.org/s-ramp/xsd/XsdDocument/
                aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6b”>
               aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6b
           </s-ramp:target>
        </s-ramp:importedXsds>
        <!-- Example of a user created generic relationship called “similarXsds”
            between this xsd artifact and two others with UUID values of “...a6b”
            and “...a6c”), respectively: -->
        <s-ramp:relationship>
            <s-ramp:relationshipType>similarXsds</s-ramp:relationshipType>
            <s-ramp:relationshipTarget>
               <s-ramp:target xlink:href=”http://example.org/s-ramp/xsd/XsdDocument/
                   aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6b”>
                  aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6b
```

```
            </s-ramp:target>
          </s-ramp:relationshipTarget>
          <s-ramp:relationshipTarget>
             <s-ramp:target xlink:href="http://example.org/s-ramp/xsd/XsdDocument/
                 aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6c">
                aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6c
            </s-ramp:target>
          </s-ramp:relationshipTarget>
        </s-ramp:relationship>

        <s-ramp:property>
            <propertyName>myPropertyName</propertyName>
            <propertyValue>myPropertyValue</propertyValue>
        </s-ramp:property>
      </s-ramp:XsdDocument>
  </s-ramp:artifact>
</entry>
```

2.3.5 Publishing Artifacts in the Coarse Grained View

The Coarse Grained View in S-RAMP supports a complete representation of all of the metadata which describes an S-RAMP artifact, mapped to an Atom entry document. Some of S-RAMP built-in properties are mapped directly to existing Atom entry elements for the convenience of clients, but as illustrated in Section 2.3.2, the S-RAMP structured extension element (s-ramp:artifact) contains a complete S-RAMP schema compliant representation of the artifact, and MUST be used to publish in the Coarse Grained View.

Publishing in this view requires understanding the s-ramp:artifact structured extension, as it is the vehicle by which all metadata associated with the entry is specified.

Publishing an artifact to the wrong collection will result in HTTP error “403” Forbidden.

2.3.5.1 Publishing an Artifact Entry

Publishing of new artifacts to an S-RAMP compliant repository is accomplished using HTTP POST. All S-RAMP artifacts have an atom:entry representation, but only those which are not derived artifacts (such as document or SOA Model artifacts) can be published directly by the client. By convention, S-RAMP documents are treated as *media resources* by the Atom Binding. Publication of non-document artifacts is accomplished using HTTP POST of an Atom entry document which represents it. Publication of a document via HTTP POST will result in the creation of the document artifact in the repository AND a Media Link Entry document which corresponds to it and is returned in response to the POST. This entry will contain initial metadata associated with the document. The UUID in it is set by the server, and the document includes an edit-media link to the media resource as does the atom:content element via the *src* attribute, although the value of the IRI of the src attribute on the atom:content element does not need to be the same as the media resource IRI contained in the edit-media link. This allows implementations to point clients at a cached version of a Media Resource. The Media Link Entry can then be updated to modify or add additional metadata and be PUT to the repository as desired. Unfortunately APP currently makes this a multi-step process if one wishes to publish a document together with domain specific metadata associated with the document. Typical steps might entail:

1. POST the document. This saves the document in the repository, and returns the initial Media Link Entry.
2. Edit the Media Link Entry to insert & change metadata as desired
3. PUT the Media Link Entry to update its contents the repository

To simplify this process for clients, S-RAMP implementations SHALL also support publication of media resources and Media Link Entries in a single step using the procedure described in the *ATOM Multi-part Media Resource Creation* draft document (hereafter referred to as “*Atom Multi-part POST*”) under review by the IETF. It specifies extensions to APP which allow simultaneous publishing of a media resource and its corresponding

Media Link Entry in a single HTTP request. The body of the request thus contains both the media resource and a boundary delimited Media Link Entry containing all the desired metadata for the resource. The result of this operation will save the document in the repository, and create a complete Media Link Entry containing an edit-media link to that document with no additional steps.

Notes:

- A client MAY specify the name of an artifact using the Slug header in the HTTP POST. Clients SHOULD not assume that the Slug header is used to influence the atom:id or the URIs.
- If an *Atom Multi-part POST* is used, than the name and UUID values SHOULD only be provided as part of the s-ramp:metadata in the Media Link Entry portion of the POST.
- UUID values can be provided by the user for a non-document Artifact being published. In order to conform to the APP specification (sections 4.3 and 9.2), the following HTTP operations will have only these uses:
    - POST
        - Only used to create new Atom entry documents.
        - Any uuid property value supplied MUST NOT already be present in the repository.
        - If the supplied uuid property value duplicates one already in the repository, the server SHALL return an HTTP error code of "409" indicating a Conflict.
        - If a UUID is not supplied, the server SHALL create one.
    - PUT
        - Only used to update an existing Atom entry document.
        - The uuid property value CANNOT be changed.
        - If the artifact being edited is not in the repository at the time of the PUT, then the server SHALL return an HTTP error code of "404" indicating a Not Found.
- Within a full Coarse Grained Artifact Entry document, the s-ramp:artifact element might contain Modeled or Derived relationships, whose type is s-ramp:target. It might also contain user provided Generic relationships, whose s-ramp:relationshipTarget element is also of type s-ramp:target). The Core Model Schema in Appendix A of the Foundation Document defines the s-ramp:target type. The s-ramp:target element in the Core Model Schema terminates in the ##any attribute. The Atom Binding requires that server responses include an xlink:href attribute in the target element. The value of this attribute MUST contain a URL pointing to the target. During a publish operation, the link need not be included and the server MUST ignore it. See Example 2 above for more.
- Document artifacts MAY only be published to a valid collection described in the Atom Service Document (see Appendix A). For example, the "…/s-ramp/xsd/XsdDocument" collection.
- Documents which are part of a ZIP file MAY be published to the top level URI for the appropriate model in which they reside (e.g., /s-ramp/xsd). The repository will determine the appropriate collection(s) in which to place its constituent entries based on the category information in the entry document(s) included in the ZIP file. For more on publishing using ZIP files, refer to Section 2.3.5.2.2. If the ZIP file contains files which belong in different S-RAMP models, the ZIP file should be published to the top level collection (i.e., /s-ramp).

Example 3 below illustrates publishing of the accountingTypes.xsd document without using the Atom Multi-part POST method. Publication is done in two steps. First we POST the document itself:

Example 3 - Publishing a Document Without Atom Multi-part POST

```
POST /s-ramp/xsd/XsdDocument HTTP/1.1
Host: example.org
Content-Type: application/xml
```

```
Content-Length: nnn
Slug: accountingTypes.xsd

{accountingTypes.xsd document content goes here}
```

In response to this POST, the server will return an initial Media Link Entry based upon the information provided on the POST and within the XSD file. For a moment, we will assume that this XSD file does NOT include or import any other XSD files and thus includes no external resource dependencies. The atom:summary element value at this point is implementation defined, and there are no user defined properties or classifications yet. Clients can adjust all of these later:

Example 4 - Initial Media Link Entry Returned Following a POST

```
HTTP/1.1 201 Created
Date: Tues, 26 May 2009 13:13:55 GMT+2:00
Content-Length: nnn
Content-Type: application/atom+xml;type=entry;charset="utf-8"
Location: http://example.org/s-ramp/xsd/XsdDocument/
            aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a
ETag: "c181bb840673b5"

<?xml version="1.0"?>
 <entry xmlns="http://www.w3.org/2005/Atom"
        xmlns:s-ramp="http://s-ramp.org/xmlns/2010/s-ramp">
    <id>urn:uuid:aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a</id>
    <updated>2009-05-26T13:13:55.013+02:00</updated>
    <title type="text">accountingTypes.xsd</title>
    <published>2009-05-26T13:13:55.013+02:00</published>
    <author>
        <name>Bellwood</name>
    </author>
    <summary type="text">accountingTypes.xsd schema document</summary>
    <content
        type="application/xml;charset=utf-8"
        src="http://example.org/s-ramp/xsd/XsdDocument/
        aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/media"/>
    <link type="application/atom+xml;type=entry" rel="self"
        href="http://example.org/s-ramp/xsd/XsdDocument/
        aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a" />
    <link type="application/atom+xml;type=entry" rel="edit-media"
        href="http://example.org/s-ramp/xsd/XsdDocument/
        aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/media" />
    <link type="application/atom+xml;type=entry" rel="edit"
        href="http://example.org/s-ramp/xsd/XsdDocument/
        aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a" />
    <!– S-RAMP defined categorizations identifying class of data represented by
        this entry -->
     <category term="XsdDocument" label="XML Schema"
        scheme="urn:x-s-ramp:2010:type" />
    <s-ramp:artifact
        xsi:schemaLocation="http://s-ramp.org/xmlns/2010/s-ramp
            http://s-ramp.org/2010/specification/schemas/xsdmodel.xsd"
        xmlns:s-ramp="http://s-ramp.org/xmlns/2010/s-ramp"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```
        <s-ramp:XsdDocument
            name=”accountingTypes.xsd”
            description=”accountingTypes.xsd schema document”
            createdBy="Bellwood" version="1.0"
            uuid="aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a "
            createdTimestamp="2009-05-26T13:13:55.013+02:00"
            lastModifiedTimestamp="2009-05-26T13:13:55.013+02:00"
            lastModifiedBy="Bellwood"  contentEncoding="UTF-8"
            contentType=“application/xml”  contentSize="4096" >
        </s-ramp:XsdDocument>
    </s-ramp:artifact>
</entry>
```

Clients can update this Media Link Entry to add additional metadata. This is done using an HTTP PUT as in Example 5 below. All such metadata added using the Coarse Grained View is done through the s-ramp:artifact extension. This example illustrates adding a user-defined property and classification in the s-ramp:artifact section

Clients SHOULD perform a GET on the entry to insure it is complete and current. S-RAMP server implementations SHALL support and return an ETag to allow conditional GET and PUT.

Example 5 - Updating an Initial Media Link Entry with Metadata

```
PUT /s-ramp/xsd/XsdDocument/aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a HTTP/1.1
Host: example.org
Content-Type: application/atom+xml;type=entry
Content-Length: nnn
If-Match: "c181bb840673b5"

<?xml version="1.0"?>
<entry xmlns=”http://www.w3.org/2005/Atom”
       xmlns:s-ramp=”http://s-ramp.org/xmlns/2010/s-ramp”
>
    <id>urn:uuid:aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a</id>
    <updated>2009-05-26T13:13:55.014+02:00</updated>
    <title type=“text”>accountingTypes.xsd</title>
    <published>2009-05-26T13:13:55.013+02:00</published>
    <author>
        <name>Bellwood</name>
    </author>
    <contributor>
        <name>Pospisil</name>
    </contributor>
    <summary type="text">accountingTypes.xsd schema document</summary>
    <content
        type="application/xml;charset=utf-8"
        src="http://example.org/s-ramp/xsd/XsdDocument/
        aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/media"/>
    <link type="application/atom+xml;type=entry" rel="self"
        href="http://example.org/s-ramp/xsd/XsdDocument/
        aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a” />
    <link type="application/atom+xml;type=entry" rel="edit-media"
        href="http://example.org/s-ramp/xsd/XsdDocument/
        aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/media" />
    <link type="application/atom+xml;type=entry" rel="edit"
```

```
            href="http://example.org/s-ramp/xsd/XsdDocument/
            aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a" />

        <!– S-RAMP defined categorizations identifying class of data represented by
            this entry -->
        <category term="XsdDocument" label="XML Schema"
            scheme="urn:x-s-ramp:2010:type" />

        <s-ramp:artifact
            xsi:schemaLocation="http://s-ramp.org/xmlns/2010/s-ramp
                http://s-ramp.org/2010/specification/schemas/xsdmodel.xsd"
            xmlns:s-ramp="http://s-ramp.org/xmlns/2010/s-ramp"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
            <s-ramp:XsdDocument
                name="accountingTypes.xsd"
                description="accountingTypes.xsd schema document"
                createdBy="Bellwood" version="1.0"
                uuid="aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a "
                createdTimestamp="2009-05-26T13:13:55.013+02:00"
                lastModifiedTimestamp="2009-05-26T13:13:56.013+02:00"
                lastModifiedBy="Bellwood"  contentEncoding="UTF-8"
                contentType="application/xml"  contentSize="4096" >
                <s-ramp:classifiedBy>
                    http://example.org/ontologies/accounting.owl/accounts
                </s-ramp:classifiedBy>
                <s-ramp:property>
                    <propertyName>accountingCalendar</propertyName>
                    <propertyValue>2009</propertyValue>
                </s-ramp:property>
            </s-ramp:XsdDocument>
        </s-ramp:artifact>
    </entry>
```

The steps illustrated in Example 3 through Example 5 above could all be performed in a single HTTP request using the *Atom Multi-part POST*. This method allows combining both the Atom Media Link Entry and its corresponding media resource document in a single POST. Example 6 below presents the same example in this combined approach:

Example 6 - Combined Publishing using Atom Multi-Part POST

```
POST /s-ramp/xsd/XsdDocument HTTP/1.1
Host: example.org
Content-Type: multipart/related;boundary="===============1605871705==";
type="application/atom+xml"
MIME-Version: 1.0

--===============1605871705==
Content-Type: application/atom+xml; charset="utf-8"
MIME-Version: 1.0

<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom"
       xmlns:s-ramp="http://s-ramp.org/xmlns/2010/s-ramp">
    <id>urn:uuid:aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a</id>
```

```
    <updated>2009-05-26T13:13:55.014+02:00</updated>
    <title type="text”>accountingTypes.xsd</title>
    <author>
        <name>Bellwood</name>
    </author>
    <summary type="text">accountingTypes.xsd schema document</summary>

    <!— S-RAMP defined categorizations identifying class of data represented by
        this entry -->
     <category term="XsdDocument" label="XML Schema"
        scheme="urn:x-s-ramp:2010:type” />

    <s-ramp:artifact
        xsi:schemaLocation="http://s-ramp.org/xmlns/2010/s-ramp
            http://s-ramp.org/2010/specification/schemas/xsdmodel.xsd"
        xmlns:s-ramp="http://s-ramp.org/xmlns/2010/s-ramp"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <s-ramp:XsdDocument
            name=”accountingTypes.xsd”
            description=”accountingTypes.xsd schema document”
            createdBy="Bellwood" version="1.0"
            uuid="aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a "
            createdTimestamp="2009-05-26T13:13:55.013+02:00"
            lastModifiedTimestamp="2009-06-26T13:13:55.013+02:00"
            lastModifiedBy="Bellwood"  contentEncoding="UTF-8"
            contentType=“application/xml”  contentSize="4096" >
            <s-ramp:classifiedBy>
                http://example.org/ontologies/accounting.owl/accounts
            </s-ramp:classifiedBy>
            <s-ramp:property>
                <propertyName>accountingCalendar</propertyName>
                <propertyValue>2009</propertyValue>
            </s-ramp:property>
        </s-ramp:XsdDocument>
    </s-ramp:artifact>
</entry>
--===============1605871705==
Content-Type: application/xml
MIME-Version: 1.0

{XML content of accountTypes.xsd document goes here}
--===============1605871705==--
```

Non-document artifacts are published directly using HTTP POST of the desired Artifact Entry document to the appropriate S-RAMP collection and do not use the *Atom Multi-part POST* method

Note: Derived Artifacts cannot be published (created or deleted) directly, as these are automatically generated and managed by the server as part of the publication of the document to which the corresponding Derived Artifact Model and all its constituent artifacts apply.

2.3.5.2 Publishing Multiple Artifact Entries

Many document types include or import other documents upon which they are dependent. For example, XSD document A.xsd imports XSD document B.. Since S-RAMP repositories require that all dependencies be

resolvable at the time of publication, allowing only the publication of one artifact at a time would mean that the dependent document B.xsd would need to be published first, followed by publication of A.xsd, so that the server could resolve this dependency. But this is clumsy, inefficient and potentially difficult for clients to manage. To simplify publication of such documents for clients, this section discusses two methods for publishing multiple resources (documents) and their associated metadata which S-RAMP compliant repositories SHALL support. Each method can support documents with nested dependencies since all dependent documents are published in a single step. All dependencies MUST be resolvable at the time of publication.

2.3.5.2.1 Using Batch POST

RFC 2387, "The MIME Multipart/Related Content-type" describes how to perform a multipart POST of binary documents. S-RAMP builds on this RFC to extend the IETF draft document "ATOM Multi-part Media Resource Creation" to support the simultaneous publishing of a collection of non-dependent and dependent resources and their associated metadata. With this approach it is possible, for example, to publish an XSD document which imports two other XSD documents by including it as well as its dependencies in a single POST to the repository.

The Batch POST method requires an atom entry document as the root body part. This root atom entry document points to any dependent atom entry documents contained in other body parts using atom:link elements with an *href* attribute whose value is the Content-ID of the relevant body part. It also points at the actual document content by specifying the Content-ID of the relevant body part as the value of the *src* attribute in the atom:content element of the atom entry document. This approach provides complete linkage between a document and its dependencies and all corresponding binary resources.

The syntactic structure of the Batch POST method is illustrated in Example 7 which publishes an A.xsd document which has a dependency on the B.xsd document. Notable items specific to the structure are in bold:

Example 7: Batch Post Construct Example

```
POST /s-ramp/xsd HTTP/1.1
Host: example.org
Content-Length: nnnn
Content-Type: multipart/related;version 1.1;msgtype=request;boundary=example-
bound;type="application/atom+xml;type=entry";start="<12@example.org>"
Slug: The Beach
MIME-Version: 1.0

--example-bound–
Content-Type: application/atom+xml;type=entry
Content-ID: <12@example.org >

<entry xmlns=”http://www.w3.org/2005/Atom”
        xmlns:s-ramp=”http://s-ramp.org/xmlns/2010/s-ramp”>
    <id>urn:uuid:aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a</id>
    <updated>2009-05-26T13:13:55.014+02:00</updated>
    <title type=“text”>A.xsd</title>
    <published>2009-05-26T13:13:55.013+02:00</published>
    <author>
        <name>Bellwood</name>
    </author>
    <contributor>
        <name>Pospisil</name>
    </contributor
    <summary type="text">A.xsd schema document</summary>
    <link href="<56@example.org>" type="application/atom+xml;type=entry" rel="related"
        title="Imported XSD"/>
    <content src=”<34@example.org>”/>
    <!– S-RAMP defined categorizations identifying class of data represented by
```

```
        this entry -->
    <category term="XsdDocument" label="XML Schema"
        scheme="urn:x-s-ramp:2010:type" />
    <s-ramp:artifact
        ...rest of artifact definition goes here...
    </ s-ramp:artifact>
</entry>

--example-bound--
Content-Type: application/xml
Content-Description: The root XSD document
Content-Transfer-Encoding: base64
Content-ID: <34@example.org>

...XML content for A.xsd...

--example-bound--
Content-Type: application/atom+xml;type=entry
Content-ID: <56@example.org>

<entry xmlns="http://www.w3.org/2005/Atom"
        xmlns:s-ramp="http://s-ramp.org/xmlns/2010/s-ramp">
    <id>urn:uuid:bbbbbbbb-bbbb-bbbb-bbbb-bbbbbbbbbb6b</id>
    <title type="text">B.xsd</title>
    <content src="<78@example.org>"/>
    ...rest of entry document goes here...
</entry>

--example-bound--
Content-Type: application/xml
Content-Description: The imported XSD document
Content-Transfer-Encoding: base64
Content-ID: <78@example.org>

...XML content for B.xsd...

--example-bound--
```

The Batch POST method is intended to support the publication of new documents which have dependencies, although it can also include non-document artifacts. Such artifacts will not have a corresponding binary section in the body. S-RAMP servers SHALL process the entire encapsulated payload in a Batch POST as a group and SHALL perform any dependent processing necessary. All parts of the body SHOULD be published successfully, or the entire request SHOULD fail and be rolled back. However, rolling back the entire batch POST on a failed request is implementation specific as some implementations MAY choose to create only those artifacts which are valid, thus doing a partial create from the Batch POST. Regardless of whether a failure results in a complete rollback or a partial create, the implementation MUST return a failure response and in the body of the response provide an explanation of the failure.

The response from a Batch POST in S-RAMP SHALL provide a return code which indicates success or failure. A successful response MUST be an HTTP 200 OK and a failure response MUST be an HTTP 409 Conflict.

In the case of a successful response from the "encapsulating" HTTP POST the response would contain a set of boundary delineated HTTP responses, which in this example would be a set of boundary delineated Media Link Entries corresponding to the two XSD files which were published.

Example 8 – Successful Batch POST Response

```
HTTP/1.1 200 OK
Date: Tues, 26 May 2009 13:13:55 GMT+02:00
Content-Length: 520
Content-Type: multipart/http;version=1.1;msgtype=response;boundary=batch
Mime-Version: 1.0

--batch
Content-ID: <aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa12@xample.org>
```

```
HTTP/1.1 201 Created
Date: Tues, 26 May 2009 13:13:55 GMT+02:00

{Updated Atom:entry for Resource A omitted for clarity}

--batch

Content-ID: <aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa67@example.org>

HTTP/1.1 201 Created
Date: Tues, 26 May 2009 13:13:55 GMT+02:00

{Updated Atom:entry for Resource B omitted for clarity}
```

In the case where the Batch POST was unsuccessful and the server rolled back the entire Batch POST request, the unsuccessful response from the "encapsulating" HTTP POST would be an HTTP 409. The response would contain an explanation of the error with enough information to allow the user to recognize the conflict. Ideally, the information provided would also allow the user to fix the conflict, however this MAY not always be possible.

Example 9 – Failed Batch POST Response – Complete Rollback

```
HTTP/1.1 409 Conflict
Date: Tues, 26 May 2009 13:13:55 GMT+02:00
Content-Length: 520
Content-Type: multipart/http;version=1.1
      ;msgtype=response;boundary=batch
Mime-Version: 1.0


--batch

Content-ID: <aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa67@example.org>

HTTP/1.1 409 Conflict
Date: Tues, 26 May 2009 13:13:55 GMT+02:00


{Explanation of error condition}
```

In the case where the Batch POST request was unsuccessful and the server chose to do a partial create of the Batch POST request, the unsuccessful response from the "encapsulating" HTTP POST would be an HTTP 409. The response would contain an explanation of the error with enough information to allow the user to recognize the content of the content that were successfully published and the content that was not successful. Ideally, the information provided would also allow the user to fix the conflict, however this MAY not always be possible.

Example 10 – Failed Batch POST Response – Partial Create

```
HTTP/1.1 409 Conflict
Date: Tues, 26 May 2009 13:13:55 GMT+02:00
Content-Length: 520
Content-Type: multipart/http;version=1.1
      ;msgtype=response;boundary=batch
Mime-Version: 1.0

--batch
Content-ID: <aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa12@xample.org>

HTTP/1.1 201 Created
Date: Tues, 26 May 2009 13:13:55 GMT+02:00

{Updated Atom:entry for Resource A omitted for clarity}
```

```
--batch

Content-ID: <aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa67@example.org>

HTTP/1.1 409 Conflict
Date: Tues, 26 May 2009 13:13:55 GMT+02:00


{Explanation of error condition}
```

2.3.5.2.2 ZIP Publishing Using POST of ZIP File

This method of publishing to an S-RAMP compliant repository can provide significant convenience for clients. The behavior described herein represents some departure from the APP specification in order to support the REQUIRED capabilities, but wherever possible, it adheres to its spirit.

To publish using a zip file, one simply performs an HTTP POST of a zip file containing the desired media resources and/or atom:entry documents to the appropriate S-RAMP collection URI. It is useful to discuss several behavioral characteristics of publishing using zip files:

- The zip file can have arbitrary internal structure (e.g., flat, or organized using folders as desired). S-RAMP uses the UUID of an artifact as its official identifier rather than any user supplied file structure. S-RAMP repositories MAY, but are not REQUIRED to, also utilize or retain client supplied file structure information.
- The zip file can contain media resources (e.g., XSD documents), corresponding Media Link Entries which contain associated metadata, or plain atom:entry file(s) (.atom files in the zip) representing non-document based repository artifacts.
- The convention for associating metadata with a resource in a zip is to append ".atom" to its file-id for the Artifact Entry document. For example, for the resource "somename.ext", the file with the name "somename.ext.atom" corresponds to its Atom Media Link Entry containing the metadata associated with the document.
- When published, the S-RAMP server processes the zip file, extracting the constituent files. Each resource is published, together any associated Media Link Entry and any other Artifact Entry documents provided.
- An S-RAMP repository does not persist or publish the zip file itself after it has been processed.
- All S-RAMP artifacts require a unique uuid property (in their s-ramp:artifact section) and atom:id. Resources which lack corresponding Media Link Entries, or those whose Media Link Entry files have no user specified uuid property and atom:id, or a user specified uuid property and atom:id which does not already exist in the repository, are considered new and treated as if they were published using a POST.
- Artifact Entry documents having a user supplied uuid property and atom:id which already exists in the repository, are treated as if they were PUT, which causes the existing artifact to be replaced with the copy in the zip file.
- As discussed in Section 2.3.2 of the Foundation Document, documents which have a Derived Model associated with them cannot be updated in the repository. They MUST be removed and republished. The implicit updates of documents which have a Derived Model is, therefore, not supported when publishing a ZIP file. Attempts to perform an update on such a document will result in the HTTP error "403" Forbidden.

Zip files are published to the `/s-ramp` S-RAMP URI space:

The basic syntax for publication using ZIP files is then:

```
POST /s-ramp HTTP/1.1
Host: example.org
Content-Type: application/zip
Content-Length: nnn
Accept: multipart/related

{ ...binary zip data...}
```

The S-RAMP server will process the component files in the zip file, placing each in the correct S-RAMP collection in the S-RAMP URI space, based upon introspection of the files.

For example, assume we wish to publish a set of related resources: a.wsdl, which has import statements for b.xsd and c.xsd:

MyFiles.zip content:

- a.wsdl
- a.wsdl.atom
- b.xsd
- b.xsd.atom
- c.xsd
- c.xsd.atom

Upon receipt of a POST request, the S-RAMP server processes the zip file and instantiates an S-RAMP artifact in the repository for each of the three documents, which in the Atom Binding are represented with Media Link Entries with document content elements for the a.wsdl, b.xsd and c.xsd media resources respectively. The user supplied Media Link Entry files extracted from the zip are used to create these S-RAMP artifacts.

The response to the POST, by the S-RAMP server returns a multipart message of content type:

```
multipart/related; boundary="== zipdivider==";  type="
application/atom+xml;type=entry;charset="utf-8 ";
```

which consists of a set of boundary delineated Media Link Entries, one for each Atom:entry document which was in the zip file, as well as an Atom Media Link Entry document for each media resource in the zip (clients MAY provide these documents together with their corresponding media resource documents in the zip, or let the server create them and then update them in a second step). The content of each entry document MAY have been altered by the server during publication (e.g., date/time of the update, setting of the ID, if not provided by in the request, etc.) The client is responsible for noting any updates which have been made.

As with the HTTP Batch approach described earlier, ALL operations implied by the zip file contents MUST succeed in order for ANY of them to succeed. If any one fails, the entire zip request is rolled back. Example 11 below provides a response for the above zip file scenario. It omits substantial content for brevity:

Example 11 - Response from Publish using ZIP File Method

```
HTTP/1.x 200 OK
Date: Fri, 2 Feb 2009 17:17:11 GMT
Content-Length: nnn
Content-Type: multipart/related;boundary="==zipdivider==";type="application/atom+xml;
type=entry;charset="utf-8 ";
Location: http://example.org/s-ramp/wsdl/WsdlDocument/aaaaaaaa-aaaa-aaaa-aaaa-
   aaaaaaaaaa6a

--==zipdivider==--

<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom"
>
     xmlns:s-ramp="http://s-ramp.org/xmlns/2010/s-ramp">
       <title>b.xsd</title>
        <id>urn:uuid:aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6b</id>
        <updated>2009-02-03T17:17:08Z</updated>
        <author><name>Pospisil</name></author>
       <summary type="text" />
```

```
        <content type="application/xml "
                src="http://example.org/s-ramp/xsd/aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6b/>
        <link type="application/atom+xml;type=entry" rel="edit-media"
            href="http://example.org/s-ramp/xsd/aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6b"/>
        <link type="application/atom+xml;type=entry" rel="edit"
           href="http://example.org/media/s-ramp/xsd/aaaaaaaa-aaaa-aaaa-aaaa-
aaaaaaaaaa6b”/>
        <s-ramp:artifact>
        {Complete/updated media link entry of b.xsd compliant with the s-ramp schemas …}
        </ s-ramp:artifact>
</entry>

--==zipdivider==

<entry xmlns=”http://www.w3.org/2005/Atom”
     xmlns:s-ramp=”http://s-ramp.org/xmlns/2010/s-ramp”>
        <title>c.xsd</title>
         <id>urn:uuid:aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6c</id>
         <updated>2009-02-03T17:17:08Z</updated>
         <author><name>Pospisil</name></author>
        <summary type="text" />
        <content type="application/xml"
                src=http://example.org/s-ramp/xsd/aaaaaaaa-aaaa-aaaa-aaaa-
   aaaaaaaaaa6c/media />
        <link type="application/atom+xml;type=entry" rel="edit-media"
            href="http://example.org/s-ramp/xsd/aaaaaaaa-aaaa-aaaa-aaaa-
aaaaaaaaaa6c/media"/>
        <link type="application/atom+xml;type=entry" rel="self"
           href="http://example.org/media/s-ramp/xsd/aaaaaaaa-aaaa-aaaa-aaaa-
aaaaaaaaaa6c”/>
        <link type="application/atom+xml;type=entry" rel="edit"
           href="http://example.org/media/s-ramp/xsd/aaaaaaaa-aaaa-aaaa-aaaa-
aaaaaaaaaa6c”/>
        < s-ramp:artifact>
        {Complete/updated media link entry artifact of c.xsd compliant with the s-ramp
schemas …}
        </ s-ramp:artifact>
</entry>

--==zipdivider==

<entry xmlns="http://www.w3.org/2005/Atom">
     <title>a.WSDL</title>
     <id> aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a</id>
     <updated>2009-02-03T17:17:08Z</updated>
     <author><name>Pospisil</name></author>
     <summary type="text" />
     <content type="application/xml"
        src="http://example.org/s-ramp/wsdl/WsdlDocument/aaaaaaaa-aaaa-aaaa-aaaa-
aaaaaaaaaa6a/media"/>
     <link type="application/atom+xml;type=entry" rel="edit-media"
        href="http://media.example.org/s-ramp/wsdl/WsdlDocument/
           aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/media" />
     <link type="application/atom+xml;type=entry" rel="self"
        href="http://example.org/media/s-ramp/wsdl/WsdlDocument/
```

```
            aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a" />
      <link type="application/atom+xml;type=entry" rel="edit"
         href="http://example.org/media/s-ramp/wsdl/WsdlDocument/
            aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a" />
     <s-ramp:artifact>
     {Complete/updated media link entry document for a.wsdl, compliant with the
      S-RAMP schema}
     </s-ramp:artifact>
</entry>

--==zipdivider==--
```

2.3.5.3 Retrieving Repository Artifacts

HTTP GET is used to retrieve Artifact Entries. Entry documents can be retrieved individually by performing an HTTP GET against a member resource URI, or as a feed of Entry Documents by performing an HTTP GET against a collection. Requests to retrieve an Artifact Entry document from the incorrect Artifact Type Model will result an HTTP "404" Not Found.

Several examples are provided below using this sample URL format:

Format to request a specific Artifact Entry document:

```
GET /s-ramp/{artifactModel}/{artifactType}/{uuid:artifact} HTTP/1.1
Host: example.org
```

Format to request document content:

```
GET /s-ramp/{artifactModel}/{artifactType}/{uuid:artifact}/media HTTP/1.1
Host: example.org
```

Format to request a feed of summary Artifact Entry documents from an artifact collection which is defined in the Service Document:

```
GET /s-ramp/{artifactCollection} HTTP/1.1
Host: example.org
```

Example 12 illustrates several requests:

Example 12 - Various Requests to Retrieve Repository Artifact(s) or Content

To request a specific XsdDocument full Artifact Entry:

```
GET /s-ramp/xsd/XsdDocument/aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a HTTP/1.1
Host: example.org
```

To request a specific XsdDocument content itself:

```
GET /s-ramp/xsd/XsdDocument/aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/media HTTP/1.1
Host: example.org
```

To request a feed of all Media Link Entry documents corresponding to each XSD document instance in the repository, in summary form:

```
GET /s-ramp/xsd/XsdDocument HTTP/1.1
```

```
Host: example.org
```

To request a feed of all Service Implementation Model entries representing Organization artifact instances, in summary form:

```
GET /s-ramp/serviceImplementation/Organization HTTP/1.1
Host: example.org
```

Notes:

- Requesting a feed of summary Artifact Entries for an entire collection can result in large result sets. The Atom defined mechanism for paging of feeds SHALL be supported by S-RAMP servers.
- The HTTP allow header can be used to indicate the HTTP methods which can be executed against the request URI.

2.3.5.3.1 Resolving Internal References

Some documents have dependencies on other documents. From our earlier scenario, a WSDL document might import an XSD document. When the WSDL document was published to the repository, the XSD document would have been published as well, but the original WSDL document published would not typically have the correct import statement to reference the XSD file which is actually persisted in the repository. More likely, it would be a local reference based on the development environment from which both files originated. It is important that the repository support the capability for tooling to resolve these dependencies to the correct version of the document which is actually published in the repository. S-RAMP provides two implementation choices for achieving this. Compliant implementations SHALL support at least one of them.

1. When returning a requested document, the repository MAY dynamically re-write the document's references to other documents to correctly reference the appropriate file(s) stored in the repository. The actual document stored in the repository is unaffected. Only the serialized content returned in a GET is altered as the content is returned.
2. When the URL of a document containing dependent links is requested, the repository MAY instead redirect to another URL of its own creation, against which all relative path references to dependent documents will resolve properly with the redirected URL as a base. S-RAMP does not describe how such an implementation is done, only the REQUIRED behavior when implementations choose to support this option.

Regardless of which option is implemented, its activation is achieved by adding the *type=relative* query string to the URL in the request (a sample URL format is shown here):

```
/s-ramp/<artifactModel>/<artifactType>/{uuid:artifactDocument}/media?type=relative
```

2.3.5.4 Editing an Artifact Entry

Example 5 - Updating an Initial Media Link Entry with already described how to perform an update of an existing Atom entry, and as described in Section 2.3.5.2, it is also possible to use PUT as part of a multi-entry update using either the HTTP Batch or ZIP publishing methods supported in S-RAMP. When the update uses the HTTP Batch technique described in Section 2.3.2, the boundary delineated section applying to the artifact to be updated simply uses a PUT instead of a POST. When the update is requested as part of a POST using a zip file as described in Section 2.3.5.2.2, the PUT is implicit. All zip file publishing is done using a POST of the zip file, but when an individual entry within the zip file references an Artifact Entry uuid property which already exists in the repository; it is treated as an update during processing.

2.3.5.5 Deleting an Artifact Entry

Deletion of an Artifact Entry is accomplished using HTTP DELETE, with this sample syntax:

To delete an Artifact Entry (for media link entries, this implicitly deletes the corresponding content as well):

```
DELETE /s-ramp/{artifactModel}/{artifactType}/{uuid:artifact} HTTP/1.1
Host: example.org
```

To delete content (this implicitly deletes the corresponding media link entry as well):

```
DELETE /s-ramp/{artifactModel}/{artifactType}/{uuid:artifact}/media HTTP/1.1
Host: example.org
```

The above URL for the entry with the indicated Artifact Entry uuid property MUST already resolve to an existing Artifact Entry in the repository in order for it to be deleted. All artifacts not belonging to a Derived Model can be deleted by a client. The deletion of an Artifact Entry also removes all of its relationships. Additional information on how this affects reporting of backward relationship feeds in the Fine Grained View can be found in Section 2.4.1.7.

If an artifact is deleted which is the Target Entry of a relationship instance owned by some other Source Entry, then that relationship instance is also deleted in that Source Entry.

The following HTTP errors MAY be returned:

- 404 Not Found (no matching Artifact Entry uuid property in repository)
- 403 Forbidden (returned in response to an attempted delete of a Derived Artifact)

2.4 Fine Grained Views

The three Fine Grained Views in the Atom Binding for S-RAMP provide a mechanism for working with each of the three classes of S-RAMP metadata: relationships, properties and classifications. While S-RAMP compliant implementations MAY choose to implement none, any, or all of these features, implementation of any of these features SHALL conform with the applicable interfaces described in these sections. Updates to metadata items using the Fine Grained Views implicitly changes the Artifact Entry which owns the metadata.

2.4.1 S-RAMP Relationships

S-RAMP models relationship metadata in the Atom Binding as resources in order to facilitate their manipulation separately from the Atom Source Entry which manages them. This is particularly useful when many relationships are present, since the fine-grained method allows manipulation of one relationship at a time without having to explicitly update the Atom Source Entry with which it is associated. Several concepts and terms are useful:

- A **relationship** is a concept that represents an association between a single Source Entry and a single Target Entry (each of which represents an S-RAMP artifact). Relationships are modeled in Atom with a **Relationship Entry**, which is an Atom entry document describing the relationship, including links to the Source Entry and Target Entry, the **Relationship Type**, as well as specific categorizations which are described later that provide relevant metadata pertaining to the Relationship Entry.
- A **Relationship Type** is a name which describes the purpose or meaning of that relationship (e.g., “includedXsds” or “similarXsds”). There can be more than one relationship having the same **Relationship Type**. A **Relationship Type Entry** is an Atom entry document which describes a particular set of Relationship Types. It contains a link to the applicable Relationship feed as well as specific categorizations which are described later that provide relevant metadata pertaining to the Relationship Type Entry.

- A **Backward Relationship Feed** is a special kind of feed whose members are Relationship Entry documents. Links to such *backwardRelationships* feed(s) are placed in the Target Entry document corresponding to the relationship target represented by a *modeled* or *derived* relationship's Target Entry link. These feeds are provided for the convenience of clients to simplify artifact navigation. Additional information is provided in Section 2.4.1.1.

2.4.1.1 Relationship Feeds

S-RAMP defines several Atom feeds which are used to access fine-grained support for relationships. These allow clients to retrieve details about each of the Relationship Types and instances associated with the Artifact Entry having these relationships. When a server implementation supports the Fine Grained View for relationships, the Atom entry document representing the relationship's source artifact (Source Entry) SHALL contain links to the following Atom feed(s):

- Link to the ***relationships*** feed of all relationships. Resolving the link to this feed will return a feed of summary Relationship Entries for every relationship instance owned by the Source Entry, regardless of the Relationship Type associated with each. This **feed** link SHALL have a *rel* attribute as follows:

  ```
  rel="urn:x-s-ramp:2010:relationships"
  ```

  For example:

  ```
  <link title="All Relationships"
    href="http://example.org/s-ramp/xsd/XsdDocument/
      {uuid:source.xsd}/relationships"
    type="application/atom+xml;type=feed"
    rel="urn:x-s-ramp:2010:relationships" />
  ```

- Link to the ***relationshipTypes*** feed of all Relationship Types (e.g., "includedXsds", "similarXsds"). Resolving the link to this feed will return a feed of all the summary Relationship Type Entries. There is only one such entry in this feed for each Relationship Type represented across the entire set of relationship instances owned by the Source Entry. This **feed** link SHALL have a *rel* attribute as follows:

  ```
  rel="urn:x-s-ramp:2010:relationshipTypes"
  ```

  For example:

  ```
  <link title="Relationships Types"
    href="http://example.org/s-ramp/xsd/XsdDocument/
      {uuid:source.xsd}/relationshipTypes"
    type="application/atom+xml;type=feed"
    rel="urn:x-s-ramp:2010:relationshipTypes" />
  ```

- Link(s) to individual *relationships* feed(s) of all relationships sharing a particular Relationship Type. These link(s) will appear in the Source Artifact corresponding to each Relationship Type Entry. These links provide a means to retrieve Relationship Entry documents owned by the Source Entry corresponding to a particular Relationship Type. These **feed** links SHALL have a *rel* attribute of the following form:

  ```
  rel="urn:x-s-ramp:2010:relationships:{Relationship Type}"
  ```

  For example:

  ```
  <link title="Relationships of type includedXsds "
    href="http://example.org/s-ramp/xsd/XsdDocument/
        {uuid:source.xsd}/relationships/includedXsds"
    type="application/atom+xml;type=feed"
    rel="urn:x-s-ramp:2010:relationships:includedXsds" />
  ```

While all S-RAMP relationships are unidirectional, the S-RAMP Atom Binding provides a convenience feature supported for at least all *modeled* and *derived* relationships which facilitates a client's ability to navigate backward from the Target Entry of a given relationship to the Source Entry, without the need for performing a query to discover the Source Entry side of a relationship instance. As noted in Section 2.4.1 above, these are called Backward Relationship Feeds.

A *backwardRelationships* feed is defined corresponding to each *relationships* feed (including those for specific Relationship Types). These *backwardRelationships* feed(s) are created by the server by placing the same

Relationship Entry that occurs in the *relationships* feed of the relationship's Source Entry into the corresponding *backwardRelationships* feed of the Target Entry referenced by that relationship. In addition, applicable Relationship Type specific *backwardRelationships/{Relationship Type}* feeds are also present in the Target Entry. Clients can use these feeds to navigate backward using the Source Entry link associated with a relationship's Target Entry.

Backward Relationship Feeds are read-only. Clients cannot add or remove a Relationship Type Entry documents from any *backwardRelationships* feed. The S-RAMP server provides these feeds in the Target Entry serialization only as a convenience to clients. All relationships are still managed from the applicable Source Entry's relationship feed links.

The Target Entry document representing the relationship's target artifact SHALL contain links to the following Atom feed(s):

- Link to the ***backwardRelationships*** feed. Resolving this link will return a feed of summary Relationship Entry documents for at least every *modeled* and *derived* kind of relationship instance, regardless of its Relationship Type, for which the Target Entry link resolves to this Artifact Entry. Note that this feed will be empty if there is no Source Entry with a relationship having this Artifact Entry as its target. This **feed** link SHALL have a *rel* attribute of the following form:

  ```
  rel="urn:x-s-ramp:2010:backwardRelationships"
  ```

  For example:

  ```
  <link title="Back Links from this Target Entry for all Relationship Types"
    href="http://example.org/s-ramp/xsd/XsdDocument/
      {uuid:foo.xsd}/backwardRelationships"
    type="application/atom+xml;type=feed"
    rel="urn:x-s-ramp:2010:backwardRelationships" />
  ```

- Link(s) to individual Relationship Type specific *backwardRelationships*/{Relationship Type} feed(s) of all relationships sharing a particular Relationship Type, and whose targets are this Artifact Entry. These provide a means to retrieve all Relationship Entry documents of a particular Relationship Type for use in backward navigation to the source side of each such relationship. These **feed** links SHALL have a *rel* attribute of the following form:

  ```
  rel="urn:x-s-ramp:2010:backwardRelationships:{Relationship Type}"
  ```

  For example:

  ```
  <link title="Back Links for serviceImplementation model relationships of type
  hasServiceEndpoint"
    href="http://example.org/s-ramp/serviceImplementation/ServiceEndpoint/
      {uuid:target.xsd}/backwardRelationships/hasServiceEndpoint"
    type="application/atom+xml;type=feed"
    rel="urn:x-s-ramp:2010:backwardRelationships:hasServiceEndpoint" />
  ```

Example 13 below illustrates both *relationships* and *backwardRelationships* feeds using three types of relationships (*derived*, *modeled* and *generic*). It uses the same summary Atom (media link) entry which corresponds to the "accountingTypes" XSD document artifact found in Section 2.3.2, except that this version includes the necessary links to support the Fine Grained View for Relationships. To illustrate all these features, lets add three more XSD documents, and an S-RAMP Service Implementation Model ServiceInstance artifact called "myServiceInstance" which has a "describedBy" relationship to the "accountingTypes" document.

Example 13 - Complex Relationship Scenario Summary Entry

Assume a set of four documents:

1. "customer.xsd", which includes
2. "accountingTypes.xsd", which includes
3. "dataTypes.xsd"
4. "related.xsd" (a similar schema which is related to "accountingTypes.xsd")

The following relationships exist among the Artifact Entries representing each of these four document artifacts and the Service Implementation Model artifact:

- Relationship 1:
  - Type: “includedXsds”
  - Kind: “derived”
  - Source: “customer.xsd” entry
  - Target: “accountingTypes.xsd” entry
- Relationship 2:
  - Type: “includedXsds”
  - Kind: “derived”
  - Source: “accountingTypes.xsd” entry
  - Target: “dataTypes.xsd” entry
- Relationship 3:
  - Type: “similarXsds”
  - Kind: “generic”
  - Source: “accountingTypes.xsd” entry
  - Target: “related.xsd” entry
- Relationship 4:
  - Type: “describedBy”
  - Kind: “modeled”
  - Source: “myServiceInstance” entry
  - Target: “accountingTypes.xsd” entry

The summary Source Entry below corresponding to the “accountingTypes.xsd” artifact has relationship instances for the “includedXsds” and “similarXsds” Relationship Types associated with it, and thus exposes the various feeds shown:

```
<entry xmlns=http://www.w3.org/2005/Atom xmlns:s-ramp=”http://s-ramp.org/xmlns/2010/s-ramp”>
    <id>urn:uuid:aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a</id>
    <updated>2009-05-26T13:13:55.013+02:00</updated>
    <title type=“text”>accountingTypes.xsd</title>
    <published>2009-05-26T13:13:55.013+02:00</published>
    <author>
        <name>Bellwood</name>
    </author>
    <contributor>
        <name>Pospisil</name>
    </contributor>
    <summary type="text">accountingTypes.xsd schema document</summary>
    <content
        type="application/xml"
        src="http://example.org/s-ramp/xsd/XsdDocument/
        aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/media"/>
    <link type="application/atom+xml;type=entry" rel="self"
        href="http://example.org/s-ramp/xsd/XsdDocument/
        aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a” />
    <link type="application/atom+xml;type=entry" rel="edit-media"
```

```
    href="http://example.org/s-ramp/xsd/XsdDocument/
    aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/media" />
<link type="application/atom+xml;type=entry" rel="edit"
    href="http://example.org/s-ramp/xsd/XsdDocument/
    aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a" />

<!-- Link to relationships feed of all Relationship entries
       over all Relationship Types. In this example, that includes Relationships #2
           and #3 above. -->
<link title="All Relationships"
    href="http://example.org/s-ramp/xsd/XsdDocument/
    aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/relationships"
     type="application/atom+xml;type=feed"
    rel="urn:x-s-ramp:2010:relationships" />
<!-- Link to backwardRelationships feed containing all Relationship Entries of any
Relationship Type whose target is this Artifact Entry. In this example, that includes
Relationships #1 and #4 above. -->
<link title="All Backward Relationship Targets"
    href="http://example.org/s-ramp/xsd/XsdDocument/
    aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/backwardRelationships"
     type="application/atom+xml;type=feed"
    rel="urn:x-s-ramp:2010:backwardRelationships" />

<!-- Link to relationshipTypes feed of Relationship Type Entries.  In this
    example, that means entries for the "includedXsds" and "similarXsds"
    Relationship Types.  -->
<link title="All Relationship Types"
    href="http://example.org/s-ramp/xsd/XsdDocument/
    aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/relationshipTypes"
     type="application/atom+xml;type=feed"
    rel="urn:x-s-ramp:2010:relationshipTypes" />

<!-- Link to feed of all Relationship Entries whose
       Relationship Type = "includedXsds" -->
<link title="All includedXsds Type Relationships"
    href="http://example.org/sramp/xsd/XsdDocument/
    aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/relationships/includedXsds"
     type="application/atom+xml;type=feed"
    rel="urn:x-s-ramp:2010:relationships:includedXsds" />
<!-- Link to backwardRelationships feed containing all Relationship Entries of
Relationship Type = "includedXsds", whose corresponding Target Entry is this Artifact
Entry. In this example, that includes only Relationship #1 above. -->
<link title="All includedXsds Backward Relationship Targets"
href="http://example.org/s-ramp/xsd/XsdDocument/
    aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/backwardRelationships/includedXsds"
     type="application/atom+xml;type=feed"
    rel="urn:x-s-ramp:2010:backwardRelationships:includedXsds" />

<!-- Link to backwardRelationships feed containing all Relationship Entries of
Relationship Type = "describedBy", whose corresponding Target Entry is this Artifact
Entry. In this example, that includes only Relationship #4 above. There are no forward
describedBy relationships in this example. -->
<link title="All describedBy Backward Relationship Targets"
href="http://example.org/s-ramp/xsd/XsdDocument/
    aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/backwardRelationships/describedBy"
     type="application/atom+xml;type=feed"
    rel="urn:x-s-ramp:2010:backwardRelationships:describedBy" />
```

```
    <!-- Link to feed of all Relationship Entries whose
           Relationship Type = "similarXsds". In this example, that includes only
           Relationship #3 above. Note that there are no backward feeds for generic
           relationships. -->
    <link title="All similarXsds Type Relationships"
        href="http://example.org/sramp/xsd/XsdDocument/
        aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/relationships/similarXsds"
         type="application/atom+xml;type=feed"
        rel="urn:x-s-ramp:2010:relationships:similarXsds" />


    <!– S-RAMP defined categorizations identifying class of data represented by
        this entry -->
    <category term="XsdDocument" label="XML Schema Document"
        scheme="urn:x-s-ramp:2010:type" />
</entry>
```

2.4.1.2 Relationship Entry Documents

Resolving the link to a *relationships* feed will return a feed of Relationship Entry documents. A Relationship Entry document is a valid Atom entry document which contains information about a single relationship instance associated with a Source Entry. S-RAMP requires that the representation of summary and full Relationship Entry documents SHALL be the same. This makes it possible to retrieve all relationship information for a Source Entry in a single step by retrieving the *relationships* feed. The following items SHALL appear in a Relationship Entry Document:

- The basic Atom elements REQUIRED for a legal entry document. The REQUIRED atom:author element value SHALL be set by the server to match the value found in the Source Entry.
- An Atom link to the Source Entry. This link SHALL use the following *rel* attribute value:
    - rel="urn:x-s-ramp:2010:relationship:source".
- An Atom link to the Target Entry. This link SHALL use the following *rel* attribute value:
    - rel="urn:x-s-ramp:2010:relationship:target".
- An Atom link to the Relationship Type Entry from the *relationshipTypes* feed which corresponds to the Relationship Type of this Relationship Entry. This entry SHALL use the following *rel* attribute value:
    - rel="urn:x-s-ramp:2010:relationship:type"
- Atom:content text element describing the entry document.
- A structured extension element s-ramp:relationshipData containing the Relationship Type, source UUID and target UUID of this Relationship Entry.
- Atom:category elements describing the particular Relationship Entry:
    - The entry type:
        - scheme="urn:x-s-ramp:2010:type"
        - The only valid value for the *term* attribute here is "relationship"
    - The kind of relationship:
        - scheme="urn:x-s-ramp:2010:kind"
        - Valid values for *term* attribute are:
            - "derived"
            - "modeled"
            - "generic"

The example below builds on the one in Section 2.4.1.1, with the *relationships* feed of all relationships owned by the Source Artifact. With reference to that prior example, this feed contains Relationship Entries for relationships #2 and #3:

Example 14 - Complex Relationship Scenario Relationships Feed

```
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:s-ramp="http://s-ramp.org/xmlns/2010/s-
ramp">
  <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344eccc0</id>
  <link href="http://example.org/s-ramp/xsd/XsdDocument/
    1225c695-cfb8-4ebb-aaaa-80da344eccc0/relationships"
    rel="self" type="application/atom+xml;type=feed" />
  <updated>2009-05-26T13:13:55.013+02:00</updated>
  <title type="text">accountingTypes.xsd : All relationships feed</title>
  <author>
    <name>Bellwood</name>
  </author>

  <!—First Relationship Entry in the feed -->
  <entry>
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344eccc1</id>
    <updated>2009-05-26T13:13:55.013+02:00</updated>
    <title type="text">
        includedXsds Relationship for accountingTypes.xsd Source Entry
    </title>
    <published>2009-05-26T13:13:55.013+02:00</published>
    <!-- Note that derived relationship entries do not have an "edit" link -->
    <link type="application/atom+xml;type=entry"
        href="http://example.org/s-ramp/xsd/XsdDocument/
        aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/relationships/includedXsds/
        1225c695-cfb8-4ebb-aaaa-80da344eccc1" rel="self" />

    <!--  Content element identifies this as a Relationship Entry -->
    <content type="text">Relationship Entry</content>

    <-- S-RAMP structured extension for Relationship Entry data -->
    <s-ramp:relationshipData>
        <s-ramp:relationshipType>includedXsds</s-ramp:relationshipType>
        <s-ramp:sourceId>aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a</sourceId>
        <s-ramp:targetId>aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6b</targetId>
    </ s-ramp:relationshipData>

    <!— Link to relationship's Source Entry -->
    <link title="Relationship Source Entry"
        href="http://example.org/s-ramp/xsd/XsdDocument/
            aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a"
        type="application/atom+xml;type=entry"
        rel="urn:x-s-ramp:2010:relationship:source"/>

    <!— Link to relationship's Target Entry -->
    <link title="Relationship Entry"
        href="http://example.org/s-ramp/xsd/XsdDocument/
            aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6b"
        type="application/atom+xml;type=entry"
```

```
      rel="urn:x-s-ramp:2010:relationship:target"/>

  <-- Link to corresponding includedXsds Relationship Type Entry -->
  <link href="http://example.org/s-ramp/xsd/XsdDocument/
      aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/relationshipTypes/
      1225c695-cfb8-4ebb-aaaa-80da344eddd1”
      type="application/atom+xml;type=entry"
      rel="urn:x-s-ramp:2010:relationship:type" />

 <--  Categorizations describing the Relationship Entry -->
  <category term="derived" label="Derived S-RAMP relationship."
             scheme="urn:x-s-ramp:2010:kind" />
  <category term="relationship" label="Relationship Entry type"
             scheme="urn:x-s-ramp:2010:type" />
</entry>


<!—Second Relationship Entry in the feed -->
<entry>
  <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344eccc2</id>
  <updated>2009-05-26T13:13:55.013+02:00</updated>
  <title type=“text”>
      similarXsds relationship for accountingTypes.xsd Source Entry.
  </title>
  <published>2009-05-26T13:13:55.013+02:00</published>
  <link href="http://example.org/s-ramp/xsd/XsdDocument/
      aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/relationships/similarXsds/
      1225c695-cfb8-4ebb-aaaa-80da344eccc2”
      type="application/atom+xml;type=entry" rel="self" />

  <-- Generic Relationship Entry documents include an “edit” link: -->
  <link href="http://example.org/s-ramp/xsd/XsdDocument/
      aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/relationships/similarXsds/
      1225c695-cfb8-4ebb-aaaa-80da344eccc2”
      type="application/atom+xml;type=entry" rel="edit" />

  <!--  Content element identifies this as a Relationship Entry -->
  <content type="text">Relationship Entry</content>

  <-- S-RAMP structured extension for Relationship Entry data -->
  <s-ramp:relationshipData>
      <s-ramp:relationshipType>similarXsds</s-ramp:relationshipType>
      <s-ramp:sourceId>aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a</sourceId>
      <s-ramp:targetId>aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6c</targetId>
  </s-ramp:relationshipData>

  <!—Link to Relationship’s Source Entry -->
  <link title="Relationship’s Source Entry"
      href="http://example.org/s-ramp/xsd/XsdDocument/
          aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a”
      type="application/atom+xml;type=entry"
      rel="urn:x-s-ramp:2010:relationship:source"/>

  <!—Link to Relationship’s Target Entry -->
```

```
    <link title="Relationship's Target Entry"
        href="http://example.org/s-ramp/xsd/XsdDocument/
            aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6c"
        type="application/atom+xml;type=entry"
        rel="urn:x-s-ramp:2010:relationship:target"/>

    <-- Link to corresponding similarXsds Relationship Type Entry -->
    <link href="http://example.org/s-ramp/xsd/XsdDocument/
        aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/relationshipTypes/
               1225c695-cfb8-4ebb-aaaa-80da344eddd2"
        type="application/atom+xml;type=entry"
        rel="urn:x-s-ramp:2010:relationship:type" />

   <--  Categorizations describing the Relationship Entry -->
    <category term="generic" label="Generic S-RAMP relationship."
               scheme="urn:x-s-ramp:2010:kind" />
    <category term="relationship" label="S-RAMP Relationship Entry"
        scheme="urn:x-s-ramp:2010:type" />
  </entry>
</feed>
```

It is also useful to examine the *backwardRelationships* feed from the example in Section 2.4.1.1, since it contains a "describedBy" relationship entry because this Source Entry is the target of that relationship which is owned by the "myServiceInstance" Artifact Entry.

Example 15 – Backward Relationships Feed

```
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:s-ramp="http://s-ramp.org/xmlns/2010/s-
ramp">
  <id>{urn:uuid:backwardRelationships:feed}</id>
  <link href="http://example.org/s-ramp/xsd/XsdDocument/
    {uuid:backwardRelationships:feed}/backwardRelationships"
    rel="self" type="application/atom+xml;type=feed" />
  <updated>2009-05-26T13:13:55.013+02:00</updated>
  <title type="text">accountingTypes.xsd : All backward relationships feed</title>
  <author>
    <name>Bellwood</name>
  </author>

  <!—For this example, this feed only contains Relationship #4 from Section 2.4.1.1
    although to be complete, it would also have contained Relationship #1. -->

  <entry>
    <id>urn:uuid:{myServiceInstance:1:describedBy}</id>
    <updated>2009-05-26T13:13:55.013+02:00</updated>
    <title type="text">
        The describedBy relationship for myServiceInstance:1 Source Entry
    </title>
    <published>2009-05-26T13:13:55.013+02:00</published>

    <link href="http://example.org/s-ramp/serviceImplementation/ServiceInstance/
        {uuid:myServiceInstance:1}/relationships/
        {uuid:describedBy:1}"
        type="application/atom+xml;type=entry" rel="self" />
```

```
<link href="http://example.org/s-ramp/serviceImplementation/ServiceInstance/
    {uuid:myServiceInstance:1}/relationships/
    {uuid:describedBy:1}”
    type="application/atom+xml;type=entry" rel="edit" />

<!--  Content element identifies this as a Relationship Entry -->
<content type="text">Relationship Entry</content>

<-- S-RAMP structured extension for Relationship Entry data -->
<s-ramp:relationshipData>
    <s-ramp:relationshipType>describedBy</s-ramp:relationshipType>
    <s-ramp:sourceId>{uuid:ServiceInstance:1}</sourceId>
    <s-ramp:targetId>{uuid:accountingTypes:1}</targetId>
</s-ramp:relationshipData>

<!— Link to relationship’s Source Entry -->
<link title="Relationship Source Entry"
    href="http://example.org/s-ramp/serviceImplementation/ServiceInstance/
        {uuid:ServiceInstance:1}”
    type="application/atom+xml;type=entry"
    rel="urn:x-s-ramp:2010:relationship:source"/>

<!— Link to relationship’s Target Entry -->
<link title="Relationship Entry"
    href="http://example.org/s-ramp/xsd/XsdDocument/
        {uuid:accountingTypes:1}”
    type="application/atom+xml;type=entry"
    rel="urn:x-s-ramp:2010:relationship:target"/>

<-- Link to corresponding describedBy Relationship Type Entry -->
<link href="http://example.org/s-ramp/serviceImplementation/ServiceInstance/
    {uuid:myServiceInstance:1}/relationshipTypes/
    {uuid:describedBy:1}”
    type="application/atom+xml;type=entry"
    rel="urn:x-s-ramp:2010:relationship:type" />

<--  Categorizations describing the Relationship Entry -->
<category term="modeled" label="Modeled S-RAMP relationship."
          scheme="urn:x-s-ramp:2010:kind" />
<category term="relationship" label="Relationship Entry type"
          scheme="urn:x-s-ramp:2010:type" />
</entry>
```

2.4.1.3 Relationship Type Entry Documents

A Relationship Type Entry document provides information about a particular Relationship Type (such as *includedXsds*). Exactly one such document exists for each Relationship Type associated with the Source Entry.

Resolving the link to a *relationshipTypes* feed will return a feed of Relationship Type Entry documents. Each such document includes a link to the *relationships* feed of all relationship instances whose Relationship Type is represented by this Relationship Type Entry.

The members of this feed are maintained by the S-RAMP server. The following behaviors are normative for S-RAMP server implementations supporting the Fine Grained View for relationships:

- A Relationship Type Entry document is automatically generated and added to the *relationshipTypes* feed of the Source Entry whenever the client adds a relationship to the *relationships* feed of the Source Entry if its Relationship Type is not already represented in the *relationshipTypes* feed.
- Relationship Type Entry documents are NOT deleted when all relationship instances having a Relationship Type value matching that of this entry are deleted from the *relationships* feed of the Source Entry. In this situation, the Relationship Type Entry represents a relationship of that Relationship Type which has no target.
- Clients can create a Relationship Type Entry and add it to the Relationship Type feed if one of that type is not already present. This is useful when the client wants a relationship of a given type which has no targets.
- Clients can delete a Relationship Type Entry from a *relationshipTypes* feed. Doing so will automatically delete all relationship instances in the *relationships* feed which share the same Relationship Type value. The link to the Relationship Type specific *relationships/{Relationship Type}* feed will also no longer appear in the Source Entry.

A Relationship Types Entry document contains the following items:

- The basic Atom elements REQUIRED for a legal entry document. The REQUIRED atom:author element value SHALL be set by the server to match the value found in the Source Entry.
- An Atom link to the applicable Relationship Targets feed corresponding to the Relationship Type associated with this Relationship Type Entry. This is the same link as provided in the Source Entry. As described in Section 2.4.1.1, the value of this link's *rel* attribute SHALL conform to this format:

  rel="urn:x-s-ramp:2010:relationships:{Relationship Type}"
- Atom content element providing a text description of the entry document.
- A structured extension element s-ramp:relationshipTypeData containing the Relationship Type value of this Relationship Type Entry (e.g., includedXsds, similarXsds, etc.).
- Atom category elements describing the particular Relationship Type Entry:
  - The entry type:
    - scheme="urn:x-s-ramp:2010:type"
    - The only valid value for the *term* attribute here is "relationshipType"
  - The kind of relationship:
    - scheme="urn:x-s-ramp:2010:kind"
    - Valid values for *term* attribute are:
      - "derived"
      - "modeled"
      - "generic"

Below is an example of a Relationship Types Feed with two Relationship Type Entry summary documents. Note that the summary and full versions of these entries are the same in S-RAMP:

Example 16 - Relationship Types Feed

```
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:s-ramp="http://s-ramp.org/xmlns/2010/s-ramp">
  <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344eddd0</id>
  <link href="http://example.org/s-ramp/xsd/XsdDocument/
    aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/relationshipTypes/1225c695-cfb8-4ebb-aaaa-80da344eddd0"
    rel="self" type="application/atom+xml;type=feed" />
  <updated>2009-05-26T13:13:55.013+02:00</updated>
```

```
<title type=“text”>accountingTypes.xsd : Relationship Types feed</title>
<author>
  <name>Bellwood</name>
</author>

<!—First Relationship Type Entry in the feed -->
<entry>
  <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344eddd1</id>
  <updated>2009-05-26T13:13:55.013+02:00</updated>
  <title type=“text”>
      Relationship Type Entry for includedXsds relationship
  </title>
  <published>2009-05-26T13:13:55.013+02:00</published>
  <link href="http://example.org/s-ramp/xsd/XsdDocument/
      aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/relationshipTypes/
      1225c695-cfb8-4ebb-aaaa-80da344eddd1”
      type="application/atom+xml;type=entry" rel="self" />

  <!--  Content element identifies this as a Relationship Type Entry -->
  <content type="text">Relationship Type Entry</content>

  <!-- S-RAMP structured extension for Relationship Type Entry data -->
  <s-ramp:relationshipTypeData>
      <s-ramp:relationshipType>includedXsds</s-ramp:relationshipType>
  </s-ramp:relationshipTypeData>

  <!— Link to relationships feed for includedXsds Relationship Type -->
  <link title="All includedXsds Type Relationship Instances"
      href="http://example.org/sramp/xsd/XsdDocument/
      aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/relationships/includedXsds"
       type="application/atom+xml;type=feed"
      rel="urn:x-s-ramp:2010:relationships:includedXsds" />

  <category term="derived" label="Derived S-RAMP Relationship"
            scheme="urn:x-s-ramp:2010:kind" />
  <category term="relationshipType" label="S-RAMP Relationship Type Entry"
      scheme="urn:x-s-ramp:2010:type” />
</entry>

<!—Second Relationship Type Entry in the feed -->
<entry>
  <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344eddd2</id>
  <updated>2009-05-26T13:13:55.013+02:00</updated>
  <title type=“text”>
      Relationship for accountingTypes.xsd Source Entry: similarXsds
  </title>
  <published>2009-05-26T13:13:55.013+02:00</published>
  <link href="http://example.org/s-ramp/xsd/XsdDocument/
      aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/relationshipTypes/
      1225c695-cfb8-4ebb-aaaa-80da344eddd2”
      type="application/atom+xml;type=entry"
      type="application/atom+xml;type=entry" rel="self" />

  <--  Generic relationships include an “edit” link: -->
```

```
    <link href="http://example.org/s-ramp/xsd/XsdDocument/
        aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/relationshipTypes/
        1225c695-cfb8-4ebb-aaaa-80da344eddd2″
        type="application/atom+xml;type=entry"
        type="application/atom+xml;type=entry" rel="edit" />

    <!--  Content element identifies this as a Relationship Type Entry -->
    <content type="text">Relationship Type Entry</content>

    <!-- S-RAMP structured extension for Relationship Type Entry data -->
    <s-ramp:relationshipTypeData>
        <s-ramp:relationshipType>similarXsds</s-ramp:relationshipType>
    </s-ramp:relationshipTypeData>

    <!— Link to relationships feed for similarXsds Relationship Type -->
    <link title="All similarXsds Type Relationships"
        href="http://example.org/sramp/xsd/XsdDocument/
            aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/relationships/similarXsds"
        type="application/atom+xml;type=feed"
        rel="urn:x-s-ramp:2010:relationships:similarXsds" />

    <category term="generic" label="Generic Relationship"
              scheme="urn:x-s-ramp:2010:kind" />
    <category term="relationshipType" label="S-RAMP Relationship Type Entry"
        scheme="urn:x-s-ramp:2010:type″ />
  </entry>
</feed>
```

2.4.1.4 Creating a Relationship Instance

The client's ability to create new relationships varies by the kind of relationship (*derived*, *modeled*, or *generic*), the artifact types permitted for the source and target of the relationship, and on the cardinality rules for the relationship.

Creating *Derived* Relationships:

Derived relationships associated with an S-RAMP Derived Model cannot be directly created by the client. They are managed by the server based upon operations performed against document resources upon which the Derived Model containing that modeled relationship is based. To create such a relationship, it is necessary to alter the document resource itself (e.g., an XSD file).

Creating *Modeled* Relationships:

Modeled relationships (i.e., Relationship Entries) can be created and deleted within the confines of the artifact types between which they are defined, and are subject to the cardinality rules defined for them:

- The Source Artifact type and the Target Artifact type MUST match the types described in the model (i.e., the SOA Model, Service Implementation Model, or a user defined model).
- When the Minimum Cardinality >= 0, relationships can be created.
- When the Maximum Cardinality < unbounded, relationships can only be created if doing so does not violate the upper limit on cardinality.

Creating *Generic* Relationships:

Generic (ad-hoc) relationships can be created at any time in any type of Artifact Entry in any of the defined models supported by S-RAMP.

As an example of creating a generic relationship, consider two Artifact Entries, conveniently called source.xsd and target.xsd. We wish to add a *similarXsds* relationship between them. Prior to doing this, performing a GET to resolve the link to the *relationships* feed in the Atom entry for source.xsd, might return an empty feed:

Example 17 - Creating Generic Relationships - Before

```
GET /s-ramp/xsd/XsdDocument/{uuid:source.xsd}/relationships HTTP/1.1
Host: example.org
```

returns this empty feed:

```
<feed xmlns="http://www.w3.org/2005/Atom"
      xmlns:s-ramp="http://s-ramp.org/xmlns/2010/s-ramp">
  <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344eccc0</id>
  <link href="http://example.org/s-ramp/xsd/XsdDocument/
       {uuid:source.xsd}/relationships"
    rel="self" type="application/atom+xml;type=feed" />
  <updated>2009-05-26T13:13:55.013+02:00</updated>
  <title type="text">source.xsd : All relationships feed</title>
  <author>
    <name>Bellwood</name>
  </author>
</feed>
```

Now to add the desired *similarXsds* relationship, the client would POST the following Atom entry document to the source.xsd entry *relationships* feed:

Example 18 - Creating Generic Relationships - Adding the Relationship

```
POST /s-ramp/xsd/XsdDocument/{uuid:source.xsd}/relationshipsTargets HTTP/1.1
Host: example.org
Content-Type: application/atom+xml;type=entry
Content-Length: nnn

<?xml version="1.0" ?>
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:s-ramp="http://s-ramp.org/xmlns/2010/s-
ramp">
    <id>{urn:uuid:relationship:1}<id/>
    <updated />
    <title />

    <!--  Content element identifies this as a Relationship Entry -->
    <content type="text">Relationship Entry</content>

    <!-- S-RAMP structured extension for Relationship Entry data -->
    <s-ramp:relationshipData>
        <s-ramp:relationshipType>similarXsds</s-ramp:relationshipType>
        <s-ramp:sourceId>{uuid:source.xsd}</sourceId>
        <s-ramp:targetId>{uuid:target.xsd</targetId>
    </s-ramp:relationshipData>
    <!-- Note that Links to the source and target are not included on the POST, but the
        server will include them in the response to the POST and on subsequent GET
```

```
        requests. -->

    <category term="generic" label="This is a user-defined s-ramp relationship."
            scheme="urn:x-s-ramp:2010:kind" />
    <category term="relationship" label="Relationship Target"
        scheme="urn:x-s-ramp:2010:type" />
</entry>
```

After the *similarXsds* relationship above has been added to the *relationships* feed, performing another GET on the Source Entry's *relationships* feed would return:

Example 19 - Creating Generic Relationships - After

```
<feed xmlns="http://www.w3.org/2005/Atom"
      xmlns:s-ramp="http://s-ramp.org/xmlns/2010/s-ramp">
  <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344eccc0</id>
  <link href="http://example.org/s-ramp/xsd/XsdDocument/
        {uuid:source.xsd}/relationships"
    rel="self" type="application/atom+xml;type=feed" />
  <updated>2009-05-26T13:13:55.013+02:00</updated>
  <title type="text">source.xsd : All relationships feed</title>
  <author>
    <name>Bellwood</name>
  </author>

  <!—First Relationship Entry in feed -->
  <entry>
    <id>{urn:uuid:relationship:1}</id>
    <updated>2009-05-26T13:13:55.013+02:00</updated>
    <title type="text">
        Relationship for source.xsd Source Entry: similarXsd
    </title>
    <published>2009-05-26T13:13:55.013+02:00</published>

    <link href="http://example.org/s-ramp/xsd/XsdDocument/
            {uuid:source.xsd}/relationships/
            {uuid:relationship:1}"
            type="application/atom+xml;type=entry" rel="self" />
    <link href="http://example.org/s-ramp/xsd/XsdDocument/
            {uuid:source.xsd}/relationships/
            {uuid:relationship:1}"
            type="application/atom+xml;type=entry" rel="edit" />

    <!--  Content element identifies this as a Relationship Entry -->
    <content type="text">Relationship Entry</content>

    <!-- S-RAMP structured extension for Relationship Entry data -->
    <s-ramp:relationshipData>
        <s-ramp:relationshipType>similarXsds</s-ramp:relationshipType>
        <s-ramp:sourceId>{uuid:source.xsd}</sourceId>
        <s-ramp:targetId>{uuid:target.xsd}</targetId>
    </s-ramp:relationshipData>
```

```
    <!—Link to relationship Source Entry -->
    <link title="Relationship Source Entry"
        href="http://example.org/s-ramp/xsd/XsdDocument/{uuid:source.xsd}”
        type="application/atom+xml;type=entry"
        rel="urn:x-s-ramp:2010:relationship:source"/>

    <!—Link to relationship entry -->
    <link title="Relationship Entry"
        href="http://example.org/s-ramp/xsd/XsdDocument/{uuid:target.xsd}”
        type="application/atom+xml;type=entry"
        rel="urn:x-s-ramp:2010:relationship:target"/>

    <-- Link to corresponding similarXsds Relationship Type Entry
        which the server creates if this is the first similarXsds type
        relationship -->
    <link href="http://example.org/s-ramp/xsd/XsdDocument/
        {uuid:source.xsd}/relationshipTypes/
        1225c695-cfb8-4ebb-aaaa-80da344eddd2”
        type="application/atom+xml;type=entry"
        rel="urn:x-s-ramp:2010:relationship:type" />

    <category term="generic" label="This is a user-defined S-RAMP relationship."
              scheme="urn:x-s-ramp:2010:kind" />
    <category term="relationship" label="Relationship Entry type"
              scheme="urn:x-s-ramp:2010:type" />
  </entry>
</feed>
```

As noted in Section 2.4.1.3, Relationship Type Entry documents are automatically created by the server and added to the *relationshipTypes* feed in response to the first creation of a relationship instance of a new Relationship Type. To create a relationship having no targets for a Relationship Type not already present in the Source Entry, it is necessary to POST a Relationship Type Entry for the new Relationship Type to the *relationshipTypes* feed. An example of adding a relationship with no targets whose Relationship Type is called “myNewRelationshipType” follows:

Example 20 - Adding a Relationship with No Targets

```
POST /s-ramp/xsd/XsdDocument/{uuid:source.xsd}/relationshipsTypes HTTP/1.1
Host: example.org
Content-Type: application/atom+xml;type=entry
Content-Length: nnn

<?xml version="1.0" ?>
<entry>
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344eddd3</id>
    <updated>2009-05-26T13:13:55.013+02:00</updated>
    <title type=“text”>
        Relationship Type Entry for myNewRelationshipType relationship
    </title>
    <author>
      <name>Bellwood</name>
    </author>

    <!--  Content element identifies this as a Relationship Type Entry -->
```

```
    <content type="text">Relationship Type Entry</content>

    <!-- S-RAMP structured extension for Relationship Type Entry data -->
    <s-ramp:relationshipTypeData>
        <s-ramp:relationshipType>myNewRelationshipType</s-ramp:relationshipType>
    </s-ramp:relationshipTypeData>

    <category term="generic" label="Generic S-RAMP Relationship"
              scheme="urn:x-s-ramp:2010:kind" />
    <category term="relationshipType" label="S-RAMP Relationship Type Entry"
        scheme="urn:x-s-ramp:2010:type" />
</entry>
```

2.4.1.5 Retrieving a Relationship Instance

To retrieve the metadata for a particular relationship, the client simply performs a GET on the URL of the desired Relationship Entry. Following the example from the previous section, this might look like:

Example 21 - Retrieving a Relationship Entry Instance

```
GET /s-ramp/xsd/XsdDocument/{uuid:source.xsd}/relationships/similarXsds
{uuid:relationshipEntry} HTTP/1.1
Host: example.org
```

which would return the same Relationship Entry document as above:

```
<entry>
    <id>{urn:uuid:source.xsd:relationship:1}</id>
    <updated>2009-05-26T13:13:55.013+02:00</updated>
    <title type="text">
        Relationship for source.xsd Source Entry: similarXsd
    </title>
    <author>
      <name>Bellwood</name>
    </author>
    <published>2009-05-26T13:13:55.013+02:00</published>
    <link href="http://example.org/s-ramp/xsd/XsdDocument/
            {uuid:source.xsd}/relationships/similarXsds/
            {uuid:relationshipEntry}"
            type="application/atom+xml;type=entry" rel="self" />
    <link href="http://example.org/s-ramp/xsd/XsdDocument/
            {uuid:source.xsd}/relationships/similarXsds/
            {uuid:relationshipEntry}"
            type="application/atom+xml;type=entry" rel="edit" />

    <!--  Content element identifies this as a Relationship Entry -->
    <content type="text">Relationship Entry</content>

    <-- S-RAMP structured extension for Relationship Entry data -->
    <s-ramp:relationshipData>
        <s-ramp:relationshipType>similarXsds</s-ramp:relationshipType>
        <s-ramp:sourceId>{uuid:source.xsd}</sourceId>
        <s-ramp:targetId>{uuid:target.xsd}</targetId>
    </s-ramp:relationshipData>
```

```
    <!—Link to relationship’s Source Entry -->
    <link title="Relationship Source Entry"
        href="http://example.org/s-ramp/xsd/XsdDocument/{uuid:source.xsd}”
        type="application/atom+xml;type=entry"
        rel="urn:x-s-ramp:2010:relationship:source"/>

    <!—Link to relationship’s Target Entry -->
    <link title="Relationship Entry"
        href="http://example.org/s-ramp/xsd/XsdDocument/{uuid:target.xsd}”
        type="application/atom+xml;type=entry"
        rel="urn:x-s-ramp:2010:relationship:target"/>

    <-- Link to corresponding similarXsds Relationship Type Entry -->
    <link href="http://example.org/s-ramp/xsd/XsdDocument/
        aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/relationshipTypes/
        1225c695-cfb8-4ebb-aaaa-80da344eccc2”
        type="application/atom+xml;type=entry"
        rel="urn:x-s-ramp:2010:relationship:type" />

    <category term="generic" label="This is a user-defined S-RAMP relationship."
              scheme="urn:x-s-ramp:2010:kind" />
    <category term="relationship" label="Relationship Entry type"
              scheme="urn:x-s-ramp:2010:type" />
</entry>
```

2.4.1.6 Editing a Relationship Instance

Editing of an existing Relationship Entry instance document is prohibited in S-RAMP. To accomplish an edit of a non-derived relationship, one first DELETEs the existing relationship, then POSTs a new relationship with the desired changes.

Similarly, editing of an existing Relationship Type Entry instance document is prohibited in S-RAMP. As previously noted, Relationship Type Entry documents are typically created and are managed by the server, but MAY also be by the client (see Section 2.4.1.3).

2.4.1.7 Deleting a Relationship

The client’s ability to delete a relationship varies by what kind of relationship it is: *derived*, *modeled* or *generic*. Requests to delete a Relationship Entry as well as a Relationship Type Entry are discussed here for each kind of relationship. In neither case is the actual Source Entry nor Target Entry referenced in a relationship instance deleted, although the applicable feeds referenced in each are in general affected.

Syntax for deleting a Relationship Entry document:

```
DELETE /{relationshipEntryURL}  HTTP/1.1
Host: example.org
```

For example, to delete a particular generic Relationship Entry document for a relationship of type “similarXsds”:

```
DELETE /s-ramp/xsd/XsdDocument/{uuid:source.xsd}/
relationships/similarXsds/{uuid:relationship:1} HTTP/1.1
Host: example.org
```

Syntax for deleting a Relationship Type Entry document:

```
DELETE /{relationshipTypeEntryURL}  HTTP/1.1
Host: example.org
```

For example, to delete a generic Relationship Type entry for “similarXsds”:

```
DELETE /s-ramp/xsd/XsdDocument/{uuid:source.xsd}/
relationshipType/{uuid:Relationship Type Entry} HTTP/1.1
Host: example.org
```

Deleting *Derived* Relationships:

Derived relationships are part of an S-RAMP Derived Model and cannot be directly deleted by the client. This applies to both Relationship Entry and Relationship Type Entry documents. These relationships are managed by the server based upon operations performed against the document resource upon which the Derived Model containing that modeled relationship is based (e.g., the A.xsd document). To delete a derived relationship, one MUST make a material change to the referenced document itself in such a way as to cause its Derived Model to no longer contain that relationship. For example, removing the include of the B.xsd document from the a A.xsd document will cause the Derived Model for A.xsd to be regenerated and no longer contain an includedXsds relationship with a target of B.xsd. The Relationship Type Entry for the includedXsds relationship will still exist on the A.xsd document artifact.

Deleting *Modeled* Relationships:

Modeled Relationships are predefined (e.g., in the UML Service Implementation Model or in a user defined model), but they can be deleted by the client within the constraints of their cardinality rules, although the behavior of the server for modeled relationships differs from other kinds of relationships:

- The server SHALL ensure that there is always a link to the applicable *relationships/{Relationship Type}* feed in the relationship’s Source Entry, even when this feed is empty.
- The server SHALL ensure that there is always a Relationship Type Entry in the *relationshipTypes* feed for each modeled Relationship Type defined for the artifact type (e.g., see the SOA Model UML in Figure 2 of the the Foundation Document of this specification).

Beyond these, the behavior for requests to delete modeled relationships is subject to these cardinality rules:

1. Modeled Relationships with Minimum Cardinality = 0
   - Requesting DELETE of a modeled Relationship Entry:
     - Always permitted.
     - Removes subject Relationship Entry instance document from the *relationships* feed and the applicable *relationships/{Relationship Type}* feed for the Source Entry.
     - Does not affect the *relationshipTypes* feed.
     - All occurrences of the deleted Relationship Entry instance document are removed from the *backwardRelationships* and *backwardRelationships/{Relationship Type}* feeds of the Target Artifact entry identified by the deleted relationship’s target.
   - Requesting DELETE of a Relationship Type Entry:
     - Removes all Relationship Entry instances having the Relationship Type in the request from the *relationships* feed and the applicable *relationships/{Relationship Type}* feed.
     - The Relationship Type Entry is NOT deleted from the *relationshipTypes* feed for the Source Entry. This indicates that the relationship still exists, but that it now has 0 targets. The server SHALL still return an HTTP return status code of 200 OK upon successful completion, because the request completed correctly within the defined behavior of S-RAMP.
     - The link to the (now empty) *relationships/{Relationship Type}* feed will remain in the Source Entry.
     - The corresponding Relationship Type Entry remains in the *relationshipTypes* feed.

- All occurrences of the deleted Relationship Entry instance documents are removed from the *backwardRelationships* and *backwardRelationships/{Relationship Type}* feeds of the Target Artifact entries identified by the deleted relationship targets.

2. Modeled Relationships with Minimum Cardinality > 0
   - DELETE of a modeled Relationship Entry:
     - Permitted, unless this operation would result in a violation of the minimum cardinality for this Relationship Type. Behavior when permitted is the same as for the Cardinality = 0 case.
   - DELETE of a modeled Relationship Type Entry:
     - Invalid operation. This would result in a violation of the minimum cardinality for relationships of this Relationship Type.

Deleting *Generic* Relationships:

Since generic relationships are created and controlled by the client, they MAY always be deleted. Details on deletion behavior follow:

- Requesting DELETE of a generic Relationship Entry:
  - Removes subject Relationship Entry instance document from the *relationships* feed and the applicable *relationships/{Relationship Type}* feed for the Source Entry.
  - Does not affect the *relationshipTypes* feed.
- Requesting DELETE of a generic Relationship Type Entry:
  - Removes all Relationship Entry instances having the Relationship Type in the request from the *relationships* feed and the applicable *relationships/{Relationship Type}* feed.
  - Removes the Relationship Type Entry from the *relationshipTypes* feed for the Source Entry.
  - The link to the applicable *relationships/{Relationship Type}* feed is removed from the Source Entry.

2.4.2 S-RAMP Properties

If supported by the S-RAMP server implementation, a Fine Grained View is also available for S-RAMP properties in order to facilitate their manipulation separately from the Atom Source Entry with which they are associated. This is particularly useful when the s-ramp:artifact structured extension element in the Coarse Grained View contains a large amount of data, since this view allows manipulation of one property at a time without having to explicitly update the Atom Source Entry itself.

In the Coarse Grained View, some of the built-in S-RAMP Artifact properties are mapped directly to existing Atom elements for the convenience of clients. These, together with the remaining built-in properties defined in the various models, as well as all user-defined properties, are available in the Fine Grained View. System defined properties are usually read-only. All user defined properties are editable.

If the Fine Grained View for Properties is supported, then the Artifact Entry document which describes the Coarse Grained View will always contain a link to the *properties* feed (see Section 2.3.2). For example:

```
<link href="http://example.org/s-ramp/xsd/XsdDocument/
        aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6a/properties"
     type="application/atom+xml;type=feed"
    rel="urn:x-s-ramp:2010:properties" />
```

2.4.2.1 Property Entry Documents

Resolving the link to a *properties* feed in an Artifact Entry will return a feed of Property Entry documents. A Property Entry document is a valid Atom entry document which contains information about a single S-RAMP property which is associated with the Artifact Entry in which the *properties* feed is found.

The following items SHALL appear in a Property Entry document, both in its summary and full entry representations:

- The basic Atom elements REQUIRED for a legal entry document. The REQUIRED atom:author element value SHALL be set by the server to match the value found in the Artifact Entry.
- Atom content text element describing the property.
- A structured extension element s-ramp:propertyData containing the name and value of the property.
- Atom category elements describing the particular Property Entry:
    - The entry type:
        - *scheme*="urn:x-s-ramp:2010:type"
        - The only valid value for the *term* attribute here is “property”
    - The kind of property:
        - *scheme*="urn:x-s-ramp:2010:kind"
        - Valid values for *term* attribute are:
            - “derived”
                - Built-in property defined in a Derived Model. Never editable.
            - “modeled”
                - Pre-defined property in the Core Model, SOA Model, Service Implementation Model or a user defined model. Editable, but server can override values for properties in the Core Model.
            - “generic”
                - Client defined (ad-hoc) property. Always editable.
                -

Consistent with properties contained in the s-ramp:artifact structured extension element of the Coarse Grained View, the *properties* feed for the Fine Grained View can only contain one unique Property Entry document instance for a given property name.

The representation of summary and full Property Entry documents SHALL be the same for all Property Entry documents. This makes it possible to retrieve all property data for the Artifact Entry in one step by resolving the link to the *properties* feed. Below is an example of a *properties* feed containing a user-defined (generic) Property Entry document instance, along with the modeled properties from the Core Model which are always present. For brevity, this feed only illustrates the *name* property among these:

Example 22 - Property Entry Feed

```
<feed xmlns=”http://www.w3.org/2005/Atom”
      xmlns:s-ramp=”http://s-ramp.org/xmlns/2010/s-ramp”>
  <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344eeee0</id>
  <link href="http://example.org/s-ramp/xsd/XsdDocument/
       {uuid:accountingTypes.xsd}/properties"
    rel="self" type="application/atom+xml;type=feed" />
  <updated>2009-05-26T13:13:55.013+02:00</updated>
  <title type=“text”>accountingTypes.xsd : Properties feed</title>
  <author>
    <name>Bellwood</name>
```

```
</author>

<!-- First Property Entry in feed -->
<entry>
  <id>{urn:uuid:property:1}</id>
  <updated>2009-05-26T13:13:55.013+02:00</updated>
  <title type=“text”>
     Property for accountingTypes.xsd
  </title>
  <published>2009-05-26T13:13:55.013+02:00</published>

  <link href="http://example.org/s-ramp/xsd/XsdDocument/
         {uuid:accountingTypes.xsd}/properties/
         {uuid:property:1}”
         type="application/atom+xml;type=entry” rel="self" />
  <link href="http://example.org/s-ramp/xsd/XsdDocument/
         {uuid: accountingTypes.xsd}/properties/
         {uuid:property:1}”
         type="application/atom+xml;type=entry” rel="edit" />

  <!--  Content element identifies this as a Property Entry -->
  <content type="text">Property Entry</content>

  <-- S-RAMP structured extension for Property Entry data -->
  <s-ramp:propertyData>
     <s-ramp:property>
        <s-ramp:propertyName>foo</s-ramp:propertyName>
        <s-ramp:propertyValue>bar</s-ramp:propertyValue>
     </s-ramp:property>
  </s-ramp:propertyData>

  <category term="generic" label="This is a user-defined S-RAMP property."
            scheme="urn:x-s-ramp:2010:kind" />
  <category term="property" label="Property Entry type"
            scheme="urn:x-s-ramp:2010:type" />
</entry>

<!-- Second Property Entry in the feed -->
<entry>
  <id>{urn:uuid:property:2}</id>
  <updated>2009-05-26T13:13:55.013+02:00</updated>
  <title type=“text”>
     Name property for accountingTypes.xsd
  </title>
  <published>2009-05-26T13:13:55.013+02:00</published>

  <link href="http://example.org/s-ramp/xsd/XsdDocument/
         {uuid:accountingTypes.xsd}/properties/
         {uuid:property:2}”
         type="application/atom+xml;type=entry” rel="self" />

  <!--  Content element identifies this as a Property Entry -->
  <content type="text">Property Entry</content>
```

```
    <-- S-RAMP structured extension for Property Entry data -->
    <s-ramp:propertyData>
        <s-ramp:property>
            <s-ramp:propertyName>name</s-ramp:propertyName>
            <s-ramp:propertyValue>accountingTypes</s-ramp:propertyValue>
        </s-ramp:property>
    </s-ramp:propertyData>

    <category term="modeled" label="This is a modeled S-RAMP property."
              scheme="urn:x-s-ramp:2010:kind" />
    <category term="property" label="Property Entry type"
              scheme="urn:x-s-ramp:2010:type" />
  </entry>
</feed>
```

2.4.2.2 Creating Properties

User-defined (generic) properties can be created by clients and associated with an Artifact Entry. To accomplish this, the client simply performs a POST of a Property Entry document to the Artifact Entry's *properties* feed.

For example, consider again our Artifact Entry, which is still conveniently called artifact.xsd. We wish to add a property called "foo" with value "bar" to this entry. To add the desired *foo* property, the client would POST the following Atom entry document to the artifact.xsd entry's *properties* feed:

Example 23 - Creating a Property - Adding the Property

```
POST /s-ramp/xsd/XsdDocument/{uuid:artifact.xsd}/properties HTTP/1.1
Host: example.org
Content-Type: application/atom+xml;type=entry
Content-Length: nnn

<?xml version="1.0" ?>
<entry xmlns="http://www.w3.org/2005/Atom"
xmlns:s-ramp="http://s-ramp.org/xmlns/2010/s-ramp">
    <id>{urn:uuid:property:1}<id/>
    <updated />
    <title />
    <author>
        <name>Bellwood</name>
    </author>

    <!--  Content element identifies this as a Property Entry -->
    <content type="text">Property Entry</content>

    <-- S-RAMP structured extension for Property Entry data -->
    <s-ramp:propertyData>
        <s-ramp:property>
            <s-ramp:propertyName>foo</s-ramp:propertyName>
            <s-ramp:propertyValue>bar</s-ramp:propertyValue>
        </s-ramp:property>
    </s-ramp:propertyData>

    <category term="generic" label="This is a user-defined property."
            scheme="urn:x-s-ramp:2010:kind" />
```

```
    <category term="property" label="Property entry"
        scheme="urn:x-s-ramp:2010:type" />
</entry>
```

After the *foo* property above has been added to the *properties* feed, performing a GET on the Artifact Entry's *properties* feed would return:

Example 24 - Creating a Property - After

```
<feed xmlns="http://www.w3.org/2005/Atom"
      xmlns:s-ramp="http://s-ramp.org/xmlns/2010/s-ramp">
  <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344eeee0</id>
  <link href="http://example.org/s-ramp/xsd/XsdDocument/
        {uuid:artifact.xsd}/properties"
    rel="self" type="application/atom+xml;type=feed" />
  <updated>2009-05-26T13:13:55.013+02:00</updated>
  <title type="text">source.xsd : Feed of all properties</title>
  <author>
    <name>Bellwood</name>
  </author>

  <!—First Property Entry in feed -->
  <entry>
    <id>{urn:uuid:property:1}</id>
    <updated>2009-05-26T13:13:55.013+02:00</updated>
    <title type="text">
        foo property for artifact.xsd Entry
    </title>
    <published>2009-05-26T13:13:55.013+02:00</published>

    <link href="http://example.org/s-ramp/xsd/XsdDocument/
           {uuid:artifact.xsd}/properties/
           {uuid:property:1}"
           type="application/atom+xml;type=entry" rel="self" />
    <link href="http://example.org/s-ramp/xsd/XsdDocument/
           {uuid:artifact.xsd}/properties/
           {uuid:property:1}"
           type="application/atom+xml;type=entry" rel="edit" />

    <!--  Content element identifies this as a Property Entry -->
    <content type="text">Property Entry</content>

    <-- S-RAMP structured extension for Property Entry data -->
    <s-ramp:propertyData>
        <s-ramp:property>
            <s-ramp:propertyName>foo</s-ramp:propertyName>
            <s-ramp:propertyValue>bar</s-ramp:propertyValue>
        </s-ramp:property>
    </s-ramp:propertyData>

    <category term="generic" label="This is a user-defined property."
           scheme="urn:x-s-ramp:2010:kind" />
    <category term="property" label="Property entry"
```

```
      scheme="urn:x-s-ramp:2010:type" />
  </entry>
  …
</feed>
```

2.4.2.3 Retrieving Properties

To retrieve the metadata for a particular property, the client simply performs a GET on the URL of the desired Property Entry. Following the example from the previous section, this might look like:

Example 25 - Retrieving a Property Entry Document

```
GET /s-ramp/xsd/XsdDocument/{uuid:artifact.xsd}/properties/
{uuid:property:1} HTTP/1.1
Host: example.org
```

would return the same Property Entry document as above:

```
  <entry>
    <id>{urn:uuid:property:1}</id>
    <updated>2009-05-26T13:13:55.013+02:00</updated>
    <title type="text">
        foo property for artifact.xsd Entry
    </title>
    <published>2009-05-26T13:13:55.013+02:00</published>

    <link href="http://example.org/s-ramp/xsd/XsdDocument/
           {uuid:artifact.xsd}/properties/
           {uuid:property:1}"
           type="application/atom+xml;type=entry" rel="self" />
    <link href="http://example.org/s-ramp/xsd/XsdDocument/
           {uuid:artifact.xsd}/properties/
           {uuid:property:1}"
           type="application/atom+xml;type=entry" rel="edit" />

    <!--  Content element identifies this as a Property Entry -->
    <content type="text">Property Entry</content>

    <-- S-RAMP structured extension for Property Entry data -->
    <s-ramp:propertyData>
        <s-ramp:property>
            <s-ramp:propertyName>foo</s-ramp:propertyName>
            <s-ramp:propertyValue>bar</s-ramp:propertyValue>
        </s-ramp:property>
    </s-ramp:propertyData>

    <category term="generic" label="This is a user-defined property."
           scheme="urn:x-s-ramp:2010:kind" />
    <category term="property" label="Property entry"
        scheme="urn:x-s-ramp:2010:type" />
  </entry>
```

2.4.2.4 Editing Properties

Editing of an existing Property Instance document is limited to altering the property value. The property name is always read only and cannot be changed by editing. Requests to alter the property name of an existing Property Entry document will return HTTP error “403” Forbidden.

To edit the property value in a Property Entry document, the client performs an HTTP PUT of the complete Property Entry document with the changed value, to the member resource URI of the Property Entry document. The PUT operation will replace the property value with whatever value is specified here. An example which replaces the property value in the previous example with a new value is illustrated below:

Example 26 - Editing a Property Entry Document

```
PUT /s-ramp/xsd/XsdDocument/{uuid:artiact.xsd}/properties HTTP/1.1
Host: example.org
Content-Type: application/atom+xml;type=entry
Content-Length: nnn

<?xml version="1.0" ?>
<entry>
    <id>{urn:uuid:property:1}</id>
    <updated>2009-05-26T13:13:55.013+02:00</updated>
    <title type=“text”>
        foo property for artifact.xsd Entry
    </title>
    <published>2009-05-26T13:13:55.013+02:00</published>

    <link href="http://example.org/s-ramp/xsd/XsdDocument/
            {uuid:artifact.xsd}/properties/
            {uuid:property:1}”
            type=”application/atom+xml;type=entry rel="self" />
    <link href="http://example.org/s-ramp/xsd/XsdDocument/
            {uuid:artifact.xsd}/properties/
            {uuid:property:1}”
            type=”application/atom+xml;type=entry” rel="edit" />
    \<!--  Content element identifies this as a Property Entry -->
    <content type="text">Property Entry</content>

    <-- S-RAMP structured extension for Property Entry data -->
    <s-ramp:propertyData>
        <s-ramp:property>
            <s-ramp:propertyName>foo</s-ramp:propertyName>
            <s-ramp:propertyValue>bar1</s-ramp:propertyValue>
        </s-ramp:property>
    </s-ramp:propertyData>

    <category term="generic" label="This is a user-defined property."
            scheme="urn:x-s-ramp:2010:kind" />
    <category term="property" label="Property entry"
        scheme="urn:x-s-ramp:2010:type” />
  </entry>
```

2.4.2.5 Deleting Properties

To delete a *generic* (ad-hoc) property and remove it from the *properties* feed associated with an Artifact Entry, a client simply performs a DELETE against the URL of the desired Property Entry. Continuing with the *generic* property example from the previous sections this might look like:

```
DELETE /s-ramp/xsd/XsdDocument/{uuid:artifact.xsd}/properties/
{uuid:property:1} HTTP/1.1
Host: example.org
```

2.4.3 S-RAMP Classifications

S-RAMP classifications are another important class of metadata which describe an S-RAMP Artifact. The S-RAMP schema models a classification with a URL for the value of the s-ramp:classifiedBy element. Individual classification values within an S-RAMP Artifact Entry SHALL be unique. The same classification URL value can only be represented once in an s-ramp:artifact structured extension in the Artifact Entry.

If supported by the S-RAMP server implementation, a Fine Grained View is also available for S-RAMP classifications in order to facilitate their manipulation separately from the Artifact Entry with which they are associated. This is particularly useful when the s-ramp:artifact element contains a large amount of data, since this view allows manipulation of one classification at a time without having to explicitly update the Artifact Entry itself.

If the Fine Grained View for Classifications is supported, then the Artifact Entry document which describes the Coarse Grained View will always contain a link to the *classifications* feed (see Section 2.3.2). For example:

```
<link href="http://{host}/s-ramp/{uuid:Artifact-Entry}/classifications"
      type="application/atom+xml;type=feed"
     rel="urn:x-s-ramp:2010:classification" />
```

2.4.3.1 The Classification Entry Document

Resolving the link to a *classifications* feed in an Artifact Entry will return a feed of Classification Entry documents. A Classifications Entry document is a valid Atom entry document which contains information about a single S-RAMP classification which is associated with the Artifact Entry in which this *classifications* feed is found.

The following items SHALL appear in a Classifications Entry document, both in its summary and full entry representations:

- The basic Atom elements REQUIRED for a legal entry document. The REQUIRED atom:author element value SHALL be set by the server to match the value found in the Artifact Entry.
- Atom content text element describing the classification entry.
- A structured extension element s-ramp:classificationData containing the URL of the OWL classification value.
- Atom category element describing the particular Classification Entry:
    - The entry type:
        - scheme="urn:x-s-ramp:2010:type"
        - The only valid value for the *term* attribute here is “classification”

As with the Coarse Grained View, the *classifications* feed for the Fine Grained View can only contain one unique Classification Entry document instance for a given classification URL value. These values correspond one-to-one with the s-ramp:classifiedBy element values in the s-ramp:artifact structured extension found in a full Artifact Entry document.

The representation of both summary and full Classification Entry documents SHALL be the same for all Classification Entry documents. This makes it possible to retrieve all classification data for the Artifact Entry in one step by resolving the link to the *classifications* feed. Below is an example of a *classifications* feed containing a Classification Entry document instance:

Example 27 - Classification Entry Feed

```
<feed xmlns="http://www.w3.org/2005/Atom"
      xmlns:s-ramp="http://s-ramp.org/xmlns/2010/s-ramp">
  <id>urn:uuid:aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaff0</id>
  <link href="http://example.org/s-ramp/xsd/XsdDocument/
       {uuid:accountingTypes.xsd}/classifications"
    rel="self" type="application/atom+xml;type=feed" />
  <updated>2009-05-26T13:13:55.013+02:00</updated>
  <title type="text">accountingTypes.xsd : Classifications feed</title>
  <author>
    <name>Bellwood</name>
  </author>

  <!—First Classification Entry in feed -->
  <entry>
    <id>{urn:uuid:classification:1}</id>
    <updated>2009-05-26T13:13:55.013+02:00</updated>
    <title type="text">
       Classification for accountingTypes.xsd
    </title>
    <published>2009-05-26T13:13:55.013+02:00</published>

    <link href="http://example.org/s-ramp/xsd/XsdDocument/
           {uuid:accountingTypes.xsd}/classifications/
           {uuid:classification:1}"
           type="application/atom+xml;type=entry" rel="self" />
    <link href="http://example.org/s-ramp/xsd/XsdDocument/
           {uuid: accountingTypes.xsd}/classifications/
           {uuid:classification:1}"
           type="application/atom+xml;type=entry" rel="edit" />

    <!--  Content element identifies this as a Classification Entry -->
    <content type="text">Classification Entry</content>

    <-- S-RAMP structured extension for Classification Entry data -->
    <s-ramp:classificationData>
       <s-ramp:classifiedBy>
          http://example.org/ontologies/accounting.owl/accounts#checking
       </s-ramp:classifiedBy>
    </s-ramp:classificationData>

    <category term="classification" label="Classification Entry type"
              scheme="urn:x-s-ramp:2010:type" />
  </entry>
</feed>
```

2.4.3.2 Creating Classifications

User-defined (generic) classifications can be created by clients and associated with an Artifact Entry. To accomplish this, the client simply performs a POST of a Classification Entry document to the Artifact Entry's *classifications* feed.

For example, consider our Artifact Entry called artifact.xsd again. We wish to add a classification instance value denoting a savings account using the accounting.owl ontology. Prior to doing this, performing a GET to resolve the link to the *classifications* feed in the Atom entry for artifact.xsd, might return an empty feed:

Example 28 - Creating a Classification - Before

```
GET /s-ramp/xsd/XsdDocument/{uuid:artiact.xsd}/classifications HTTP/1.1
Host: example.org
```

returns this empty feed:

```
<feed xmlns="http://www.w3.org/2005/Atom"
      xmlns:s-ramp="http://s-ramp.org/xmlns/2010/s-ramp">
  <id>urn:uuid:aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaff0</id>
  <link href="http://example.org/s-ramp/xsd/XsdDocument/
       {uuid:artifact.xsd}/classifications"
    rel="self" type="application/atom+xml;type=feed" />
  <updated>2009-05-26T13:13:55.013+02:00</updated>
  <title type="text">classifications.xsd : Feed of all  classifications</title>
  <author>
    <name>Bellwood</name>
  </author>
</feed>
```

Now to add the desired savings account classification, the client would POST the following Classification Entry document to the artifact.xsd entry's *classifications* feed:

Example 29 - Creating a Classification - Adding the Classification Entry

```
POST /s-ramp/xsd/XsdDocument/{uuid:artifact.xsd}/classifications HTTP/1.1
Host: example.org
Content-Type: application/atom+xml;type=entry
Content-Length: nnn

<?xml version="1.0" ?>
<entry xmlns="http://www.w3.org/2005/Atom"
       xmlns:s-ramp="http://s-ramp.org/xmlns/2010/s-ramp"
       xmlns:s-rampatom=" http://s-ramp.org/xmlns/2010/s-ramp/atombinding">
    <id>{urn:uuid:classification:1}<id/>
    <updated />
    <title />
    <author>
        <name>Bellwood</name>
    </author>

    <!--  Content element identifies this as a Classification Entry -->
    <content type="text">Classification Entry</content>

    <-- S-RAMP structured extension for Classification Entry data -->
```

```
    <s-ramp:classificationData>
        <s-ramp:classifiedBy>
            http://example.org/ontologies/accounting.owl/accounts#savings
        </s-ramp:classifiedBy>
    </s-ramp:classificationData>

    <category term="classification" label="Classification entry"
        scheme="urn:x-s-ramp:2010:type" />
</entry>
```

After the *savings account* classification above has been added to the *classifications* feed, performing another GET on the Artifact Entry's *classifications* feed would return:

Example 30 - Creating a Classification - After

```
<feed xmlns="http://www.w3.org/2005/Atom"
      xmlns:s-ramp="http://s-ramp.org/xmlns/2010/s-ramp">
  <id>urn:uuid:aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaff0</id>
  <link href="http://example.org/s-ramp/xsd/XsdDocument/
        {uuid:artifact.xsd}/classifications"
    rel="self" type="application/atom+xml;type=entry" />
  <updated>2009-05-26T13:13:55.013+02:00</updated>
  <title type="text">source.xsd : Feed of all classifications</title>
  <author>
    <name>Bellwood</name>
  </author>

  <!-- First Classification Entry in feed -->
  <entry>
    <id>{urn:uuid:classification:1}</id>
    <updated>2009-05-26T13:13:55.013+02:00</updated>
    <title type="text">
        Account savings classification for artifact.xsd entry
    </title>
    <published>2009-05-26T13:13:55.013+02:00</published>

    <link href="http://example.org/s-ramp/xsd/XsdDocument/
            {uuid:artifact.xsd}/classifications/
            {uuid:classification:1}"
            type="application/xml;type=entry" rel="self" />
    <link href="http://example.org/s-ramp/xsd/XsdDocument/
            {uuid:artifact.xsd}/classifications/
            {uuid:classification:1}"
            type="application/xml;type=entry" rel="edit" />

    <!--  Content element identifies this as a Classification Entry -->
    <content type="text">Classification Entry</content>

    <-- S-RAMP structured extension for Relationship Entry data -->
    <s-ramp:classificationData>
        <s-ramp:classifiedBy>
            http://example.org/ontologies/accounting.owl/accounts#savings
        </s-ramp:classifiedBy>
```

```
    </s-ramp:classificationData>

    <category term="classification" label="Classification entry"
        scheme="urn:x-s-ramp:2010:type" />
  </entry>
</feed>
```

2.4.3.3 Retrieving Classifications

To retrieve the metadata for a particular classification value, the client simply performs a GET on the URL of the desired Classification Entry. Following the example from the previous section, this might look like:

Example 31 - Retrieving a Classification Entry Document

```
GET /s-ramp/xsd/XsdDocument/{uuid:artifact.xsd}/classifications/
{uuid:classification:1} HTTP/1.1
Host: example.org
```

would return the same Classification Entry document as above:

```
 <entry>
    <id>{urn:uuid:classification:1}</id>
    <updated>2009-05-26T13:13:55.013+02:00</updated>
    <title type="text">
        Account savings classification for artifact.xsd entry
    </title>
    <published>2009-05-26T13:13:55.013+02:00</published>

    <link href="http://example.org/s-ramp/xsd/XsdDocument/
           {uuid:artifact.xsd}/classifications/
           {uuid:classification:1}"
           type="application/xml;type=entry" rel="self" />
    <link href="http://example.org/s-ramp/xsd/XsdDocument/
           {uuid:artifact.xsd}/classifications/
           {uuid:classification:1}"
           type="application/xml;type=entry" rel="edit" />

    <!--  Content element identifies this as a Classification Entry -->
    <content type="text">Classification Entry</content>

    <-- S-RAMP structured extension for Relationship Entry data -->
    <s-ramp:classificationData>
        <s-ramp:classifiedBy>
            http://example.org/ontologies/accounting.owl/accounts#savings
        </s-ramp:classifiedBy>
    </s-ramp:classificationData>

    <category term="classification" label="Classification entry"
        scheme="urn:x-s-ramp:2010:type" />
  </entry>
```

2.4.3.4 Editing Classifications

Editing of an existing Classification Entry document instance is prohibited in S-RAMP. To accomplish an edit of a classification using the Fine Grained View, the client first performs a DELETE of the existing classification, then a POST of a new classification with the desired changes.

2.4.3.5 Deleting Classifications

To delete a *generic* classification and remove it from the *classifications* feed associated with an Artifact Entry, a client simply performs a DELETE against the URL of the desired Classification Entry. Continuing with the classification example from the previous sections this might look like:

```
DELETE /s-ramp/xsd/XsdDocument/{uuid:artifact.xsd}/classifications/
{uuid:classification:1} HTTP/1.1
Host: example.org
```

3. S-RAMP Query Using Atom Binding

3.1 Searching Repository Artifacts

S-RAMP supports a rich query capability, which is based upon the use of flexible XPath 2 based filter arguments. Refer to the *SOA Repository Artifact Model and Protocol Specification – Foundation* document, Section 5 for details on how to form S-RAMP query predicates. This document only describes the Atom specific syntax needed for query using the Atom Binding.

A successful query using the Atom Binding will return a feed. Feeds contain summary Atom entry documents which cannot be assumed to be complete. To retrieve a full entry, it is necessary to perform a subsequent GET on the desired entry.

S-RAMP supports execution of queries via an inline (ad-hoc) syntax, as well as through the use of Stored Queries which have been stored in the repository. Each is discussed in the following sections.

3.2 Inline Queries

Ad-hoc queries can be performed in one of two ways:

1. Using HTTP GET where the query arguments are included in the URL
2. Using HTTP POST where the query arguments are the content being posted

To perform an ad-hoc query using HTTP GET use the following syntax:

```
GET /{query path}?query={predicate-filter-string}&{label}={query parameter}…  HTTP/1.1
Host: example.org
```

To perform an ad-hoc query using HTTP POST use the following syntax:

```
POST /s-ramp
Content-Type: multipart/form-data; boundary=AaB03x

--AaB03x
Content-Disposition: form-data; name=”query”

{query predicate}

--AaB03x
Content-Disposition: form-data; name={label}

{query parameter}
--AaB03x

--AaB03x--
```

The only legal values for {label} = {query parameter} are defined here. Sets of these can be repeated an arbitrary number of times:

- propertyName = {property name value}. This allows specifying property name(s) whose values SHALL be included in applicable Entry documents in the results feed.

The {query predicate} syntax is defined in the *Foundation* document. Example 32 - Ad-hoc Queries below illustrates both approaches for specifying an optional namespace and a propertyName to be returned in the results of the query (note that uses of the HTML reserved character “:” in these examples would need to be URL encoded as %3A):

Example 32 - Ad-hoc Queries

Query using HTTP GET with arguments contained in the URL:

```
GET /s-ramp?query=serviceImplementation/ServiceInstance[@reliability!=”high”]
&propertyName=”reliability”&xmlns:acme=”http://acme.org/s-ramp/custom” HTTP/1.1
Host: example.org
```

Query using HTTP POST with arguments as content:

```
POST /s-ramp
Content-Type: multipart/form-data; boundary=AaB03x

--AaB03x
Content-Disposition: form-data; name=”query”

query=serviceImplementation/ServiceInstance[reliability!
=”high”]&xmlns:acme=http://acme.org/s-ramp/custom

--AaB03x
Content-Disposition: form-data; name=”propertyName”

reliability

--AaB03x--
```

The response from either form of ad-hoc query is an Atom *feed* of summary entry documents which match the criteria of the query. If there are no matches, the feed will be empty. If one or more optional propertyName values is included as a {query parameter}, this will cause each entry document returned to include an *s-ramp:artifact* section containing the specific properties listed. The “s-ramp:artifact” section included SHALL NOT be considered a complete representation of the entry. The propertyName parameter option is a convenience to allow clients to recover all specifically requested properties in the feed of entries. This MAY avoid the need to perform a subsequent GET on individual entries if the full entry is not needed. Outside of explicit use of this parameter, S-RAMP does not prescribe which portions of the Artifact Entry are included in the summary entries returned. This will vary by implementation. Clients MUST perform a GET operation on the member resource URI in order to guarantee that they have complete information for a particular artifact.

Example 33 - Ad-hoc Query Response

```
<feed xmlns=”http://www.w3.org/2005/Atom”
      xmlns:s-ramp=”http://s-ramp.org/xmlns/2010/s-ramp”>
  <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344ezzz0</id>
  <link href=”http://example.org/s-ramp?query=s-
ramp/serviceImplementation/ServiceInstance[@acme:reliability=’high’]
        &xmlns:acme=’http://acme.org/s-ramp/custom’”
        rel="self" type="application/atom+xml;type=feed" />
  <updated>2009-05-26T13:13:55.013+02:00</updated>
  <title type=“text”>Query Response</title>
  <author>
    <name>Bellwood</name>
  </author>

  <!—First Matching Entry in feed -->
  <entry xmlns=”http://www.w3.org/2005/Atom”
       xmlns:s-ramp=”http://s-ramp.org/xmlns/2010/s-ramp”>
    <id>urn:uuid:aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaabb</id>
```

```
    <updated>2009-05-26T13:13:55.013+02:00</updated>
    <title type="text">myServiceInstance</title>
    <published>2009-05-26T13:13:55.013+02:00</published>
    <author>
        <name>Bellwood</name>
    </author>
    <summary type="text">My Service Instance document</summary>
    <content type="application/xml"
        src="http://example.org/s-ramp/serviceImplementation/ServiceInstance/
        aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaa6z"/>
    </content>
    <content
        type="application/xml"
        src="http://example.org/s-ramp/{path}/{artifact id}"/>

    <link type="application/atom+xml;type=entry" rel="self"
        href="http://example.org/s-ramp/{uuid:myServiceInstance}" />
    <link type="application/atom+xml;type=entry" rel="edit"
        href="http://example.org/s-ramp/{uuid:myServiceInstance}" />

    <!– Client defined classifications: -->
    <category term="http://example.org/ontologies/accounting.owl/accounts"
            label="User defined classification"
            scheme="urn:x-s-ramp:user-defined-classification" />

    <!– S-RAMP defined categorizations identifying class of data represented by
        this entry -->
    <category term="ServiceInstance" label="Service Instance"
        scheme="urn:x-s-ramp:2010:type" />
    </entry>
</feed>
```

3.3 Stored Queries

Query filters can be stored as artifacts in S-RAMP using the StoredQuery element described in the *Foundation* document, Section 5.6. Clients can request the server to execute a Stored Query, whose results are then made available at a particular URL. Stored Queries can be prepared for general use, or can be created by the client. Stored Queries are specified to an S-RAMP server using a Stored Query Entry document. Note that since the StoredQuery does not derive from baseArtifactType, it is much simpler. The name of the Stored Query SHALL be a unique client provided string.

3.3.1 Stored Query Entry Documents

The following items SHALL appear in a Stored Query Entry document, both in its summary and full entry representations:

- The basic Atom elements REQUIRED for a legal entry document.
- Atom content text element indicating that the document represents a Stored Query Entry.
- S-RAMP structured extension (s-ramp:StoredQueryData element) as described in Appendix C, which contains the following items.
    - A unique client provided name for the query (queryName)
    - A query expression (queryString)

    - An optional list of property names (propertyName values) which indicate to the server the properties whose values SHALL be included in the feed of Stored Query Entries made available at the results URL, to whatever extent those properties are present in each entry returned in the feed as a result of executing the query.

- Link to the ***results*** collection of Artifact Entry documents returned as a result of executing the Stored Query. Resolving this link will return a feed of summary Artifact Entry documents which matched the criteria, including parameter substitution, of the queryString in the Stored Query. This link is not included on publication of the Stored Query Entry. The server sets it during processing of the POST. This feed link SHALL have a *rel* attribute of the following form:

  `rel="urn:{host}:{version}:query/{queryName}/results"`

  For example:

  ```
  <link title="Query Results for findImpleByVersion Stored Query"
    href="http://example.org/s-ramp/query/findImplsByVersion/results"
    type="application/atom+xml;type=feed"
    rel="urn:x-s-ramp:2010:query:results" />
  ```

- Atom category element identifying the document as a Stored Query Entry:
    - The entry type:
        - *scheme*="urn:x-s-ramp:2010:type"
        - The only valid value for the *term* attribute here is “query”

Stored Query Entry documents MAY also be used as templates, allowing simple substitution of client specified parameter values during execution. The syntax for parameter substitution follows the XPath2 style to represent a variable within the query filter:

```
${var-name}
```

A value for the var-name can then be specified as part of the query invocation. Default values are not supported.

All Stored Queries accessible to a given client are stored as members of the s-ramp *query* collection:

`{host}/s-ramp/query`

Resolving this URL using HTTP GET will return a feed of all Stored Query Entry documents available to the client. An individual Stored Query Entry document follows from that root:

```
{host}/s-ramp/query/{queryName}
```

Example 34 below illustrates a Stored Query Entry document which supports parameter substitution:

Example 34 - Stored Query Entry Document

```
<entry>
  <id>{urn:uuid:findImplsByVersion}</id>
  <updated>2009-05-26T13:13:55.013+02:00</updated>
  <title type=“text”>
     Stored Query to retrieve ServiceInstance documents by version
  </title>
  <published>2009-05-26T13:13:55.013+02:00</published>

  <link href=”http://example.org/s-ramp/query/findImplsByVersion"
     type="application/atom+xml;type=entry" rel="self" />
  <link href="http://example.org/s-ramp/query/findImplsByVersion"
```

```
        type="application/atom+xml;type=entry" rel="edit" />

    <-- When returned by the server after publication via POST, server includes
        a link to the results collection for this Store Query -->
    <link href="http://example.org/s-ramp/query/findImplsByVersion/results"
        type="application/atom+xml;type=feed" rel="urn:x-s-ramp:2010:query:results" />

    <!--  Content element identifies this as a Stored Query Entry -->
    <content type="text">Stored Query Entry</content>

    <-- S-RAMP structured extension for Stored Query Data -->
    <s-ramp:StoredQueryData>
        <s-ramp:queryName>FindImplsByVersion</s-ramp:StoredQueryName>
        <s-ramp:queryString>
            s-ramp/serviceImplementation/ServiceInstance[@version >= ${MINVERSION}]>
        </s-ramp:queryString>
        <s-ramp:propertyName>version</s-ramp:propertyName>
        <s-ramp:propertyName>importantPropertyToKnow</s-ramp:propertyName>
    </s-ramp:StoredQueryData>

    <category term="query" label="Stored Query entry"
        scheme="urn:x-s-ramp:2010:type" />
</entry>
```

Stored Query Entry documents are managed in the same way as all other Artifact Entry documents are in the Atom Binding. HTTP POST, PUT, GET and DELETE.

4. Security

The S-RAMP Specification does not attempt to define a security model for products which implement it. For the Atom Binding, the only security requirement is that at a minimum, client and server implementations MUST be capable of being configured to use HTTP Basic Authentication in conjunction with a connection made with TLS.

A S-RAMP Atom Service Document

The Atom Service Document for S-RAMP defines a set of workspaces. Each of these workspaces contains a collection which can be published in a S-RAMP compliant repository:

- Core Model Workspace
- WSDL Model Workspace
- Service Implementation Model Workspace
- SOAP WSDL Model Workspace
- SOA Model Workspace
- XSD Model Workspace
- Policy Model Workspace
- Query Model Workspace

All collections are classified according to the type of entry documents which they can contain.

```
<?xml version="1.0" encoding="UTF-8"?>
<service xmlns:atom="http://www.w3.org/2005/Atom" xmlns="http://www.w3.org/2007/app">
    <workspace>
        <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Query Model
        </atom:title>
        <collection href="http://example.org/s-ramp/query">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Query Model
                Objects</atom:title>
            <accept>application/atom+xml; type=entry
            </accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="query"
                    label="Query" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
    </workspace>
    <workspace>
        <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Core Model
        </atom:title>
        <collection
            href="http://example.org/s-ramp/core/XmlDocument">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">XML Documents
            </atom:title>
            <accept>application/xml</accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="XmlDocument"
                    label="XML Document" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
```

```
        </collection>
        <collection
            href="http://example.org/s-ramp/core/document">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Documents
            </atom:title>
            <accept>application/octet-stream</accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="Document"
                    label="Document" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection href="http://example.org/s-ramp/core">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Core Model
Objects
            </atom:title>
            <accept>application/zip</accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="Document"
                    label="Document" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="XmlDocument"
                    label="XML Document" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
    </workspace>
    <workspace>
        <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">WSDL Model
        </atom:title>
        <collection
            href="http://example.org/s-ramp/wsdl/BindingOperationOutput">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Binding
                Operation Outputs</atom:title>
            <accept></accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="BindingOperationOutput"
                    label="Binding Operation Output"
xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/wsdl/BindingOperation">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Binding
                Operations</atom:title>
            <accept></accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="BindingOperation"
                    label="Binding Operation" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/wsdl/WsdlDocument">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">WSDL Documents
            </atom:title>
            <accept>application/xml</accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
```

```
        <category scheme="urn:x-s-ramp:2010:type" term="WsdlDocument"
            label="WSDL Document" xmlns="http://www.w3.org/2005/Atom"></category>
    </categories>
</collection>
<collection
    href="http://example.org/s-ramp/wsdl/Binding">
    <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Bindings
    </atom:title>
    <accept></accept>
    <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
        <category scheme="urn:x-s-ramp:2010:type" term="Binding"
            label="Binding" xmlns="http://www.w3.org/2005/Atom"></category>
    </categories>
</collection>
<collection
    href="http://example.org/s-ramp/wsdl/OperationInput">
    <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Operation
        Inputs</atom:title>
    <accept></accept>
    <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
        <category scheme="urn:x-s-ramp:2010:type" term="OperationInput"
            label="Operation Input" xmlns="http://www.w3.org/2005/Atom"></category>
    </categories>
</collection>
<collection
    href="http://example.org/s-ramp/wsdl/Message">
    <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Messages
    </atom:title>
    <accept></accept>
    <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
        <category scheme="urn:x-s-ramp:2010:type" term="Message"
            label="Message" xmlns="http://www.w3.org/2005/Atom"></category>
    </categories>
</collection>
<collection
    href="http://example.org/s-ramp/wsdl/Fault">
    <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Faults
    </atom:title>
    <accept></accept>
    <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
        <category scheme="urn:x-s-ramp:2010:type" term="Fault"
            label="Fault" xmlns="http://www.w3.org/2005/Atom"></category>
    </categories>
</collection>
<collection
    href="http://example.org/s-ramp/wsdl/Operation">
    <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Operations
    </atom:title>
    <accept></accept>
    <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
        <category scheme="urn:x-s-ramp:2010:type" term="Operation"
            label="Operation" xmlns="http://www.w3.org/2005/Atom"></category>
    </categories>
</collection>
```

```
        <collection href="http://example.org/s-ramp/wsdl">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">WSDL Model
Objects
            </atom:title>
            <accept>application/zip</accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="WsdlDocument"
                    label="WSDL Document" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="WsdlDerivedArtifactType"
                    label="WSDL Derived Artifact"
xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="NamedWsdlDerivedArtifactType"
                    label="Named WSDL Derived Artifact"
xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="Service"
                    label="Service" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="Port"
                    label="Port" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="WsdlExtension"
                    label="WSDL Extension" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="Part"
                    label="Part" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="Message"
                    label="Message" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="Fault"
                    label="Fault" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="PortType"
                    label="Port Type" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="Operation"
                    label="Operation" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="OperationInput"
                    label="Operation Input" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="OperationOutput"
                    label="Operation Output" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="Binding"
                    label="Binding" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="BindingOperation"
                    label="Binding Operation" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="BindingOperationInput"
                    label="Binding Operation Input"
xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="BindingOperationOutput"
                    label="Binding Operation Output"
xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="BindingOperationFault"
                    label="Binding Operation Fault"
xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/wsdl/WsdlExtension">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">WSDL Extensions
            </atom:title>
            <accept></accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
```

```
                <category scheme="urn:x-s-ramp:2010:type" term="WsdlExtension"
                    label="WSDL Extension" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/wsdl/WsdlDerivedArtifactType">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">WSDL Derived
                Artifacts</atom:title>
            <accept></accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="WsdlDerivedArtifactType"
                    label="WSDL Derived Artifact"
xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/wsdl/OperationOutput">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Operation
                Outputs</atom:title>
            <accept></accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="OperationOutput"
                    label="Operation Output" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/wsdl/NamedWSDLDerivedArtifactType">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Named WSDL
Derived
                Artifacts</atom:title>
            <accept></accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="NamedWsdlDerivedArtifactType"
                    label="Named WSDL Derived Artifact"
xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/wsdl/Port">
            <atom:title type="text"
xmlns:atom="http://www.w3.org/2005/Atom">Ports</atom:title>
            <accept></accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="Port"
                    label="Port" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/wsdl/Part">
            <atom:title type="text"
xmlns:atom="http://www.w3.org/2005/Atom">Parts</atom:title>
            <accept></accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="Part"
                    label="Part" xmlns="http://www.w3.org/2005/Atom"></category>
```

```
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/wsdl/PortType">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Port Types
            </atom:title>
            <accept></accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="PortType"
                    label="Port Type" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/wsdl/BindingOperationFault">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Binding
                Operation Faults</atom:title>
            <accept></accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="BindingOperationFault"
                    label="Binding Operation Fault"
xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/wsdl/BindingOperationInput">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Binding
                Operation Inputs</atom:title>
            <accept></accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="BindingOperationInput"
                    label="Binding Operation Input"
xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/wsdl/Service">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Service
            </atom:title>
            <accept>application/atom+xml; type=entry
            </accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="Service"
                    label="Service" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
    </workspace>
    <workspace>
        <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Service
            Implementation Model</atom:title>
        <collection
            href="http://example.org/s-ramp/serviceImplementation/ServiceOperation">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Service
                Operations</atom:title>
            <accept>application/atom+xml; type=entry
```

```
        </accept>
        <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
            <category scheme="urn:x-s-ramp:2010:type" term="ServiceOperation"
                label="Service Operation" xmlns="http://www.w3.org/2005/Atom"></category>
        </categories>
    </collection>
    <collection
        href="http://example.org/s-ramp/serviceImplementation/ServiceInstance">
        <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Service
            Instances</atom:title>
        <accept>application/atom+xml; type=entry
        </accept>
        <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
            <category scheme="urn:x-s-ramp:2010:type" term="ServiceInstance"
                label="Service Instance" xmlns="http://www.w3.org/2005/Atom"></category>
        </categories>
    </collection>
    <collection
        href="http://example.org/s-ramp/serviceImplementation/ServiceEndpoint">
        <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Service
            Endpoints</atom:title>
        <accept>application/atom+xml; type=entry
        </accept>
        <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
            <category scheme="urn:x-s-ramp:2010:type" term="ServiceEndpoint"
                label="Service Endpoint" xmlns="http://www.w3.org/2005/Atom"></category>
        </categories>
    </collection>
    <collection
        href="http://example.org/s-ramp/serviceImplementation">
        <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Service
            Implementation Objects</atom:title>
        <accept></accept>
        <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
            <category scheme="urn:x-s-ramp:2010:type" term="ServiceEndpoint"
                label="Service Endpoint" xmlns="http://www.w3.org/2005/Atom"></category>
            <category scheme="urn:x-s-ramp:2010:type" term="ServiceInstance"
                label="Service Instance" xmlns="http://www.w3.org/2005/Atom"></category>
            <category scheme="urn:x-s-ramp:2010:type" term="ServiceOperation"
                label="Service Operation" xmlns="http://www.w3.org/2005/Atom"></category>
            <category scheme="urn:x-s-ramp:2010:type" term="Organization"
                label="Organization" xmlns="http://www.w3.org/2005/Atom"></category>
        </categories>
    </collection>
</workspace>
<workspace>
    <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">SOAP WSDL Model
    </atom:title>
    <collection
        href="http://example.org/s-ramp/soapWsdl/SoapBinding">
        <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">SOAP Bindings
        </atom:title>
        <accept></accept>
        <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
```

```
                <category scheme="urn:x-s-ramp:2010:type" term="SoapBinding"
                    label="SOAP Binding" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/soapWsdl/SoapAddress">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">SOAP Addresses
            </atom:title>
            <accept></accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="SoapAddress"
                    label="SOAP Address" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/soapWsdl">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">SOAP WSDL Model
                Objects</atom:title>
            <accept>application/zip</accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="SoapAddress"
                    label="SOAP Address" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="SoapBinding"
                    label="SOAP Binding" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
    </workspace>
    <workspace>
        <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">SOA Model
        </atom:title>
        <collection
            href="http://example.org/s-ramp/soa/ServiceInterface">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Service
                Interface</atom:title>
            <accept>application/atom+xml;type=entry</accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="ServiceInterface"
                    label="Service Interface" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection href="http://example.org/s-ramp/soa">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">SOA Model Objects
            </atom:title>
            <accept>application/zip</accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="HumanActor"
                    label="HumanActor" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="Choreography"
                    label="Choreography" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="ChoreographyProcess"
                    label="Choreography Process"
xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="Collaboration"
                    label="Collaboration" xmlns="http://www.w3.org/2005/Atom"></category>
```

```
                <category scheme="urn:x-s-ramp:2010:type" term="CollaborationProcess"
                    label="Collaboration Process"
xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="Composition"
                    label="Composition" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="Effect"
                    label="Effect" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="Element"
                    label="Element" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="Event"
                    label="Event" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="InformationType"
                    label="Information Type" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="Orchestration"
                    label="Orchestration" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="OrchestrationProcess"
                    label="Orchestration Process"
xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="Policy"
                    label="Policy" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="PolicySubject"
                    label="Policy Subject" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="Process"
                    label="Process" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="Service"
                    label="Service" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="ServiceContract"
                    label="Service Contract" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="ServiceComposition"
                     label="Service Composition"
                     xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="ServiceInterface"
                    label="Service Interface" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="System"
                    label="System" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="Task"
                    label="Task" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/soa/CollaborationProcess">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Collaboration
                Process</atom:title>
            <accept>application/atom+xml;type=entry</accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="CollaborationProcess"
                    label="Collaboration Process"
xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/soa/Process">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Process
            </atom:title>
```

```
            <accept>application/atom+xml;type=entry</accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="Process"
                    label="Process" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/serviceImplementation/HumanActor">
            <atom:title type="text"
xmlns:atom="http://www.w3.org/2005/Atom">HumanActor</atom:title>
            <accept>application/atom+xml;type=entry</accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="HumanActor"
                    label="HumanActor" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/soa/Collaboration">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Collaboration
            </atom:title>
            <accept></accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="Collaboration"
                    label="Collaboration" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/soa/Composition">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Composition
            </atom:title>
            <accept>application/atom+xml;type=entry</accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="Composition"
                    label="Composition" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/soa/Element">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Element
            </atom:title>
            <accept>application/atom+xml;type=entry</accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="Element"
                    label="Element" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/soa/Event">
            <atom:title type="text"
xmlns:atom="http://www.w3.org/2005/Atom">Event</atom:title>
            <accept>application/atom+xml;type=entry</accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="Event"
                    label="Event" xmlns="http://www.w3.org/2005/Atom"></category>
```

```
    </categories>
</collection>
<collection
    href="http://example.org/s-ramp/soa/Orchestration">
    <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Orchestration
    </atom:title>
    <accept>application/atom+xml;type=entry</accept>
    <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
        <category scheme="urn:x-s-ramp:2010:type" term="Orchestration"
            label="Orchestration" xmlns="http://www.w3.org/2005/Atom"></category>
    </categories>
</collection>
<collection
    href="http://example.org/s-ramp/soa/PolicySubject">
    <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Policy Subject
    </atom:title>
    <accept>application/atom+xml;type=entry</accept>
    <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
        <category scheme="urn:x-s-ramp:2010:type" term="PolicySubject"
            label="Policy Subject" xmlns="http://www.w3.org/2005/Atom"></category>
    </categories>
</collection>
<collection
    href="http://example.org/s-ramp/soa/InformationType">
    <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Information
        Type</atom:title>
    <accept>application/atom+xml;type=entry</accept>
    <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
        <category scheme="urn:x-s-ramp:2010:type" term="InformationType"
            label="Information Type" xmlns="http://www.w3.org/2005/Atom"></category>
    </categories>
</collection>
<collection
    href="http://example.org/s-ramp/soa/Task">
    <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Task</atom:title>
    <accept>application/atom+xml;type=entry</accept>
    <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
        <category scheme="urn:x-s-ramp:2010:type" term="Task"
            label="Task" xmlns="http://www.w3.org/2005/Atom"></category>
    </categories>
</collection>
<collection
    href="http://example.org/s-ramp/soa/System">
    <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">System
    </atom:title>
    <accept>application/atom+xml;type=entry</accept>
    <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
        <category scheme="urn:x-s-ramp:2010:type" term="System"
            label="System" xmlns="http://www.w3.org/2005/Atom"></category>
    </categories>
</collection>
<collection
    href="http://example.org/s-ramp/soa/Policy">
    <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Policy
```

```
            </atom:title>
            <accept>application/atom+xml;type=entry</accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="Policy"
                    label="Policy" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/soa/Choreography">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Choreography
            </atom:title>
            <accept>application/atom+xml;type=entry</accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="Choreography"
                    label="Choreography" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/soa/Effect">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Effect
            </atom:title>
            <accept>application/atom+xml;type=entry</accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="Effect"
                    label="Effect" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/soa/ServiceContract">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Service Contract
            </atom:title>
            <accept>application/atom+xml;type=entry</accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="ServiceContract"
                    label="Service Contract" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/soa/OrchestrationProcess">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Orchestration
                Process</atom:title>
            <accept>application/atom+xml;type=entry</accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="OrchestrationProcess"
                    label="Orchestration Process"
xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/serviceImplemenation/Organization">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Organization
            </atom:title>
            <accept>application/atom+xml; type=entry</accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
```

```
                <category scheme="urn:x-s-ramp:2010:type" term="Organization"
                    label="Organization" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/soa/Service">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Service
            </atom:title>
            <accept>application/atom+xml; type=entry
            </accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="Service"
                    label="Service" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/soa/ChoreographyProcess">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Choreography
                Process</atom:title>
            <accept>application/atom+xml;type=entry</accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="ChoreographyProcess"
                    label="Choreography Process"
xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
    </workspace>
    <workspace>
        <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">XSD Model
        </atom:title>
        <collection
            href="http://example.org/s-ramp/xsd/XsdType">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">XSD Types
            </atom:title>
            <accept></accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="XsdType"
                    label="XSD Type" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/xsd/ElementDeclaration">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Element
                Declarations</atom:title>
            <accept></accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="ElementDeclaration"
                    label="Element Declaration" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/xsd/AttributeDeclaration">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Attribute
                Declarations</atom:title>
```

```
            <accept></accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="AttributeDeclaration"
                    label="Attribute Declaration"
xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/xsd/ComplexTypeDeclaration">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Complex Type
                Declarations</atom:title>
            <accept></accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="ComplexTypeDeclaration"
                    label="Complex Type Declaration"
xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/xsd/SimpleTypeDeclaration">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Simple Type
                Declarations</atom:title>
            <accept></accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="SimpleTypeDeclaration"
                    label="Simple Type Declaration"
xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/xsd/XsdDocument">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">XSD Documents
            </atom:title>
            <accept>application/xml</accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="XsdDocument"
                    label="XSD Document" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection href="http://example.org/s-ramp/xsd">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">XSD Model Objects
            </atom:title>
            <accept>application/zip</accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="XsdDocument"
                    label="XSD Document" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="AttributeDeclaration"
                    label="Attribute Declaration"
xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="XsdType"
                    label="XSD Type" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="ElementDeclaration"
                    label="Element Declaration" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="SimpleTypeDeclaration"
```

```
                    label="Simple Type Declaration"
xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="ComplexTypeDeclaration"
                    label="Complex Type Declaration"
xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
    </workspace>
    <workspace>
        <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Policy Model
        </atom:title>
        <collection
            href="http://example.org/s-ramp/policy/PolicyDocument">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Policy Documents
            </atom:title>
            <accept>application/xml</accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="PolicyDocument"
                    label="Policy Document" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection href="http://example.org/s-ramp/policy">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Policy Model
                Objects</atom:title>
            <accept>application/zip</accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="PolicyDocument"
                    label="Policy Document" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="PolicyExpression"
                    label="Policy Expression" xmlns="http://www.w3.org/2005/Atom"></category>
                <category scheme="urn:x-s-ramp:2010:type" term="PolicyAttachment"
                    label="Policy Attachment" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/policy/PolicyAttachment">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Policy
                Attachments</atom:title>
            <accept></accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="PolicyAttachment"
                    label="Policy Attachment" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
        <collection
            href="http://example.org/s-ramp/policy/PolicyExpression">
            <atom:title type="text" xmlns:atom="http://www.w3.org/2005/Atom">Policy
                Expressions</atom:title>
            <accept></accept>
            <categories fixed="yes" xmlns:atom="http://www.w3.org/2005/Atom">
                <category scheme="urn:x-s-ramp:2010:type" term="PolicyExpression"
                    label="Policy Expression" xmlns="http://www.w3.org/2005/Atom"></category>
            </categories>
        </collection>
```

```
    </workspace>
</service>
```

B S-RAMP URI Space

The URI space for S-RAMP is organized according to the logical structure of the S-RAMP Artifact Type Model. All S-RAMP artifacts are mapped to this URI space. The following URI syntax applies:

```
/s-ramp/{Primary-Qualifier}/{Secondary-Qualifier}
```

Typically, the Primary-Qualifier corresponds to the name of the Artifact Model, and the Secondary-Qualifier corresponds to an artifact type name. Exceptions include query and Service Document references. Table 6 defines the valid values for the components of S-RAMP URIs.

Table 6 - S-RAMP URI Space

Primary Qualifier	Secondary Qualifier
Document	
XmlDocument	
xsd	{Refer to the Artifact Type values in the xsd section of Table 7 of the Foundation Document}
policy	{Refer to the Artifact Type values in the policy section of Table 7 of the Foundation Document}
soapWsdl	{Refer to the Artifact Type values in the soapwsdl section of Table 7 of the Foundation Document}
wsdl	{Refer to the Artifact Type values in the wsdl section of Table 7 of the Foundation Document}
soa	{Refer to the Artifact Type values in the soa section of Table 7 of the Foundation Document}
serviceImplementation	{Refer to the Artifact Type values in the xsd section of Table 7 of the Foundation Document}
user	{User Defined Artifact Type}
query	{StoredQuery name}
	results
servicedocument	

X S-RAMP Atom Binding Schema

This appendix describes the S-RAMP structured extensions used in the Atom Binding. For convenience, an S-RAMP Atom Binding Schema xsd file is also provided at:

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

```
<xsd:schema
    targetNamespace="http://s-ramp.org/xmlns/2010/s-ramp" version="1.0"
    elementFormDefault="qualified"
    xmlns:tns="http://s-ramp.org/xmlns/2010/s-ramp"
    xmlns:s-ramp="http://s-ramp.org/xmlns/2010/s-ramp"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <xsd:include schemaLocation="serviceimplementationmodel.xsd"/>
    <xsd:include schemaLocation="coremodel.xsd"/>
    <xsd:include schemaLocation="wsdlmodel.xsd"/>
    <xsd:include schemaLocation="xsdmodel.xsd"/>
    <xsd:include schemaLocation="policymodel.xsd"/>
    <xsd:include schemaLocation="soamodel.xsd"/>
    <xsd:include schemaLocation="soapwsdlmodel.xsd"/>

    <!-- Base type for all Derived Artifacts in S-RAMP -->
    <xsd:element name="artifact">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:choice>
                    <!-- Concrete Artifact Types from Core Model -->
                    <xsd:element name="XmlDocument" type="s-ramp:XmlDocument"
                        minOccurs="1" maxOccurs="1"/>

                    <!-- Concrete Artifact Types from Service Implementation Model -->
                    <xsd:element name="Organization" type="s-ramp:Organization"
                        minOccurs="1" maxOccurs="1"/>
                    <xsd:element name="ServiceInstance" type="s-ramp:ServiceInstance"
                        minOccurs="1" maxOccurs="1"/>
                    <xsd:element name="ServiceOperation" type="s-ramp:ServiceOperation"
                        minOccurs="1" maxOccurs="1"/>
                    <xsd:element name="ServiceEndpoint" type="s-ramp:ServiceEndpoint"
                        minOccurs="1" maxOccurs="1"/>

                    <!-- Concrete Artifact Types from SOA Model -->
                    <xsd:element name="HumanActor" type="s-ramp:HumanActor"
minOccurs="1"
                        maxOccurs="1"/>
                    <xsd:element name="Choreography" type="s-ramp:Choreography"
                        minOccurs="1" maxOccurs="1"/>
                    <xsd:element name="ChoreographyProcess"
                        type="s-ramp:ChoreographyProcess" minOccurs="1" maxOccurs="1"/>
                    <xsd:element name="Collaboration" type="s-ramp:Collaboration"
                        minOccurs="1" maxOccurs="1"/>
                    <xsd:element name="CollaborationProcess"
                        type="s-ramp:CollaborationProcess" minOccurs="1"
                        maxOccurs="1"/>
                    <xsd:element name="Composition" type="s-ramp:Composition"
                        minOccurs="1" maxOccurs="1"/>
                    <xsd:element name="Effect" type="s-ramp:Effect" minOccurs="1"
                        maxOccurs="1"/>
                    <xsd:element name="Element" type="s-ramp:Element" minOccurs="1"
                        maxOccurs="1"/>
                    <xsd:element name="Event" type="s-ramp:Event" minOccurs="1"
                        maxOccurs="1"/>
                    <xsd:element name="InformationType" type="s-ramp:InformationType"
                        minOccurs="1" maxOccurs="1"/>
                    <xsd:element name="Orchestration" type="s-ramp:Orchestration"
                        minOccurs="1" maxOccurs="1"/>
                    <xsd:element name="OrchestrationProcess"
                        type="s-ramp:OrchestrationProcess" minOccurs="1"
                        maxOccurs="1"/>
                    <xsd:element name="Policy" type="s-ramp:Policy" minOccurs="1"
                        maxOccurs="1"/>
                    <xsd:element name="Organization"
                        type="s-ramp:Organization" minOccurs="1"
                        maxOccurs="1"/>
                    <xsd:element name="PolicySubject" type="s-ramp:PolicySubject"
                        minOccurs="1" maxOccurs="1"/>
```

```
<xsd:element name="Process" type="s-ramp:Process" minOccurs="1"
    maxOccurs="1"/>
<xsd:element name="Service" type="s-ramp:Service" minOccurs="1"
    maxOccurs="1"/>
<xsd:element name="ServiceContract" type="s-ramp:ServiceContract"
    minOccurs="1" maxOccurs="1"/>
<xsd:element name="ServiceInterface" type="s-ramp:ServiceInterface"
    minOccurs="1" maxOccurs="1"/>
<xsd:element name="System" type="s-ramp:System" minOccurs="1"
    maxOccurs="1"/>
<xsd:element name="Task" type="s-ramp:Task" minOccurs="1"
    maxOccurs="1"/>

<!-- Concrete Artifact Types from Policy Model -->
<xsd:element name="PolicyAttachment" type="s-ramp:PolicyAttachment"
    minOccurs="1" maxOccurs="1"/>
<xsd:element name="PolicyExpression" type="s-ramp:PolicyExpression"
    minOccurs="1" maxOccurs="1"/>
<xsd:element name="PolicyDocument" type="s-ramp:PolicyDocument"
    minOccurs="1" maxOccurs="1"/>

<!-- Concrete Artifact Types from XSD Model -->
<xsd:element name="XsdDocument" type="s-ramp:XsdDocument"
    minOccurs="1" maxOccurs="1"/>
<xsd:element name="AttributeDeclaration"
    type="s-ramp:AttributeDeclaration" minOccurs="1"
    maxOccurs="1"/>
<xsd:element name="ElementDeclaration"
    type="s-ramp:ElementDeclaration" minOccurs="1" maxOccurs="1"/>
<xsd:element name="XsdType" type="s-ramp:XsdType" minOccurs="1"
    maxOccurs="1"/>
<xsd:element name="ComplexTypeDeclaration"
    type="s-ramp:ComplexTypeDeclaration" minOccurs="1"
    maxOccurs="1"/>
<xsd:element name="SimpleTypeDeclaration"
    type="s-ramp:SimpleTypeDeclaration" minOccurs="1"
    maxOccurs="1"/>

<!-- Concrete Artifact Types from WSDL Model -->
<xsd:element name="WsdlDocument" type="s-ramp:WsdlDocument"
    minOccurs="1" maxOccurs="1"/>
<xsd:element name="WsdlService" type="s-ramp:WsdlService"
    minOccurs="1" maxOccurs="1"/>
<xsd:element name="Port" type="s-ramp:Port" minOccurs="1"
    maxOccurs="1"/>
<xsd:element name="Binding" type="s-ramp:Binding" minOccurs="1"
    maxOccurs="1"/>
<xsd:element name="PortType" type="s-ramp:PortType" minOccurs="1"
    maxOccurs="1"/>
<xsd:element name="BindingOperation" type="s-ramp:BindingOperation"
    minOccurs="1" maxOccurs="1"/>
<xsd:element name="BindingOperationInput"
    type="s-ramp:BindingOperationInput" minOccurs="1"
    maxOccurs="1"/>
<xsd:element name="BindingOperationFault"
    type="s-ramp:BindingOperationFault" minOccurs="1"
    maxOccurs="1"/>
<xsd:element name="OperationInput" type="s-ramp:OperationInput"
    minOccurs="1" maxOccurs="1"/>
<xsd:element name="Fault" type="s-ramp:Fault" minOccurs="1"
    maxOccurs="1"/>
<xsd:element name="Message" type="s-ramp:Message" minOccurs="1"
    maxOccurs="1"/>
<xsd:element name="Part" type="s-ramp:Part" minOccurs="1"
    maxOccurs="1"/>
<xsd:element name="BindingOperationOutput"
    type="s-ramp:BindingOperationOutput" minOccurs="1"
    maxOccurs="1"/>
<xsd:element name="OperationOutput"
    type="s-ramp:OperationOutput" minOccurs="1" maxOccurs="1"/>
<xsd:element name="WsdlExtension" type="s-ramp:WsdlExtension"
```

```
                    minOccurs="1" maxOccurs="1"/>

                <!--  Concrete Artifact Types from SOAP WSDL Model -->

                <xsd:element name="SoapAddress" type="s-ramp:SoapAddress"
                    minOccurs="1" maxOccurs="1"/>
                <xsd:element name="SoapBinding" type="s-ramp:SoapBinding"
                    minOccurs="1" maxOccurs="1"/>
            </xsd:choice>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<!-- Relationship Data element used in S-RAMP Relationship Entry documents -->
<xsd:element name="relationshipData">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="s-ramp:relationshipType" minOccurs="0"
                maxOccurs="1" />
            <!-- sourceId is the UUID of the source artifact -->
            <xsd:element ref="tns:sourceId" minOccurs="0" maxOccurs="1" />
            <!-- targetId is the UUID of the target artifact -->
            <xsd:element ref="tns:targetId" minOccurs="1" maxOccurs="1" />
</xsd:sequence>
    </xsd:complexType>
</xsd:element>

<!-- Relationship Type Data element used in S-RAMP Relationship Type Entry
documents. For now this only includes the s-ramp:relationshipType.  -->
<xsd:element name="relationshipTypeData">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="s-ramp:relationshipType" minOccurs="1"
                maxOccurs="1" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<!-- Properties Data element used in S-RAMP Property Entry documents.  -->
<xsd:element name="propertyData">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="s-ramp:property" minOccurs="1" maxOccurs="1" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<!-- Properties Data element used in S-RAMP Classification Entry documents.  -->
<xsd:element name="classificationData">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="s-ramp:classifiedBy" minOccurs="1" maxOccurs="1" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<!-- Stored Query Data element used in S-RAMP Classification Entry documents.  -->
<xsd:element name="storedQueryData">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="queryName" type="xsd:string" minOccurs="1"
                maxOccurs="1" />
            <xsd:element name="queryString" type="xsd:string" minOccurs="1"
                maxOccurs="1" />
            <xsd:element ref="s-ramp:propertyName" minOccurs="0"
                maxOccurs="unbounded" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
</xsd:schema>
```