Overlord S-RAMP Guide

Chapter 1. Introduction to S-RAMP

1.1. The S-RAMP Specification

S-RAMP stands for SOA Repository Artifact Model and Protocol. *S-RAMP* [https://www.oasis-open.org/committees/s-ramp/charter.php] is a new specification worked on by the OASIS Technical Committee.

> The SOA Repository Artifact Model and Protocol (S-RAMP) TC defines a common data model for SOA repositories as well as an interaction protocol to facilitate the use of common tooling and sharing of data. The TC will define an ATOM binding which documents the syntax for interaction with a compliant repository for create, read, update, delete and query operations.
>
> — OASIS Charter *https://www.oasis-open.org/committees/s-ramp/charter.php*

The first version of the specification (1.0) should be finalized in the first half of 2013. Two of the developers on the project participated in the Technical Committee.

1.2. Purpose

The OASIS S-RAMP (SOA Repository Artifact Model and Protocol) specification is intended to provide a common data model and protocol for interacting with a repository of (primarily) SOA artifacts. The goal of the specification is to foster interoperability between repository implementations by standardizing on a data model and API.

This guide will discuss both the OASIS standard and the Overlord open source implementation.

1.3. Overview

The S-RAMP specification includes a foundation document and an Atom based protocol binding document. The foundation document describes core concepts and will be the focus of the first part of this guide. The Atom binding document describes an Atom based API that implementations should provide.

An S-RAMP repository primarily stores artifacts. An artifact is comprised of the following meta-data:

1.4. Core Properties

Relationships. All artifacts in S-RAMP contain a set of core properties such as name, description, creation date, etc. Many of these properties are automatically set by the server when the artifact is added and/or updated. Others, such as description, can be set by clients.

However, most importantly every artifact has an Artifact Model and an Artifact Type. These two properties determine what kind of artifact it is (more on artifact types later, in the Data Models section of this Guide).

Additionally, some artifact types contain additional core properties. For example, the Document artifact type includes additional core properties of contentType and contentSize, while the XsdDocument artifact type includes the targetNamespace property.

1.5. Custom Properties

An artifact may have additional properties set on it by clients. These custom properties are simply arbitrary name/value pairs. The only restriction is that a custom property may not have the same name as a Core Property.

1.6. Classifiers

Another type of meta-data found on S-RAMP artifacts are classifiers. Classifiers are a lot like keywords or tags except that they are hierarchical. Every artifact has a collection of classifiers configured by the client, where each classifier must be a node in an ontology previously uploaded to the repository (presumably by an admin).

An ontology is simply a hierarchy of tags (defined as a subset of the OWL Lite format). This approach allows the repository to be configured with a pre-defined set of hierarchical tags (classifiers) that can be associated with an artifact.

An example is helpful in this case. First, a repository administrator would define and upload an ontology:

```
World
  |-> North America
     |-> United States
         |-> Alabama
         |-> Alaska
     |-> Mexico
     |-> Canada
  |-> South America
  |-> Australia
```

Once this ontology has been added to the repository, then clients can add, for example, #Alaska or #Canada as classifiers on artifacts. This provides a way to "tag" artifacts in interesting and meaningful ways, and provides a useful means of querying (more on that later).

For more information about ontologies and classifiers, have a look at Section 3 of the S-RAMP Foundation document.

1.7. Relationships

The final bit of meta-data that can be found on an artifact are relationships An S-RAMP relationship is a uni-directional link between a source artifact and a target artifact. Artifacts can have arbitrary, client-defined relationships. Every relationship has a name and a target artifact. For example, a

client might define a relationship named "documentedBy" between a wsdl artifact and a text or PDF artifact, indicating that the latter provides documentation for the former.

Chapter 2. Getting Started

2.1. Prerequisites

The S-RAMP application is written in Java. To get started make sure your system has the following:

- Java JDK 1.6 or newer
- Apache Ant 1.7 or newer to use the installer
- Maven 3.0.3 or newer to build and run the examples

2.2. Download, Installation and Configuration

The `s-ramp-<version>.zip` (or tar.gz) archive can be downloaded from the *http://www.jboss.org/overlord* website. Grab the latest, extract the archive and run:

```
ant install
```

The installer will ask you to choose a runtime platform. Currently the following platforms are supported:

- JBoss EAP 6
- JBoss Fuse 6.1
- Tomcat 7
- Jetty 8

Simply follow the installer instructions to install onto the platform of your choice. We recommend installing into a clean version of which platform you choose, to minimize the risk of colliding with other projects.

Note that you must have already downloaded and installed the platform on which you wish to run, with the exception of Tomcat 7 and Jetty 8 (which the installer will download for you if you choose).

Finally, please make sure the admin password you choose (the installer will prompt you for this) contains letters, numbers, and punctuation (and is at least 8 characters long).

Tip

Read the installer output carefully - extra instructions are given for certain platforms.

Once S-RAMP is installed on your preferred platform, you should be able to go ahead and start it up. The instructions for starting the server depend on the chosen platform:

Tomcat 7.

```
bin/startup.sh
```

Jetty 8.

```
bin/jetty.sh run
```

JBoss EAP 6.

```
bin/standalone.sh
```

JBoss Fuse 6.1.

```
bin/fuse
```

2.3. Check your Installation

To make sure your installation works you can fire up the *s-ramp-ui* [http://localhost:8080/s-ramp-ui]. You should see the GUI dashboard and be able to navigate to either the Artifacts or Ontologies management pages:

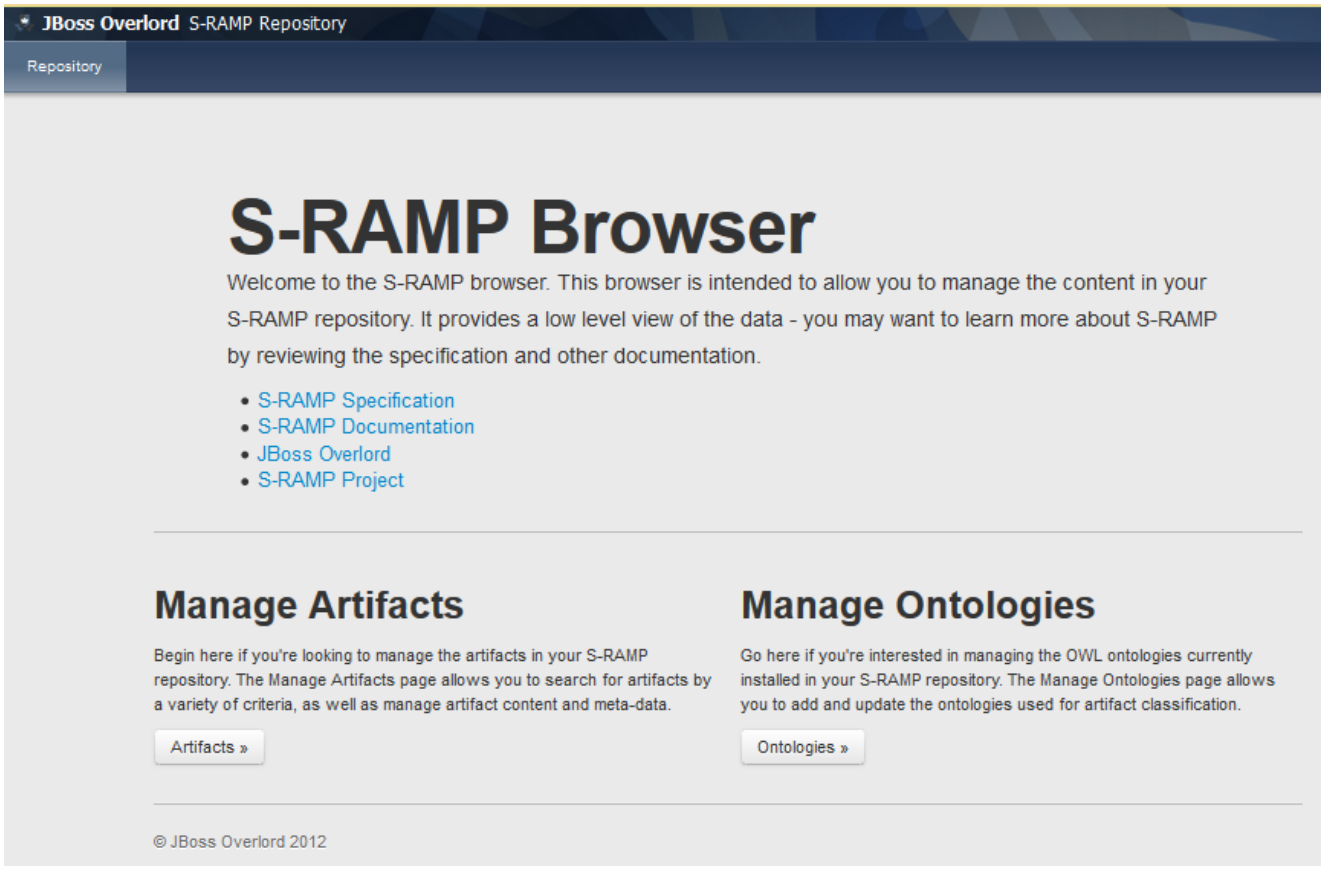

Figure 2.1. Welcome screen of the s-ramp-ui.

You can click on `Artifacts` and see a list of files related to the S-RAMP default workflows.

Alternatively you can fire up the `s-ramp shell` in the `bin` directory of the distribution:

```
./bin/s-ramp.sh
************************************************************************
          _____       ______  ___ ___  ________
         /  ___|      | ___ \/ _ \|  \/  | ___ \
         \ `--. ______| |_/ / /_\ \ .  . | |_/ /
          `--. \______|    /|  _  | |\/| |  __/
         /\__/ /      | |\ \| | | | |  | | |
         \____/       \_| \_\_| |_|_|  |_|_|

  JBoss S-RAMP Kurt Stam and Eric Wittmann, Licensed under the
  Apache License, V2.0, Copyright 2012
************************************************************************
s-ramp>
```

To connect the shell to the server type `connect` and hit the tab key. It should auto-complete to say `s-ramp:connect http://localhost:8080/s-ramp-server` and when hitting the return key you should be prompted for user credentials. Use *admin* and whatever password you entered during installation. If this succeeds, the shell cursor/prompt will go from red to green. To browse the artifacts in the repository (there will likely not be any) run the following query:

```
s-ramp> s-ramp:query "/s-ramp"
Querying the S-RAMP repository:
 /s-ramp
Atom Feed (0 entries)
  Idx                   Type Name
  ---                   ---- ----
```

In later chapters will go into more detail, but if this all worked you can be sure that your installation is in good working order.

Chapter 3. User Management

3.1. Overview

In order to perform any S-RAMP operations (including both read and write operations) a valid user must be authenticated. The specific details regarding how to create and manage the list of allowed users will vary depending on the runtime configuration. This guide will focus on the mechanisms supported by the S-RAMP community installer.

Tip

Please note that the installer creates a single user (named *admin*) during the installation process.

3.2. Required Roles

There are several roles that the user must have in order to interact with the S-RAMP repository. These roles are as follows:

- **overlorduser** : users must have this role in order to access the S-RAMP user interface (browser)
- **admin.sramp** : users must have this role in order to access the S-RAMP repository (both read and write)

Tip

if the ModeShape repository name is changed, the above *admin.sramp* role would need to reflect that change.

3.3. Adding a User

3.3.1. JBoss EAP 6

By default S-RAMP uses the standard EAP Application Realm configuration as its authentication source. This means that adding users is a simple matter of using the existing EAP **add-user** script. If you are running on Windows you can use the add-user.bat script. Otherwise run the add-user.sh script. Both of these scripts can be found in EAP's *bin* directory.

Here is an example of how to add an S-RAMP user using the **add-user.sh** script:

```
[user@host jboss-eap-6.x]$ pwd
/home/user/FSW6/jboss-eap-6.x
```

```
[user@host jboss-eap-6.x]$ ./bin/add-user.sh

What type of user do you wish to add?
 a) Management User (mgmt-users.properties)
 b) Application User (application-users.properties)
(a): b

Enter the details of the new user to add.
Realm (ApplicationRealm) : ApplicationRealm
Username : fitzuser
Password : P4SSW0RD!
Re-enter Password : P4SSW0RD!
What roles do you want this user to belong to? (Please enter a comma
 separated list, or leave blank for none)[  ]: overlorduser,admin.sramp
About to add user 'fitzuser' for realm 'ApplicationRealm'
Is this correct yes/no? yes
Added user 'fitzuser' to file '/home/user/FSW6/jboss-eap-6.x/standalone/
configuration/application-users.properties'
Added user 'fitzuser' to file '/home/user/FSW6/jboss-eap-6.x/domain/
configuration/application-users.properties'
Added user 'fitzuser' with roles overlorduser,admin.sramp to file '/
home/user/FSW6/jboss-eap-6.x/standalone/configuration/application-
roles.properties'
Added user 'fitzuser' with roles overlorduser,admin.sramp to file '/home/
user/FSW6/jboss-eap-6.x/domain/configuration/application-roles.properties'
Is this new user going to be used for one AS process to connect to another
 AS process?
e.g. for a slave host controller connecting to the master or for a Remoting
 connection for server to server EJB calls.
yes/no? no
```

3.3.2. JBoss Fuse 6.1

When running S-RAMP in JBoss Fuse 6.1, the user credentials are stored in a plain text properties file in the `etc` directory.

etc/users.properties.

```
#user=password,role1,role2
admin=ADMIN_PASSWORD,overlorduser,admin.sramp
```

Simply add users to this file and restart Fuse. Make sure you include the necessary roles of `overlorduser` and `admin.sramp` in any user you create.

3.3.3. Tomcat 7

When running S-RAMP in Tomcat 7, the source of authentication is an XML configuration file located in Tomcat's *conf* directory named **tomcat-users.xml**. To add another user, simply add

a **user** element to this XML configuration file. For example, adding a user named *fitzuser* might make the file look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<tomcat-users>
<!--
  NOTE:  By default, no user is included in the "manager-gui" role required
  to operate the "/manager/html" web application.  If you wish to use this
 app,
  you must define such a user - the username and password are arbitrary.
-->
  <role rolename="tomcat"/>
  <role rolename="overlorduser"/>
  <role rolename="admin.sramp" />
  <user username="admin" password="4dm1n!"
 roles="tomcat,overlorduser,admin.sramp"/>
  <user username="fitzuser" password="P4SSW0RD!"
 roles="tomcat,overlorduser,admin.sramp"/>
</tomcat-users>
```

3.3.4. Jetty 8

When running in Jetty 8, the users are configured in the $JETTY_HOME/etc/realm.properties file. The format is the same as the above Fuse 6 documentation:

```
username=password123!,overlorduser,admin.sramp
```

Chapter 4. S-RAMP Data Models

The S-RAMP specification defines a number of built-in artifact types, while also allowing clients to define their own (implicit) types. This section of the Guide describes these different models.

An artifact may have document (e.g file) content or it may be a purely logical artifact. In either case, clients typically add artifacts to the repository directly (e.g. via the S-RAMP Atom API, described later in this guide).

Additionally, some document style artifact types when added to the repository, will result in the creation of a set of "derived" artifacts. For example, if an XSD document is added to the repository, the server will automatically extract the element declarations from the content of the file resulting in a set of additional artifacts "related" to the original. This will be described in detail further in the XSD Data Model section.

4.1. Core Data Model (core)

The S-RAMP core model defines some basic artifact types including Document and XmlDocument. These basic types allow clients to add simple files to the repository as artifacts.

Artifact Type	Parent Type	Properties
Document		contentType, contentSize, contentHash
XmlDocument	Document	contentEncoding

4.2. XML Schema (XSD) Data Model (xsd)

The XSD model defines a single document style artifact, XsdDocument, and a number of derived artifact types. When an XSD document is added to the repository, the server will additionally "index" the artifact by automatically creating a number of derived artifacts of the following types from the XSD content.

Artifact Type	Parent Type	Properties
XsdDocument	XmlDocument	targetNamespace
AttributeDeclaration	<derived>	ncName, namespace
ElementDeclaration	<derived>	ncName, namespace
SimpleTypeDeclaration	<derived>	ncName, namespace
ComplexTypeDeclaration	<derived>	ncName, namespace

4.3. WSDL Data Model (wsdl)

The WSDL model defines a single document style artifact, WsdlDocument, and a number of derived artifact types. Similarly to the XsdDocument type, when a WSDL dcument is added to the

repository, the server will automatically derive additional artifacts (listed below) from the content of the WSDL file.

For further details about the WSDL Model, please see the S-RAMP specification's foundation document, section 2.4.2.

Artifact Type	Parent Type	Properties
WsdlDocument	XmlDocument	targetNamespace, xsdTargetNamespaces
WsdlService	<derived>	ncName, namespace
Port	<derived>	ncName, namespace
WsdlExtension	<derived>	ncName, namespace
Part	<derived>	ncName, namespace
Message	<derived>	ncName, namespace
Fault	<derived>	ncName, namespace
PortType	<derived>	ncName, namespace
Operation	<derived>	ncName, namespace
OperationInput	<derived>	ncName, namespace
OperationOutput	<derived>	ncName, namespace
Binding	<derived>	ncName, namespace
BindingOperation	<derived>	ncName, namespace
BindingOperationInput	<derived>	ncName, namespace
BindingOperationOutput	<derived>	ncName, namespace
BindingOperationFault	<derived>	ncName, namespace

4.4. Policy Data Model (policy)

This data model is present to represent the primary components of a WS-Policy document.

Artifact Type	Parent Type	Properties
PolicyDocument	XmlDocument	
PolicyExpression	<derived>	
PolicyAttachment	<derived>	

4.5. SOA Data Model (soa)

The SOA model exists to provide a link to the work done by the Open Group SOA Ontology group. All of the artifacts in this model are non-document artifacts which are directly instantiated by clients.

Artifact Type	Parent Type	Properties
HumanActor		

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

4.6. Service Implementation Data Model (serviceImplementation)

The Service Implementation model adds SOA service implementation artifact types underneath the (already mentioned) SOA Data Model.

Artifact Type	Parent Type	Properties
Organization		end
ServiceEndpoint		end, url
ServiceInstance		end, url
ServiceOperation		end, url

4.7. Custom/Extension Data Models (ext)

Clients can define their own (implicit) data models by using the "ext" model space defined by the S-RAMP specification. This allows clients to add documents with custom artifact types. For

example, a client can add an artifact to /s-ramp/ext/PdfDocument. This provides a way for clients to define their own data models with their own properties and relationships. Note however that the server will not have a definition of the model - it is up to the client to properly conform to their own implicit model. Custom properties and user-defined relationships allow clients to richly define their own models.

As an example, a client might define the following Data Model for a J2EE web application domain:

Artifact Type	Parent Type	Properties
WebXmlDocument	ExtendedDocument	displayName
ServletFilter	ExtendedArtifactType	displayName, filterClass
Servlet	ExtendedArtifactType	servletClass, loadOnStartup

4.8. Java Data Model

The S-RAMP server adds a custom (ext) model for dealing with Java artifacts. The following table lists the Java artifact types that are supported out of the box. These artifacts can either be directly added to the repository or they can be added as part of the expansion of some other artifact. A typical example of the latter is how JavaClass artifacts may be added because they are contained within an archive of some kind.

Artifact Type	Parent Type	File Extension	Properties
JavaArchive	ExtendedDocument	jar	
JavaWebApplication	ExtendedDocument	war	
JavaEnterpriseApplication	ExtendedDocument	ear	
BeanArchiveDescriptor	ExtendedDocument	beans.xml	
JavaClass	ExtendedDocument	class	packageName, className
JavaInterface	ExtendedDocument	class	packageName, className
JavaEnum	ExtendedDocument	class	packageName, className

4.9. KIE Data Model (Knowledge is Everything)

The S-RAMP server includes basic out of the box support for Drools (KIE) artifact types. The following table lists the KIE artifact types that are currently supported.

Artifact Type	Parent Type	File Extension	Properties
KieJarArchive	ExtendedDocument	jar	
KieXmlDocument	ExtendedDocument	kmodule.xml	

Artifact Type	Parent Type	File Extension	Properties
BpmnDocument	ExtendedDocument	bpmn	
DroolsDocument	ExtendedDocument	drl	

4.10. SwitchYard Data Model

The S-RAMP server inclues a custom (ext) data model for SwitchYard artifacts. The following table lists the artifact types currently supported. The non-derived artifacts can be added directly to the repository or expanded out of some archive type artifact.

Artifact Type	Parent Type	Properties
SwitchYardApplication	ExtendedDocument	
SwitchYardXmlDocument	ExtendedDocument	targetNamespace
SwitchYardService	<derived from SwitchYardXmlDocument>	
SwitchYardComponent	<derived from SwitchYardXmlDocument>	requires
SwitchYardComponentService	<derived from SwitchYardXmlDocument>	
SwitchYardTransformer	<derived from SwitchYardXmlDocument>	transformer-type
SwitchYardValidator	<derived from SwitchYardXmlDocument>	validator-type

Additionally, the SwitchYard derived artifacts have various relationships automatically created between and amongst them (as well as to other derived artifacts such as those derived from XML Schema artifacts). The following table outlines all the relationships currently defined in the SwitchYard Model.

Relationship	Source Artifact Type	Target Artifact Type
implementedBy	SwitchYardComponent	JavaClass
implementedBy	SwitchYardTransformer	JavaClass
implementedBy	SwitchYardValidator	JavaClass
implements	SwitchYardService	JavaInterface, PortType
implements	SwitchYardComponentService	JavaInterface, PortType
offers	SwitchYardComponent	SwitchYardComponentService
promotes	SwitchYardService	SwitchYardComponent
references	SwitchYardComponent	JavaInterface, PortType
transformsFrom	SwitchYardTransformer	JavaClass, ElementDeclaration

Relationship	Source Artifact Type	Target Artifact Type
transformsTo	SwitchYardTransformer	JavaClass, ElementDeclaration
validates	SwitchYardValidator	JavaClass, ElementDeclaration

4.11. Teiid Data Model (Teiid)

The Teiid model adds Teiid-related artifact types, derived artifacts, and relationships. There are artifact types for the following Teiid resources: VDBs (`*.vdb`), models (`*.xml`), VDB manifests (usually named `vdb.xml`), and VDB configuration files (`ConfigurationInfo.def`). Teiid resources should be added to the repository using the corresponding artifact types listed in the following table:

Artifact Type	Parent Type	Properties
TeiidVdb		
TeiidModel	ExtendedArtifactType	description, maxSetSize, mmuuid, modelType, nameInSource, primaryMetamodelUri, producerName, producerVersion, visible
TeiidVdbConfigInfo	ExtendedArtifactType	
TeiidVdbManifest	ExtendedArtifactType	description, preview, UseConnectorMetadata, vdbVersion, <custom properties>
TeiidVdbDataPolicy	<derived from TeiidVdbManifest>	anyAuthenticated, description, roleNames, tempTableCreatable
TeiidVdbEntry	<derived from TeiidVdbManifest>	description, <custom properties>
TeiidVdbImportVdb	<derived from TeiidVdbManifest>	importDataPolicies, vdbVersion
TeiidVdbPermission	<derived from TeiidVdbManifest>	alterable, condition, creatable, deletable, executable, languagable, mask, readable, updatable
TeiidVdbSchema	<derived from TeiidVdbManifest>	builtIn, checksum, description, indexName, metadata, metadataType, modelClass, modelUuid,

Artifact Type	Parent Type	Properties
		pathInVdb, schemaType, visible, <custom properties>
TeiidVdbSource	<derived from TeiidVdbManifest>	jndiName, translatorName
TeiidVdbTranslator	<derived from TeiidVdbManifest>	description, translatorType, <custom properties>
TeiidVdbValidationError	<derived from TeiidVdbManifest>	message, severity

When a `TeiidVDB` or a `TeiidVdbManifest` artifact type is added to the repository, relationships between it and its derived artifacts are created. Note that the `TeiidVdbContains` relationship is the inverse of the `expandedFromDocument` relationship. Here is a list of the Teiid relationship types:

Relationship Type	Source Type	Target Type	Multiplicity
TeiidVdbContains	TeiidVdbManifest	TeiidVdbDataPolicy, TeiidVdbEntry, TeiidVdbSchema, TeiidVdbTranslator, TeiidVdbImportVdb	1 to many
TeiidVdbDataPolicyPermissions	TeiidVdbDataPolicy	TeiidVdbPermission	1 to many
TeiidVdbPermissionDataPolicy	TeiidVdbPermission	TeiidVdbDataPolicy	1 to 1
TeiidVdbSchemaSource	TeiidVdbSchema	TeiidVdbSource	1 to 1
TeiidVdbSchemaValidationErrors	TeiidVdbSchema	TeiidVdbValidationError	1 to many
TeiidVdbSourceSchemas	TeiidVdbSource	TeiidVdbSchema	1 to many
TeiidVdbSourceTranslator	TeiidVdbSource	TeiidVdbTranslator	1 to 1
TeiidVdbTranslatorSources	TeiidVdbTranslator	TeiidVdbSource	1 to many
TeiidVdbValidationErrorSource	TeiidVdbValidationError	TeiidVdbSource	1 to 1
expandedFromDocument	TeiidVdbDataPolicy, TeiidVdbEntry, TeiidVdbSchema, TeiidVdbTranslator, TeiidVdbImportVdb	TeiidVdbManifest	many to 1

Chapter 5. Query Language

Another key aspect of the S-RAMP specification is the query language it defines, which allows clients to find artifacts by various criteria. The S-RAMP query language is a subset of the XPath 2.0 language, designed specifically to find and select S-RAMP artifacts.

Tip

for detailed information about the S-RAMP Query Language, see Section 4 of the S-RAMP specification's foundation document.

As you might imagine, the query language allows clients to find artifacts based on any of the already discussed artifact meta-data, including:

*Core Properties *Custom Properties *Classifiers *Relationships

The basic structure of a typical S-RAMP query looks like this:

```
/s-ramp/<artifactModel>/<artifactType>[ <artifact-predicate> ]/
relationship[ <target-artifact-predicate> ]
```

Of course, not all of the components of the above query are required. It is likely best to provide the following table of examples of a range of queries:

Query	What It Selects
/s-ramp	All artifacts.
/s-ramp/core	All Core Model artifacts.
/s-ramp/xsd/XsdDocument	All XsdDocument artifacts.
/s-ramp/xsd/XsdDocument[@my-prop]	All XsdDocument artifacts that have the custom property *my-prop* defined (with any value).
/s-ramp/xsd/ XsdDocument[@name='core.xsd']	XsdDocument artifacts named *core.xsd*.
/s-ramp/xsd/XsdDocument[@name='core.xsd' and @version='1.0']	XsdDocument artifacts named *core.xsd* and versioned as *1.0*.
/s-ramp/soa[@myCustomProperty='foo']	SOA artifacts with a custom property named *myCustomProperty* that has value *foo*.
/s-ramp/core[classifiedByAnyOf(., 'Maine', 'Alaska')]	Core artifacts classified by either *Maine* or *Alaska* (presumably from the *Regions* ontology).

Query	What It Selects
/s-ramp/wsdl/PortType[@name='OrderServicePT']/operation	Artifacts related to any *PortType* artifact named *OrderServicePT* via a relationship named *operation*. (This effectively returns all of the order service port type's operations)
/s-ramp/ext/ServletFilter[relatedDocument[@uuid='12345']]	All servlet filter artifacts derived from (i.e. contain a *relatedDocument* relationship to) an artifact with UUID *12345*.
/s-ramp/wsdl/Message[xp2:matches(.,'get.*')]/part[element]	Element style WSDL parts from WSDL messages with names starting with *get*.

Chapter 6. S-RAMP REST API

The intent of the S-RAMP specification is to outline a data model and protocol designed to define how a repository should store and manipulate artifacts. The foundation document defines the former, while various protocol binding documents define the latter. Version 1 of the S-RAMP specification includes a single, Atom based protocol binding.

This section of the guide describes the highlights of the Atom API.

Tip

For more information on the S-RAMP Atom API, see the S-RAMP Atom Binding document.

6.1. Overview

The S-RAMP specification does not dictate the format of the Atom REST endpoints. Instead, the client is expected to query a service document endpoint and inspect it to find the various relevant endpoints. The specification does present a notional format, but implementations are not required to follow it. This Guide will give examples based on this notional format. But it bears repeating that for any given server implementation, a client should first query the Atom service document at the following endpoint:

```
GET /s-ramp/servicedocument
```

The resulting service document will contain a set of workspaces representing the artifact collections supported by the server, along with endpoints indicating how to manipulate them.

However, the Atom Binding document does outline the inputs, outputs, and REST verbs that must be used for each of the supported operations. In general, the Atom API data models are used to wrap custom S-RAMP specific XML structures. Atom Entry documents are used when dealing with individual artifacts, while Atom Feed documents are used when dealing with lists of documents.

6.2. Adding Artifacts

There are two basic types of artifacts from the protocol standpoint: document style artifacts (those artifacts that are based on files/binary content) and logical (direct instantiation) artifacts. In the case of a document-style artifact, the client must POST the binary content to the correct Atom Endpoint. In the case of a direct artifact (no document content) the client must POST an Atom Entry containing an S-RAMP artifact XML entity to the appropriate endpoint. The server will respond with an Atom Entry containing the full meta data of the newly created artifact (if successful).

Notional REST Endpoint

```
POST /s-ramp/{model}/{type}
```

6.3. Updating Artifacts

A client can update the meta data (properties, classifiers, relationships) by performing a PUT request to the artifact's endpoint. The artifact's endpoint can be found either by querying for the artifact or as part of the returned Atom Entry returned when the artifact was created.

Notional REST Endpoint

```
PUT /s-ramp/{model}/{type}/{uuid}
```

6.4. Deleting Artifacts

Not surprisingly, an artifact can be deleted by a client by performing a DELETE request to the artifact's endpoint.

Notional REST Endpoint

```
DELETE /s-ramp/{model}/{type}/{uuid}
```

6.5. Querying

Performing an S-RAMP query is a matter of issueing a GET or POST to the S-RAMP query endpoint. In addition, full feeds are available for all Artifact Models and Artifact Types. In both cases, the response is an Atom Feed where each Entry provides summary information about an artifact in the respository. To retrieve full details about a given entry in the feed (custom properties, classifiers, relationships), the client must issue an additional GET. Only a subset of the core properties, such as name and description, are mapped to the Atom Entry in a feed.

6.5.1. Queries

When querying, the client can either GET or POST to the following notional endpoint:

Notional REST Endpoint

```
GET /s-ramp
```

The following parameters are supported (all parameters except the *query* param have reasonable defaults):

- *query*: The S-RAMP query.
- *startPage*: The page to start from.
- *startIndex*: The index number to start from.

- *count*: The number of artifacts to return.
- *orderBy*: The sort order to use when creating the feed.
- *ascending*: The sort direction to use when creating the feed.
- *propertyName*: Additional custom property to return for each artifact in the feed. This property can be included multiple times.

6.5.2. Feeds

When retrieving a simple model or type feed, the client must issue a GET request to the appropriate model or type endpoint.

Notional REST Endpoint

```
GET /s-ramp/{model}
GET /s-ramp/{model}/{type}
```

The following parameters are supported (all parameters have reasonable defaults):

- *startPage*: The page to start from.
- *startIndex*: The index number to start from.
- *count*: The number of artifacts to return.
- *orderBy*: The sort order to use when creating the feed.
- *ascending*: The sort direction to use when creating the feed.
- *propertyName*: Additional custom property to return for each artifact in the feed. This property can be included multiple times.

6.6. Getting Full Artifact

In order to retrieve the full meta data for an artifact, the client must issue a GET request to the appropriate artifact endpoint. This is necessary after a query or feed, when only the summary information is available. The summary information found in a feed or query response contains the UUID of the artifact, as well as a URL to the endpoint needed to retrieve the full artifact details.

Notional REST Endpoint

```
GET /s-ramp/{model}/{type}/{uuid}
```

6.7. Batch changes: S-RAMP Archives (Packages)

A powerful additional feature of the S-RAMP API is the batch processing function. The batch processing endpoint allows the client to POST an S-RAMP package, which can contain multiple

Atom Entries and binary files. The package allows a client to add, update, and delete multiple artifacts in a single batch. The format of an S-RAMP package archive is described further in the S-RAMP Atom Binding document (Section 2.3.5.2.2.1 - no I'm not kidding, that's really the section).

Chapter 7. Overlord S-RAMP Implementation

7.1. Overview

The Overlord S-RAMP implementation strives to be a fully compliant reference implementation of the S-RAMP specification. This chapter describes the overall architecture of the implementation and also provides some information about how to configure it.

Overlord S-RAMP also provides a Java based client library that consumers can use to integrate their own applications with an S-RAMP compliant server.

7.2. Server

7.2.1. Description

The server implementation is a conventional Java web application (WAR). The following technologies are used to provide the various components that make up the server implementation:

1. JCR (ModeShape) - used as the persistence engine, where all S-RAMP data is stored. Artifacts and ontologies are both stored as nodes in a JCR tree. All S-RAMP queries are mapped to JCRSQL2 queries for processing by the JCR API. The ModeShape JCR implementation is used by default. However, the persistence layer is pluggable allowing alternative providers to be implemented in the future.
2. JAX-RS (RESTEasy) - used to provide the S-RAMP Atom based REST API. The S-RAMP specification documents an Atom based REST API that implementations should make available. The Overlord S-RAMP implementation uses JAX-RS (specifically RESTEasy) to expose all of the REST endpoints defined by the specification.
3. JAXB - used to expose a Java data model based on the S-RAMP data structures defined by the specification (S-RAMP XSD schemas).

7.2.2. Configuring

Configuration of the server can be done by providing a configuration file to the server on startup. The configuration file can be provided in a number of ways:

1. sramp.properties - an external file can be provided in the JBoss application server's configuration directory. An alernative location is the home directory of the user running the application server.
2. custom external file - a custom location for an sramp.properties file can be specified by starting the application server with the **sramp.config.file.name** system property set. This is typically

done using **-Dsramp.config.file.name=<pathToFile>** on the application server's command line startup script (often in JAVA_OPTS).

3. on the classpath - if no external file is found, the classpath is used to lookup a default configuration.

The configuration file is a simple Java properties file, with the following properties available to be set:

```
# The base URL of the S-RAMP server - can be useful in some advanced
 configurations where
# the incoming Request URL is not the canonical server address.
sramp.config.baseurl
# Turn on/off auditing of changes to S-RAMP artifacts
sramp.config.auditing.enabled
# Turn on/off auditing of changes to derived S-RAMP artifacts
sramp.config.auditing.enabled-derived
```

7.2.3. Configuring (EAP)

When running in JBoss EAP this same configuration information is instead stored in the **JBOSS/standalone/configuration/standalone.xml** file under the **urn:jboss:domain:overlord-configuration:1.0** subsystem. For example:

```
    <subsystem xmlns="urn:jboss:domain:overlord-configuration:1.0">
      <configurations>
        <configuration name="sramp">
          <properties>
            <property name="sramp.config.auditing.enabled" value="true" />
            <property name="sramp.config.auditing.enabled-derived"
value="true" />
          </properties>
        </configuration>
      </configurations>
    </subsystem>
```

7.2.4. Security (Authentication)

The S-RAMP repository Atom API is protected using standard web application security and JAAS on the backend. When deploying to an application server, security should be configured according to the specifics of the container. Typically the API would be protected by a simple BASIC authentication scheme, but in some more advanced configurations it would be appropriate to use OAuth or SAML Bearer Token authentication mechanisms.

When deploying into JBoss, the S-RAMP distribution adds a JBoss security domain named "overlord-jaxrs". This security domain is configured to accept either a username and password (standard BASIC authentication) or a SAML Assertion (bearer token auth). If invoking the Atom

API directly, then typically BASIC authentication would be used. When invoking the Atom API from an application that has already authenticated the user in some way, then it is appropriate to use SAML. For example, the S-RAMP CLI application uses BASIC authentication when invoking the S-RAMP Atom API. The S-RAMP Browser (a web application) requires the user be authenticated into it, and thus is able to use SAML rather than propagate user credentials.

When using simple BASIC authentication, the security domain must be configured with a source of users and passwords. For example, in JBoss EAP this information can be configured by modifying the following files:

```
jboss-eap-6.x/standalone/configuration/application-users.properties
```

Alternative mechanisms can be used by making changes to the security domains configured in the JBoss configuration. For example:

```
jboss-eap-6.x/standalone/configuration/standalone.xml
```

The relevant section looks something like this:

```
<security-domain name="overlord-jaxrs" cache-type="default">
  <authentication>
    <login-module code="RealmDirect" flag="required">
      <module-option name="password-stacking" value="useFirstPass"/>
    </login-module>
  </authentication>
</security-domain>
```

For example, the RealmDirect (which passes through the authentication to the Application Realm) login module could be replaced with a LDAP login module.

7.2.5. Security (Authorization)

When accessing the S-RAMP Atom API, the authenticated user must have certain roles. Because the implementation leverages ModeShape as its persistence store by default, the authenticated user must have the following JAAS role, which is required by ModeShape:

```
admin.sramp
```

Additionally, the S-RAMP Atom API web application requires that the user has the following role:

```
overlorduser
```

7.2.6. Extending: Custom Deriver

As mentioned earlier in this guide. part of the S-RAMP specification is the concept of Derived content. This happens when an artifact of a certain type is added to the S-RAMP repository. The

server is responsible for creating relevant and interesting Derived Artifacts from it. For example, when an XML Schema (XSD) document is added to the repository, the server is responsible for automatically creating an artifact for every top level Element, Complex Type, Simple Type, and Attribute declaration found in the XSD.

The Overlord S-RAMP implementation includes Artifact Derivers for all of the logical models defined by the S-RAMP specification (e.g. WSDL, XSD, Policy). However, it also provides a mechanism that allows users to provide Artifact Derivers for their own artifact types. This is done by performing the following steps:

1. Write a custom Deriver Java class - it must implement **ArtifactDeriver**
2. Create a DeriverProvider (a class that implements **DeriverProvider**) - used to map artifact types to implementations of ArtifactDeriver
3. Provide a text file named **org.overlord.sramp.common.derived.DeriverProvider** in the following location: **META-INF/services**. The content of that file should simply be one line containing the fully qualified classname of the class defined in #2
4. Package everything up into a JAR and make it available either on the classpath or in an external directory configured by setting property **sramp.derivers.customDir**.

The Overlord S-RAMP distribution comes with an example of how to write and package a custom deriver - the demo is named **s-ramp-demos-custom-deriver**.

7.3. Client

As mentioned, the Overlord S-RAMP implementation also provides a Java client library that can be used to integrate with S-RAMP compliant servers. This section of the guide describes how to use this library.

7.3.1. Basic Usage

The S-RAMP client is a simple Java based client library and can be included in a Maven project by including the following pom.xml dependency:

```
<dependency>
  <groupId>org.overlord.sramp</groupId>
  <artifactId>s-ramp-client</artifactId>
  <version>${sramp.client.version}</version>
</dependency>
```

Once the library is included in your project, you can use the client by instantiating the **SrampAtomApiClient** class. Note that the client class supports pluggable authentication mechanisms, although BASIC auth is a simple matter of including the username and password upon construction of the client.

Please refer to the javadoc of that class for details, but here are some usage examples to help you get started (code simplified for readability):

***Upload an XSD document to S-RAMP*.**

```
SrampAtomApiClient client = new SrampAtomApiClient(urlToSramp);
String artifactFileName = getXSDArtifactName();
InputStream is = getXSDArtifactContentStream();
ArtifactType type = ArtifactType.XsdDocument();
BaseArtifactType  artifact  =  client.uploadArtifact(ArtifactType.XsdDocument(),
 is, artifactFileName);
```

***Create a custom artifact in S-RAMP (meta-data only, no file content)*.**

```
SrampAtomApiClient client = new SrampAtomApiClient(urlToSramp);
ExtendedArtifactType artifact = new ExtendedArtifactType();
artifact.setArtifactType(BaseArtifactEnum.EXTENDED_ARTIFACT_TYPE);
artifact.setExtendedType("MyArtifactType");
artifact.setName("My Test Artifact #1");
artifact.setDescription("Description of my test artifact.");
BaseArtifactType createdArtifact = client.createArtifact(artifact);
```

***Retrieve full meta-data for an XSD artifact by its UUID*.**

```
SrampAtomApiClient client = new SrampAtomApiClient(urlToSramp);
String uuid = getArtifactUUID();
BaseArtifactType                            metaData                            =
 client.getArtifactMetaData(ArtifactType.XsdDocument(), uuid);
```

***Retrieve artifact content*.**

```
SrampAtomApiClient client = new SrampAtomApiClient(urlToSramp);
String uuid = getArtifactUUID();
InputStream  content  =  client.getArtifactContent(ArtifactType.XsdDocument(),
 uuid);
```

***Query the S-RAMP repository (by artifact name)*.**

```
SrampAtomApiClient client = new SrampAtomApiClient(urlToSramp);
String artifactName = getArtifactName();
QueryResultSet rset = client.buildQuery("/s-ramp/xsd/XsdDocument[@name = ?]")
```

```
        .parameter(artifactName)
        .count(10)
        .query();
```

7.3.2. Extended Feature: Ontologies

Although the S-RAMP specification is silent on how the API should support the management of ontologies, the Overlord S-RAMP implementation provides an extension to the Atom based REST API to support this. Using any of the client's ontology related methods will work when communicating with the Overlord implementation of S-RAMP, but will likely fail when communicating with any other S-RAMP server.

The client supports adding, updating, and getting (both individual and a full list) ontologies from the S-RAMP repository.

7.3.3. Extended Feature: Auditing

The Overlord S-RAMP implementation also offers an extension to the Atom based REST API to get and set auditing information for artifacts in the repository.

7.3.4. Extending: Custom Expander

A special feature of the client is the ability to automatically expand archive style artifacts (artifacts that are JARs, WARs, ZIPs, etc). This feature is similar to how the server creates Derived content. The result is that certain files from the archive being uploaded as an S-RAMP artifact are extracted from the archive and also uploaded to the server. When this happens these "expanded" artifacts are added with an S-RAMP relationship (expandedFromDocument) that points to the archive artifact they were expanded from.

The Overlord S-RAMP implementation comes with a few built-in expanders (e.g. java archive, SwitchYard archive, etc). Additionally, custom expanders can be created and provided by implementing **ZipToSrampArchiveProvider**. In order to tell the client about the custom provider, put it in a JAR along with a file named:

META-INF/services/
org.overlord.sramp.atom.archive.expand.registry.ZipToSrampArchiveProvider

The contents of the file should be a single line with the fully qualified Java classname of the provider implementation.

Chapter 8. Overlord S-RAMP REST API Endpoints

The S-RAMP Atom API protocol binding does not dictate the format of the API endpoints. Clients must request the /servicedocument and then inspect the workspaces found therein. However, the Overlord implementation's endpoints conform to the notional syntax described in the S-RAMP specification's foundation document. The following table lists the endpoints available in the Overlord implementation:

Endpoint	Name
GET /s-ramp/servicedocument	*Get Service Document*
POST /s-ramp/{model}/{type}	*Publish Artifact*
PUT /s-ramp/{model}/{type}/{uuid}	*Update Artifact*
PUT /s-ramp/{model}/{type}/{uuid}/media	*Update Artifact Content*
GET /s-ramp/{model}/{type}/{uuid}	*Get Artifact*
GET /s-ramp/{model}/{type}/{uuid}/media	*Get Artifact Content*
DELETE /s-ramp/{model}/{type}/{uuid}	*Delete Artifact*
GET /s-ramp/{model}	*Get Artifact Feed (by model)*
GET /s-ramp/{model}/{type}	*Get Artifact Feed (by type)*
GET /s-ramp	*Query*
POST /s-ramp	*Query*
POST /s-ramp	*Batch Processing*
POST /s-ramp/ontology	*Add Ontology*
GET /s-ramp/ontology	*List Ontologies*
PUT /s-ramp/ontology/{uuid}	*Update Ontology*
GET /s-ramp/ontology/{uuid}	*Get Ontology*
DELETE /s-ramp/ontology/{uuid}	*Delete Ontology*
GET /s-ramp/audit/artifact/{artifactUuid}	*Get Artifact Audit History*
GET /s-ramp/audit/user/{username}	*Get User Audit History*
POST /s-ramp/audit/artifact/{artifactUuid}	*Add Artifact Audit Entry*
GET /s-ramp/audit/artifact/{artifactUuid}/{auditEntryUuid}	*Get Artifact Audit Entry*

8.1. API: Get Service Document

```
/s-ramp/servicedocument
```

Retrieves the service document.

HTTP Method	Request	Response
GET	N/A	Atom Service Document

The service document contains a workspace for each of the S-RAMP data models supported by the server.

Example Response

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<app:service xmlns:atom="http://www.w3.org/2005/Atom" xmlns:app="http://
www.w3.org/2007/app">
    <app:workspace>
        <atom:title>Core Model</atom:title>
        <app:collection href="http://example.org/s-ramp/core">
            <atom:title>Core Model Objects</atom:title>
            <app:accept>application/zip</app:accept>
            <app:categories fixed="yes">
                <atom:category label="Document" scheme="urn:x-s-
ramp:v1:type" term="Document"/>
                <atom:category label="XML Document" scheme="urn:x-s-
ramp:v1:type" term="XmlDocument"/>
            </app:categories>
        </app:collection>
        <app:collection href="http://example.org/s-ramp/core/Document">
            <atom:title>Documents</atom:title>
            <app:accept>application/octet-stream</app:accept>
            <app:categories fixed="yes">
                <atom:category label="Document" scheme="urn:x-s-
ramp:v1:type" term="Document"/>
            </app:categories>
        </app:collection>
        <app:collection href="http://example.org/s-ramp/core/XmlDocument">
            <atom:title>XML Documents</atom:title>
            <app:accept>application/xml</app:accept>
            <app:categories fixed="yes">
                <atom:category label="XML Document" scheme="urn:x-s-
ramp:v1:type" term="XmlDocument"/>
            </app:categories>
        </app:collection>
    </app:workspace>
</app:service>
```

Tip

The above example only includes the Core data model and thus the service document has a single workspace. The full service document would have multiple workspaces - one for each data model supported by the server.

8.2. API: Publish Artifact

Publishes a new artifact into the repository.

There are three ways this endpoint can be invoked, depending on the type of artifact being published:

- Document Style Artifact
- Non-Document Style Artifact
- Document Style Artifact with Meta Data

8.2.1. Publish a Document Style Artifact

```
  /s-ramp/{model}/{type}
```

HTTP Method	Request	Response
POST	Binary File	Atom Entry

Publishing a document style artifact is simply a matter of POSTing the binary content of the document to the appropriate endpoint.

Example Request

```
POST /s-ramp/core/Document HTTP/1.1

This is a simple text document, uploaded as an artifact
into S-RAMP.
```

Example Response

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<atom:entry xmlns:atom="http://www.w3.org/2005/Atom" xmlns:s-ramp="http://
docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
  xmlns:xlink="http://www.w3.org/1999/xlink" s-ramp:derived="false">
  <atom:title>test.txt</atom:title>
```

```
  <atom:link
    href="http://example.org/s-ramp/core/Document/05778de3-be85-4696-b5dc-
d889a27f1f6e/media"
    rel="alternate" type="text/plain" />
  <atom:link href="http://example.org/s-ramp/core/Document/05778de3-
be85-4696-b5dc-d889a27f1f6e"
    rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
  <atom:link
    href="http://example.org/s-ramp/core/Document/05778de3-be85-4696-b5dc-
d889a27f1f6e/media"
    rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;" />
  <atom:link href="http://example.org/s-ramp/core/Document/05778de3-
be85-4696-b5dc-d889a27f1f6e"
    rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
  <atom:category label="Document" scheme="x-s-ramp:2010:type"
 term="Document" />
  <atom:category label="Document" scheme="x-s-ramp:2010:model" term="core" /
>
  <atom:updated>2013-05-14T13:43:09.708-04:00</atom:updated>
  <atom:id>05778de3-be85-4696-b5dc-d889a27f1f6e</atom:id>
  <atom:published>2013-05-14T13:43:09.708-04:00</atom:published>
  <atom:author>
    <atom:name>ewittman</atom:name>
  </atom:author>
  <atom:content
    src="http://example.org/s-ramp/core/Document/05778de3-be85-4696-b5dc-
d889a27f1f6e/media"
    type="text" />
  <s-ramp:artifact>
    <s-ramp:Document artifactType="Document" contentSize="69"
 contentType="text/plain"
      createdBy="eric" createdTimestamp="2013-05-14T13:43:09.708-04:00"
 lastModifiedBy="eric"
      lastModifiedTimestamp="2013-05-14T13:43:09.708-04:00" name="test.txt"
 uuid="05778de3-be85-4696-b5dc-d889a27f1f6e" />
  </s-ramp:artifact>
</atom:entry>
```

8.2.2. Publish a Non-Document Style Artifact

```
  /s-ramp/{model}/{type}
```

HTTP Method	Request	Response
POST	Atom Entry	Atom Entry

Publishing a non-document style artifact requires an Atom Entry (which contains an *s-ramp:artifact* child element) to be POSTed to the appropriate endpoint. The appropriate endpoint is based on the desired artifact model and type.

Example Request

```
POST /s-ramp/ext/MyArtifact HTTP/1.1

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<atom:entry xmlns:atom="http://www.w3.org/2005/Atom" xmlns:s-ramp="http://
docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
  xmlns:xlink="http://www.w3.org/1999/xlink" s-ramp:derived="false">
  <atom:title>Example Artifact</atom:title>
  <s-ramp:artifact>
    <s-ramp:ExtendedArtifactType extendedType="MyArtifact"
      artifactType="ExtendedArtifactType" name="My Artifact One" />
  </s-ramp:artifact>
</atom:entry>
```

Example Response

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<atom:entry xmlns:s-ramp="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
 xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:atom="http://www.w3.org/2005/Atom" s-ramp:derived="false" s-
ramp:extendedType="MavenPom">
  <atom:title>pom.xml</atom:title>
  <atom:link href="http://example.org/s-ramp/ext/MavenPom/5f4cbf1e-
cafb-4479-8867-fc5df5f21867/media"
    rel="alternate" type="application/xml" />
  <atom:link href="http://example.org/s-ramp/ext/MavenPom/5f4cbf1e-
cafb-4479-8867-fc5df5f21867" rel="self"
    type="application/atom+xml;type=&quot;entry&quot;" />
  <atom:link href="http://example.org/s-ramp/ext/MavenPom/5f4cbf1e-
cafb-4479-8867-fc5df5f21867/media"
    rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;" />
  <atom:link href="http://example.org/s-ramp/ext/MavenPom/5f4cbf1e-
cafb-4479-8867-fc5df5f21867" rel="edit"
    type="application/atom+xml;type=&quot;entry&quot;" />
  <atom:category label="Extended Document" scheme="x-s-ramp:2010:type"
 term="MavenPom" />
  <atom:category label="Extended Document" scheme="x-s-ramp:2010:model"
 term="ext" />
  <atom:updated>2013-05-14T13:49:20.645-04:00</atom:updated>
  <atom:id>5f4cbf1e-cafb-4479-8867-fc5df5f21867</atom:id>
  <atom:published>2013-05-14T13:49:20.645-04:00</atom:published>
  <atom:author>
    <atom:name>ewittman</atom:name>
  </atom:author>
  <atom:content type="application/xml"
```

```
    src="http://example.org/s-ramp/ext/MavenPom/5f4cbf1e-cafb-4479-8867-
fc5df5f21867/media" />
  <s-ramp:artifact>
    <s-ramp:ExtendedDocument extendedType="MavenPom"
 contentType="application/xml"
      contentSize="4748" artifactType="ExtendedDocument" name="pom.xml"
 createdBy="eric"
      uuid="5f4cbf1e-cafb-4479-8867-fc5df5f21867"
 createdTimestamp="2013-05-14T13:49:20.645-04:00"
      lastModifiedTimestamp="2013-05-14T13:49:20.645-04:00"
 lastModifiedBy="eric"
      s-ramp:contentType="application/xml" s-ramp:contentSize="4748" />
  </s-ramp:artifact>
</atom:entry>
```

8.2.3. Publish a Document Style Artifact with Meta-Data

```
  /s-ramp/{model}/{type}
```

HTTP Method	Request	Response
POST	Multipart/Related	Atom Entry

Sometimes it is convenient to publish an artifact and update its meta-data in a single request. This can be done by POSTing a multipart/related request to the server at the appropriate endpoint. The first part in the request must be an Atom Entry (containing the meta-data being set), while the second part must be the binary content. The appropriate endpoint is based on the desired artifact model and type.

Example Request

```
POST /s-ramp/core/Document HTTP/1.1
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://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0">
  <title type="text">myfile.txt</title>
  <summary type="text">The description of my text file.</summary>
  <category term="Document" label="Document"
            scheme="urn:x-s-ramp:2013urn:x-s-ramp:2013:type" />
  <s-ramp:artifact xmlns:s-ramp="http://docs.oasis-open.org/s-ramp/ns/s-
ramp-v1.0"
```

```
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <s-ramp:Document name="myfile.txt" version="1.0"
                     description="The description of my text file." >
      <s-ramp:classifiedBy>
        http://example.org/ontologies/regions.owl/Maine
      </s-ramp:classifiedBy>
      <s-ramp:property>
        <propertyName>foo</propertyName>
        <propertyValue>pity him</propertyValue>
      </s-ramp:property>
    </s-ramp:Document>
  </s-ramp:artifact>
</entry>
--===============1605871705==
Content-Type: application/xml
MIME-Version: 1.0

This is a simple text document, uploaded as an artifact
into S-RAMP.
--===============1605871705==--
```

Example Response

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<atom:entry xmlns:atom="http://www.w3.org/2005/Atom" xmlns:s-ramp="http://
docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
  xmlns:xlink="http://www.w3.org/1999/xlink" s-ramp:derived="false">
  <atom:title>test.txt</atom:title>
  <atom:link
    href="http://example.org/s-ramp/core/Document/05778de3-be85-4696-b5dc-
d889a27f1f6e/media"
    rel="alternate" type="text/plain" />
  <atom:link href="http://example.org/s-ramp/core/Document/05778de3-
be85-4696-b5dc-d889a27f1f6e"
    rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
  <atom:link
    href="http://example.org/s-ramp/core/Document/05778de3-be85-4696-b5dc-
d889a27f1f6e/media"
    rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;" />
  <atom:link href="http://example.org/s-ramp/core/Document/05778de3-
be85-4696-b5dc-d889a27f1f6e"
    rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
  <atom:category label="Document" scheme="x-s-ramp:2010:type"
 term="Document" />
  <atom:category label="Document" scheme="x-s-ramp:2010:model" term="core" /
>
  <atom:updated>2013-05-14T13:43:09.708-04:00</atom:updated>
```

```
  <atom:id>05778de3-be85-4696-b5dc-d889a27f1f6e</atom:id>
  <atom:published>2013-05-14T13:43:09.708-04:00</atom:published>
  <atom:author>
    <atom:name>ewittman</atom:name>
  </atom:author>
  <atom:content
    src="http://example.org/s-ramp/core/Document/05778de3-be85-4696-b5dc-
d889a27f1f6e/media"
    type="text" />
  <s-ramp:artifact>
    <s-ramp:Document artifactType="Document" contentSize="69"
 contentType="text/plain"
      name="myfile.txt" uuid="05778de3-be85-4696-b5dc-d889a27f1f6e">
      description="The description of my text file." version="1.0"
      createdBy="eric" createdTimestamp="2013-05-14T13:43:09.708-04:00"
      lastModifiedBy="eric"
 lastModifiedTimestamp="2013-05-14T13:43:09.708-04:00"
      <s-ramp:classifiedBy>
        http://example.org/ontologies/regions.owl/Maine
      </s-ramp:classifiedBy>
      <s-ramp:property>
        <propertyName>foo</propertyName>
        <propertyValue>pity him</propertyValue>
      </s-ramp:property>
    </s-ramp:Document>
  </s-ramp:artifact>
</atom:entry>
```

8.3. API: Update Artifact

```
  /s-ramp/{model}/{type}/{uuid}
```

Updates an artifact's meta data.

HTTP Method	Request	Response
PUT	Atom Entry	N/A

This endpoint is used to update a single artifact's meta data, including core properties, custom properties, classifiers, and relationships. Typically the client should first retrieve the artifact (e.g. by invoking the Get Artifact endpoint), make changes to the artifact, then issue a PUT request to the Update Artifact endpoint.

Example Request

```
PUT /s-ramp/core/Document/098da465-2eae-49b7-8857-eb447f03ac02 HTTP/1.1

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

```
<atom:entry xmlns:s-ramp="http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
 xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:atom="http://www.w3.org/2005/Atom">
  <atom:title>pom.xml</atom:title>
  <atom:updated>2013-05-15T08:12:01.985-04:00</atom:updated>
  <atom:id>098da465-2eae-49b7-8857-eb447f03ac02</atom:id>
  <atom:published>2013-05-15T08:12:01.985-04:00</atom:published>
  <atom:author>
    <atom:name>ewittman</atom:name>
  </atom:author>
  <atom:summary>Sample description of my document.</atom:summary>
  <s-ramp:artifact>
    <s-ramp:Document contentType="text/plain" contentSize="4748"
 artifactType="Document"
      name="myfile.txt" description="Sample description of my document."
 createdBy="ewittman"
      uuid="098da465-2eae-49b7-8857-eb447f03ac02"
 createdTimestamp="2013-05-15T08:12:01.985-04:00"
      lastModifiedTimestamp="2013-05-15T08:12:01.985-04:00"
 lastModifiedBy="ewittman">
      <s-ramp:property>
        <s-ramp:propertyName>foo</s-ramp:propertyName>
        <s-ramp:propertyValue>bar</s-ramp:propertyValue>
      </s-ramp:property>
    </s-ramp:Document>
  </s-ramp:artifact>
</atom:entry>
```

8.4. API: Update Artifact Content

```
  /s-ramp/{model}/{type}/{uuid}/media
```

Updates an artifact's content.

HTTP Method	Request	Response
PUT	Binary Content	N/A

This endpoint is used to update a single artifact's content, regardless if the artifact is a text document or some sort of binary. The body of the request should be the new binary content of the artifact.

Example Request

```
PUT /s-ramp/core/Document/0f6f9b6b-9952-4059-ab70-7ee3442ddcf0/media
 HTTP/1.1

Some file content goes here.
```

8.5. API: Get Artifact

```
  /s-ramp/{model}/{type}/{uuid}
```

Retrieves an artifact's meta data.

HTTP Method	Request	Response
GET	N/A	Atom Entry (full)

This endpoint is used to retrieve the full meta-data for a single artifact in the repository. The data is returned wrapped up in an Atom Entry document. The Atom Entry will contain an extended XML element containing the S-RAMP artifact data.

Example Request

```
PUT /s-ramp/xsd/ComplexTypeDeclaration/0104e848-fe91-4d93-a307-fb69ec9fd638
 HTTP/1.1
```

Example Response

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<atom:entry xmlns:atom="http://www.w3.org/2005/Atom" xmlns:s-ramp="http://
docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0" xmlns:xlink="http://
www.w3.org/1999/xlink" s-ramp:derived="true">
<atom:title>submitOrderResponseType</atom:title>
<atom:link href="http://localhost:8080/s-ramp-server/s-ramp/xsd/
ComplexTypeDeclaration/0104e848-fe91-4d93-a307-fb69ec9fd638" rel="self"
 type="application/atom+xml;type=&quot;entry&quot;"/>
<atom:link href="http://localhost:8080/s-ramp-server/s-ramp/xsd/
ComplexTypeDeclaration/0104e848-fe91-4d93-a307-fb69ec9fd638/media"
 rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;"/>
<atom:link href="http://localhost:8080/s-ramp-server/s-ramp/xsd/
ComplexTypeDeclaration/0104e848-fe91-4d93-a307-fb69ec9fd638" rel="edit"
 type="application/atom+xml;type=&quot;entry&quot;"/>
<atom:category label="XML Schema Complex Type Declaration" scheme="x-s-
ramp:2010:type" term="ComplexTypeDeclaration"/>
<atom:category label="XML Schema Complex Type Declaration" scheme="x-s-
ramp:2010:model" term="xsd"/>
<atom:updated>2013-07-22T12:19:23.554-04:00</atom:updated>
<atom:id>0104e848-fe91-4d93-a307-fb69ec9fd638</atom:id>
<atom:published>2013-07-22T12:19:22.630-04:00</atom:published>
<atom:author>
<atom:name>eric</atom:name>
</atom:author>
<s-ramp:artifact>
```

```
<s-ramp:ComplexTypeDeclaration artifactType="ComplexTypeDeclaration"
 createdBy="eric" createdTimestamp="2013-07-22T12:19:22.630-04:00"
 lastModifiedBy="eric" lastModifiedTimestamp="2013-07-22T12:19:23.554-04:00"
 name="submitOrderResponseType" namespace="urn:switchyard-quickstart-
demo:multiapp:1.0" uuid="0104e848-fe91-4d93-a307-fb69ec9fd638">
<s-ramp:relatedDocument artifactType="XsdDocument">fe7b72ec-5ad9-436c-
b7aa-0391da5cc972</s-ramp:relatedDocument>
</s-ramp:ComplexTypeDeclaration>
</s-ramp:artifact>
</atom:entry>
```

8.6. API: Get Artifact Content

```
  /s-ramp/{model}/{type}/{uuid}/media
```

Retrieves an artifact's content.

HTTP Method	Request	Response
GET	N/A	Binary artifact content

This endpoint is used to retrieve the full content of a single artifact in the repository. If the artifact is not a Document style artifact, this call will fail. Otherwise it will return the full artifact content. For example, if the artifact is a PdfDocument, then this call will return the PDF file.

Example Request

```
GET /s-ramp/core/Document/0f6f9b6b-9952-4059-ab70-7ee3442ddcf0/media
 HTTP/1.1
```

Example Response

```
HTTP/1.1 200 OK

Artifact/file content returned here.
```

8.7. API: Delete Artifact

```
  /s-ramp/{model}/{type}/{uuid}
```

Deletes an artifact.

HTTP Method	Request	Response
DELETE	N/A	N/A

This endpoint is used to delete a single artifact from the repository. If the artifact does not exist or is a derived artifact, then this will fail. This might also fail if other artifacts have relationships with it. Otherwise this artifact (and all of its derived artifacts) will be deleted.

Example Request

```
DELETE /s-ramp/core/Document/0f6f9b6b-9952-4059-ab70-7ee3442ddcf0 HTTP/1.1
```

8.8. API: Get Artifact Feed (by model)

```
  /s-ramp/{model}
```

Retrieves an Atom feed of all artifacts in a given model.

HTTP Method	Request	Response
GET	N/A	Atom Feed

This endpoint is used to retrieve an Atom feed of all artifacts in a single S-RAMP model. The feed contains Atom summary Entries - one for each artifact in the feed. Standard paging options apply.

Example Request

```
GET /s-ramp/core HTTP/1.1
```

Example Response

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<atom:feed xmlns:atom="http://www.w3.org/2005/Atom" xmlns:s-ramp="http://
docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
  s-ramp:itemsPerPage="100" s-ramp:provider="JBoss Overlord" s-
ramp:startIndex="0" s-ramp:totalResults="5">
  <atom:title>S-RAMP Feed</atom:title>
  <atom:subtitle>Ad Hoc query feed</atom:subtitle>
  <atom:updated>2013-07-22T12:50:16.605-04:00</atom:updated>
  <atom:id>1647967f-a6f4-4e9c-82d3-ac422fb152f3</atom:id>
  <atom:author>
    <atom:name>anonymous</atom:name>
  </atom:author>
  <atom:entry s-ramp:derived="false">
    <atom:title>sramp.sh</atom:title>
    <atom:link href="http://localhost:8080/s-ramp/core/
Document/0f6f9b6b-9952-4059-ab70-7ee3442ddcf0/media"
      rel="alternate" type="application/x-sh" />
    <atom:link href="http://localhost:8080/s-ramp/core/
Document/0f6f9b6b-9952-4059-ab70-7ee3442ddcf0"
      rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
Document/0f6f9b6b-9952-4059-ab70-7ee3442ddcf0/media"
      rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;" />
```

```
    <atom:link href="http://localhost:8080/s-ramp/core/
Document/0f6f9b6b-9952-4059-ab70-7ee3442ddcf0"
      rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:category label="Document" scheme="x-s-ramp:2010:type"
 term="Document" />
    <atom:category label="Document" scheme="x-s-ramp:2010:model"
 term="core" />
    <atom:updated>2013-07-22T12:22:01.953-04:00</atom:updated>
    <atom:id>0f6f9b6b-9952-4059-ab70-7ee3442ddcf0</atom:id>
    <atom:published>2013-07-22T12:21:49.499-04:00</atom:published>
    <atom:author>
      <atom:name>eric</atom:name>
    </atom:author>
    <atom:content src="http://localhost:8080/s-ramp/core/
Document/0f6f9b6b-9952-4059-ab70-7ee3442ddcf0/media"
      type="application/x-sh" />
  </atom:entry>
  <atom:entry s-ramp:derived="false">
    <atom:title>beans.xml</atom:title>
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/20474032-9536-4cef-812c-4fea432fdebd/media"
      rel="alternate" type="application/xml" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/20474032-9536-4cef-812c-4fea432fdebd"
      rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/20474032-9536-4cef-812c-4fea432fdebd/media"
      rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/20474032-9536-4cef-812c-4fea432fdebd"
      rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:model"
 term="core" />
    <atom:updated>2013-07-22T12:19:27.660-04:00</atom:updated>
    <atom:id>20474032-9536-4cef-812c-4fea432fdebd</atom:id>
    <atom:published>2013-07-22T12:19:27.644-04:00</atom:published>
    <atom:author>
      <atom:name>eric</atom:name>
    </atom:author>
    <atom:content src="http://localhost:8080/s-ramp/core/
XmlDocument/20474032-9536-4cef-812c-4fea432fdebd/media"
      type="application/xml" />
  </atom:entry>
  <atom:entry s-ramp:derived="false">
    <atom:title>forge.xml</atom:title>
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/2c21a9d3-0d09-41d8-8783-f3e795d8690d/media"
```

```
      rel="alternate" type="application/xml" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/2c21a9d3-0d09-41d8-8783-f3e795d8690d"
      rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/2c21a9d3-0d09-41d8-8783-f3e795d8690d/media"
      rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/2c21a9d3-0d09-41d8-8783-f3e795d8690d"
      rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:model"
 term="core" />
    <atom:updated>2013-07-22T12:19:25.576-04:00</atom:updated>
    <atom:id>2c21a9d3-0d09-41d8-8783-f3e795d8690d</atom:id>
    <atom:published>2013-07-22T12:19:25.555-04:00</atom:published>
    <atom:author>
      <atom:name>eric</atom:name>
    </atom:author>
    <atom:content src="http://localhost:8080/s-ramp/core/
XmlDocument/2c21a9d3-0d09-41d8-8783-f3e795d8690d/media"
      type="application/xml" />
  </atom:entry>
  <atom:entry s-ramp:derived="false">
    <atom:title>route.xml</atom:title>
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/5b653bfe-4f58-451e-b738-394e61c0c5f9/media"
      rel="alternate" type="application/xml" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/5b653bfe-4f58-451e-b738-394e61c0c5f9"
      rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/5b653bfe-4f58-451e-b738-394e61c0c5f9/media"
      rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/5b653bfe-4f58-451e-b738-394e61c0c5f9"
      rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:model"
 term="core" />
    <atom:updated>2013-07-22T12:19:25.602-04:00</atom:updated>
    <atom:id>5b653bfe-4f58-451e-b738-394e61c0c5f9</atom:id>
    <atom:published>2013-07-22T12:19:25.577-04:00</atom:published>
    <atom:author>
      <atom:name>eric</atom:name>
    </atom:author>
```

```
    <atom:content src="http://localhost:8080/s-ramp/core/
XmlDocument/5b653bfe-4f58-451e-b738-394e61c0c5f9/media"
      type="application/xml" />
  </atom:entry>
  <atom:entry s-ramp:derived="false">
    <atom:title>beans.xml</atom:title>
    <atom:link href="http://localhost:8080/s-ramp/core/XmlDocument/
a3f9d4d7-0f95-4219-85f6-84df445ef270/media"
      rel="alternate" type="application/xml" />
    <atom:link href="http://localhost:8080/s-ramp/core/XmlDocument/
a3f9d4d7-0f95-4219-85f6-84df445ef270"
      rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/XmlDocument/
a3f9d4d7-0f95-4219-85f6-84df445ef270/media"
      rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/XmlDocument/
a3f9d4d7-0f95-4219-85f6-84df445ef270"
      rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:model"
 term="core" />
    <atom:updated>2013-07-22T12:19:21.498-04:00</atom:updated>
    <atom:id>a3f9d4d7-0f95-4219-85f6-84df445ef270</atom:id>
    <atom:published>2013-07-22T12:19:21.376-04:00</atom:published>
    <atom:author>
      <atom:name>eric</atom:name>
    </atom:author>
    <atom:content src="http://localhost:8080/s-ramp/core/XmlDocument/
a3f9d4d7-0f95-4219-85f6-84df445ef270/media"
      type="application/xml" />
  </atom:entry>
</atom:feed>
```

8.9. API: Get Artifact Feed (by type)

```
  /s-ramp/{model}/{type}
```

Retrieves an Atom feed of all artifacts of a specific type.

HTTP Method	Request	Response
GET	N/A	Atom Feed

This endpoint is used to retrieve an Atom feed of all artifacts of a specific S-RAMP type. The feed contains Atom summary Entries - one for each artifact in the feed. Standard paging options (as query params) apply.

Example Request

```
GET /s-ramp/core/Document HTTP/1.1
```

Example Response

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<atom:feed xmlns:atom="http://www.w3.org/2005/Atom" xmlns:s-ramp="http://
docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
  s-ramp:itemsPerPage="100" s-ramp:provider="JBoss Overlord" s-
ramp:startIndex="0" s-ramp:totalResults="5">
  <atom:title>S-RAMP Feed</atom:title>
  <atom:subtitle>Ad Hoc query feed</atom:subtitle>
  <atom:updated>2013-07-22T12:50:16.605-04:00</atom:updated>
  <atom:id>1647967f-a6f4-4e9c-82d3-ac422fb152f3</atom:id>
  <atom:author>
    <atom:name>anonymous</atom:name>
  </atom:author>
  <atom:entry s-ramp:derived="false">
    <atom:title>sramp.sh</atom:title>
    <atom:link href="http://localhost:8080/s-ramp/core/
Document/0f6f9b6b-9952-4059-ab70-7ee3442ddcf0/media"
      rel="alternate" type="application/x-sh" />
    <atom:link href="http://localhost:8080/s-ramp/core/
Document/0f6f9b6b-9952-4059-ab70-7ee3442ddcf0"
      rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
Document/0f6f9b6b-9952-4059-ab70-7ee3442ddcf0/media"
      rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
Document/0f6f9b6b-9952-4059-ab70-7ee3442ddcf0"
      rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:category label="Document" scheme="x-s-ramp:2010:type"
 term="Document" />
    <atom:category label="Document" scheme="x-s-ramp:2010:model"
 term="core" />
    <atom:updated>2013-07-22T12:22:01.953-04:00</atom:updated>
    <atom:id>0f6f9b6b-9952-4059-ab70-7ee3442ddcf0</atom:id>
    <atom:published>2013-07-22T12:21:49.499-04:00</atom:published>
    <atom:author>
      <atom:name>eric</atom:name>
    </atom:author>
    <atom:content src="http://localhost:8080/s-ramp/core/
Document/0f6f9b6b-9952-4059-ab70-7ee3442ddcf0/media"
      type="application/x-sh" />
  </atom:entry>
  <atom:entry s-ramp:derived="false">
    <atom:title>beans.xml</atom:title>
```

```
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/20474032-9536-4cef-812c-4fea432fdebd/media"
      rel="alternate" type="application/xml" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/20474032-9536-4cef-812c-4fea432fdebd"
      rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/20474032-9536-4cef-812c-4fea432fdebd/media"
      rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/20474032-9536-4cef-812c-4fea432fdebd"
      rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:model"
 term="core" />
    <atom:updated>2013-07-22T12:19:27.660-04:00</atom:updated>
    <atom:id>20474032-9536-4cef-812c-4fea432fdebd</atom:id>
    <atom:published>2013-07-22T12:19:27.644-04:00</atom:published>
    <atom:author>
      <atom:name>eric</atom:name>
    </atom:author>
    <atom:content src="http://localhost:8080/s-ramp/core/
XmlDocument/20474032-9536-4cef-812c-4fea432fdebd/media"
      type="application/xml" />
  </atom:entry>
  <atom:entry s-ramp:derived="false">
    <atom:title>forge.xml</atom:title>
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/2c21a9d3-0d09-41d8-8783-f3e795d8690d/media"
      rel="alternate" type="application/xml" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/2c21a9d3-0d09-41d8-8783-f3e795d8690d"
      rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/2c21a9d3-0d09-41d8-8783-f3e795d8690d/media"
      rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/2c21a9d3-0d09-41d8-8783-f3e795d8690d"
      rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:model"
 term="core" />
    <atom:updated>2013-07-22T12:19:25.576-04:00</atom:updated>
    <atom:id>2c21a9d3-0d09-41d8-8783-f3e795d8690d</atom:id>
    <atom:published>2013-07-22T12:19:25.555-04:00</atom:published>
    <atom:author>
      <atom:name>eric</atom:name>
```

```
    </atom:author>
    <atom:content src="http://localhost:8080/s-ramp/core/
XmlDocument/2c21a9d3-0d09-41d8-8783-f3e795d8690d/media"
      type="application/xml" />
  </atom:entry>
  <atom:entry s-ramp:derived="false">
    <atom:title>route.xml</atom:title>
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/5b653bfe-4f58-451e-b738-394e61c0c5f9/media"
      rel="alternate" type="application/xml" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/5b653bfe-4f58-451e-b738-394e61c0c5f9"
      rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/5b653bfe-4f58-451e-b738-394e61c0c5f9/media"
      rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/5b653bfe-4f58-451e-b738-394e61c0c5f9"
      rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:model"
 term="core" />
    <atom:updated>2013-07-22T12:19:25.602-04:00</atom:updated>
    <atom:id>5b653bfe-4f58-451e-b738-394e61c0c5f9</atom:id>
    <atom:published>2013-07-22T12:19:25.577-04:00</atom:published>
    <atom:author>
      <atom:name>eric</atom:name>
    </atom:author>
    <atom:content src="http://localhost:8080/s-ramp/core/
XmlDocument/5b653bfe-4f58-451e-b738-394e61c0c5f9/media"
      type="application/xml" />
  </atom:entry>
  <atom:entry s-ramp:derived="false">
    <atom:title>beans.xml</atom:title>
    <atom:link href="http://localhost:8080/s-ramp/core/XmlDocument/
a3f9d4d7-0f95-4219-85f6-84df445ef270/media"
      rel="alternate" type="application/xml" />
    <atom:link href="http://localhost:8080/s-ramp/core/XmlDocument/
a3f9d4d7-0f95-4219-85f6-84df445ef270"
      rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/XmlDocument/
a3f9d4d7-0f95-4219-85f6-84df445ef270/media"
      rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/XmlDocument/
a3f9d4d7-0f95-4219-85f6-84df445ef270"
      rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:type"
 term="XmlDocument" />
```

```
    <atom:category label="XML Document" scheme="x-s-ramp:2010:model"
 term="core" />
    <atom:updated>2013-07-22T12:19:21.498-04:00</atom:updated>
    <atom:id>a3f9d4d7-0f95-4219-85f6-84df445ef270</atom:id>
    <atom:published>2013-07-22T12:19:21.376-04:00</atom:published>
    <atom:author>
      <atom:name>eric</atom:name>
    </atom:author>
    <atom:content src="http://localhost:8080/s-ramp/core/XmlDocument/
a3f9d4d7-0f95-4219-85f6-84df445ef270/media"
      type="application/xml" />
  </atom:entry>
</atom:feed>
```

8.10. API: Query

```
  /s-ramp
```

Performs an S-RAMP query and returns an Atom feed containing the matching artifacts.

HTTP Method	Request	Response
GET	N/A	Atom Feed

This endpoint is used to perform an S-RAMP query and return an Atom Feed of the results. Ordering and paging is supported. The query and other parameters are passed as query params in the request. The feed contains Atom summary Entries - one for each artifact in the feed.

Example Request

```
GET /s-ramp?query=/s-ramp/core/Document HTTP/1.1
```

Example Response

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<atom:feed xmlns:atom="http://www.w3.org/2005/Atom" xmlns:s-ramp="http://
docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
  s-ramp:itemsPerPage="100" s-ramp:provider="JBoss Overlord" s-
ramp:startIndex="0" s-ramp:totalResults="5">
  <atom:title>S-RAMP Feed</atom:title>
  <atom:subtitle>Ad Hoc query feed</atom:subtitle>
  <atom:updated>2013-07-22T12:50:16.605-04:00</atom:updated>
  <atom:id>1647967f-a6f4-4e9c-82d3-ac422fb152f3</atom:id>
  <atom:author>
    <atom:name>anonymous</atom:name>
  </atom:author>
  <atom:entry s-ramp:derived="false">
```

```
    <atom:title>sramp.sh</atom:title>
    <atom:link href="http://localhost:8080/s-ramp/core/
Document/0f6f9b6b-9952-4059-ab70-7ee3442ddcf0/media"
      rel="alternate" type="application/x-sh" />
    <atom:link href="http://localhost:8080/s-ramp/core/
Document/0f6f9b6b-9952-4059-ab70-7ee3442ddcf0"
      rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
Document/0f6f9b6b-9952-4059-ab70-7ee3442ddcf0/media"
      rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
Document/0f6f9b6b-9952-4059-ab70-7ee3442ddcf0"
      rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:category label="Document" scheme="x-s-ramp:2010:type"
 term="Document" />
    <atom:category label="Document" scheme="x-s-ramp:2010:model"
 term="core" />
    <atom:updated>2013-07-22T12:22:01.953-04:00</atom:updated>
    <atom:id>0f6f9b6b-9952-4059-ab70-7ee3442ddcf0</atom:id>
    <atom:published>2013-07-22T12:21:49.499-04:00</atom:published>
    <atom:author>
      <atom:name>eric</atom:name>
    </atom:author>
    <atom:content src="http://localhost:8080/s-ramp/core/
Document/0f6f9b6b-9952-4059-ab70-7ee3442ddcf0/media"
      type="application/x-sh" />
  </atom:entry>
  <atom:entry s-ramp:derived="false">
    <atom:title>beans.xml</atom:title>
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/20474032-9536-4cef-812c-4fea432fdebd/media"
      rel="alternate" type="application/xml" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/20474032-9536-4cef-812c-4fea432fdebd"
      rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/20474032-9536-4cef-812c-4fea432fdebd/media"
      rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/20474032-9536-4cef-812c-4fea432fdebd"
      rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:model"
 term="core" />
    <atom:updated>2013-07-22T12:19:27.660-04:00</atom:updated>
    <atom:id>20474032-9536-4cef-812c-4fea432fdebd</atom:id>
    <atom:published>2013-07-22T12:19:27.644-04:00</atom:published>
    <atom:author>
```

```
      <atom:name>eric</atom:name>
    </atom:author>
    <atom:content src="http://localhost:8080/s-ramp/core/
XmlDocument/20474032-9536-4cef-812c-4fea432fdebd/media"
      type="application/xml" />
  </atom:entry>
  <atom:entry s-ramp:derived="false">
    <atom:title>forge.xml</atom:title>
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/2c21a9d3-0d09-41d8-8783-f3e795d8690d/media"
      rel="alternate" type="application/xml" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/2c21a9d3-0d09-41d8-8783-f3e795d8690d"
      rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/2c21a9d3-0d09-41d8-8783-f3e795d8690d/media"
      rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/2c21a9d3-0d09-41d8-8783-f3e795d8690d"
      rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:model"
 term="core" />
    <atom:updated>2013-07-22T12:19:25.576-04:00</atom:updated>
    <atom:id>2c21a9d3-0d09-41d8-8783-f3e795d8690d</atom:id>
    <atom:published>2013-07-22T12:19:25.555-04:00</atom:published>
    <atom:author>
      <atom:name>eric</atom:name>
    </atom:author>
    <atom:content src="http://localhost:8080/s-ramp/core/
XmlDocument/2c21a9d3-0d09-41d8-8783-f3e795d8690d/media"
      type="application/xml" />
  </atom:entry>
  <atom:entry s-ramp:derived="false">
    <atom:title>route.xml</atom:title>
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/5b653bfe-4f58-451e-b738-394e61c0c5f9/media"
      rel="alternate" type="application/xml" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/5b653bfe-4f58-451e-b738-394e61c0c5f9"
      rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/5b653bfe-4f58-451e-b738-394e61c0c5f9/media"
      rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/5b653bfe-4f58-451e-b738-394e61c0c5f9"
      rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
```

```
    <atom:category label="XML Document" scheme="x-s-ramp:2010:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:model"
 term="core" />
    <atom:updated>2013-07-22T12:19:25.602-04:00</atom:updated>
    <atom:id>5b653bfe-4f58-451e-b738-394e61c0c5f9</atom:id>
    <atom:published>2013-07-22T12:19:25.577-04:00</atom:published>
    <atom:author>
      <atom:name>eric</atom:name>
    </atom:author>
    <atom:content src="http://localhost:8080/s-ramp/core/
XmlDocument/5b653bfe-4f58-451e-b738-394e61c0c5f9/media"
      type="application/xml" />
  </atom:entry>
  <atom:entry s-ramp:derived="false">
    <atom:title>beans.xml</atom:title>
    <atom:link href="http://localhost:8080/s-ramp/core/XmlDocument/
a3f9d4d7-0f95-4219-85f6-84df445ef270/media"
      rel="alternate" type="application/xml" />
    <atom:link href="http://localhost:8080/s-ramp/core/XmlDocument/
a3f9d4d7-0f95-4219-85f6-84df445ef270"
      rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/XmlDocument/
a3f9d4d7-0f95-4219-85f6-84df445ef270/media"
      rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/XmlDocument/
a3f9d4d7-0f95-4219-85f6-84df445ef270"
      rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:model"
 term="core" />
    <atom:updated>2013-07-22T12:19:21.498-04:00</atom:updated>
    <atom:id>a3f9d4d7-0f95-4219-85f6-84df445ef270</atom:id>
    <atom:published>2013-07-22T12:19:21.376-04:00</atom:published>
    <atom:author>
      <atom:name>eric</atom:name>
    </atom:author>
    <atom:content src="http://localhost:8080/s-ramp/core/XmlDocument/
a3f9d4d7-0f95-4219-85f6-84df445ef270/media"
      type="application/xml" />
  </atom:entry>
</atom:feed>
```

8.11. API: Query

```
  /s-ramp
```

Performs an S-RAMP query and returns an Atom feed containing the matching artifacts.

HTTP Method	Request	Response
POST	FormData	Atom Feed

This endpoint is used to perform an S-RAMP query and return an Atom Feed of the results. Ordering and paging is supported. The query and other parameters are passed as form data params in the request body. The feed contains Atom summary Entries - one for each artifact in the feed.

Example Request

```
POST /s-ramp HTTP/1.1

--ac709f11-bfc5-48df-8918-e58b254d0490
Content-Disposition: form-data; name="query"
Content-Type: text/plain

core/Document
--ac709f11-bfc5-48df-8918-e58b254d0490
Content-Disposition: form-data; name="startIndex"
Content-Type: text/plain

0
--ac709f11-bfc5-48df-8918-e58b254d0490
Content-Disposition: form-data; name="count"
Content-Type: text/plain

100
--ac709f11-bfc5-48df-8918-e58b254d0490
Content-Disposition: form-data; name="orderBy"
Content-Type: text/plain

uuid
--ac709f11-bfc5-48df-8918-e58b254d0490
Content-Disposition: form-data; name="ascending"
Content-Type: text/plain

true
--ac709f11-bfc5-48df-8918-e58b254d0490--
```

Example Response

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<atom:feed xmlns:atom="http://www.w3.org/2005/Atom" xmlns:s-ramp="http://
docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
```

```
  s-ramp:itemsPerPage="100" s-ramp:provider="JBoss Overlord" s-
ramp:startIndex="0" s-ramp:totalResults="5">
  <atom:title>S-RAMP Feed</atom:title>
  <atom:subtitle>Ad Hoc query feed</atom:subtitle>
  <atom:updated>2013-07-22T12:50:16.605-04:00</atom:updated>
  <atom:id>1647967f-a6f4-4e9c-82d3-ac422fb152f3</atom:id>
  <atom:author>
    <atom:name>anonymous</atom:name>
  </atom:author>
  <atom:entry s-ramp:derived="false">
    <atom:title>sramp.sh</atom:title>
    <atom:link href="http://localhost:8080/s-ramp/core/
Document/0f6f9b6b-9952-4059-ab70-7ee3442ddcf0/media"
      rel="alternate" type="application/x-sh" />
    <atom:link href="http://localhost:8080/s-ramp/core/
Document/0f6f9b6b-9952-4059-ab70-7ee3442ddcf0"
      rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
Document/0f6f9b6b-9952-4059-ab70-7ee3442ddcf0/media"
      rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
Document/0f6f9b6b-9952-4059-ab70-7ee3442ddcf0"
      rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:category label="Document" scheme="x-s-ramp:2010:type"
 term="Document" />
    <atom:category label="Document" scheme="x-s-ramp:2010:model"
 term="core" />
    <atom:updated>2013-07-22T12:22:01.953-04:00</atom:updated>
    <atom:id>0f6f9b6b-9952-4059-ab70-7ee3442ddcf0</atom:id>
    <atom:published>2013-07-22T12:21:49.499-04:00</atom:published>
    <atom:author>
      <atom:name>eric</atom:name>
    </atom:author>
    <atom:content src="http://localhost:8080/s-ramp/core/
Document/0f6f9b6b-9952-4059-ab70-7ee3442ddcf0/media"
      type="application/x-sh" />
  </atom:entry>
  <atom:entry s-ramp:derived="false">
    <atom:title>beans.xml</atom:title>
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/20474032-9536-4cef-812c-4fea432fdebd/media"
      rel="alternate" type="application/xml" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/20474032-9536-4cef-812c-4fea432fdebd"
      rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/20474032-9536-4cef-812c-4fea432fdebd/media"
      rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;" />
```

```
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/20474032-9536-4cef-812c-4fea432fdebd"
      rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:model"
 term="core" />
    <atom:updated>2013-07-22T12:19:27.660-04:00</atom:updated>
    <atom:id>20474032-9536-4cef-812c-4fea432fdebd</atom:id>
    <atom:published>2013-07-22T12:19:27.644-04:00</atom:published>
    <atom:author>
      <atom:name>eric</atom:name>
    </atom:author>
    <atom:content src="http://localhost:8080/s-ramp/core/
XmlDocument/20474032-9536-4cef-812c-4fea432fdebd/media"
      type="application/xml" />
  </atom:entry>
  <atom:entry s-ramp:derived="false">
    <atom:title>forge.xml</atom:title>
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/2c21a9d3-0d09-41d8-8783-f3e795d8690d/media"
      rel="alternate" type="application/xml" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/2c21a9d3-0d09-41d8-8783-f3e795d8690d"
      rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/2c21a9d3-0d09-41d8-8783-f3e795d8690d/media"
      rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/2c21a9d3-0d09-41d8-8783-f3e795d8690d"
      rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:model"
 term="core" />
    <atom:updated>2013-07-22T12:19:25.576-04:00</atom:updated>
    <atom:id>2c21a9d3-0d09-41d8-8783-f3e795d8690d</atom:id>
    <atom:published>2013-07-22T12:19:25.555-04:00</atom:published>
    <atom:author>
      <atom:name>eric</atom:name>
    </atom:author>
    <atom:content src="http://localhost:8080/s-ramp/core/
XmlDocument/2c21a9d3-0d09-41d8-8783-f3e795d8690d/media"
      type="application/xml" />
  </atom:entry>
  <atom:entry s-ramp:derived="false">
    <atom:title>route.xml</atom:title>
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/5b653bfe-4f58-451e-b738-394e61c0c5f9/media"
```

```
      rel="alternate" type="application/xml" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/5b653bfe-4f58-451e-b738-394e61c0c5f9"
      rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/5b653bfe-4f58-451e-b738-394e61c0c5f9/media"
      rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/
XmlDocument/5b653bfe-4f58-451e-b738-394e61c0c5f9"
      rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:model"
 term="core" />
    <atom:updated>2013-07-22T12:19:25.602-04:00</atom:updated>
    <atom:id>5b653bfe-4f58-451e-b738-394e61c0c5f9</atom:id>
    <atom:published>2013-07-22T12:19:25.577-04:00</atom:published>
    <atom:author>
      <atom:name>eric</atom:name>
    </atom:author>
    <atom:content src="http://localhost:8080/s-ramp/core/
XmlDocument/5b653bfe-4f58-451e-b738-394e61c0c5f9/media"
      type="application/xml" />
  </atom:entry>
  <atom:entry s-ramp:derived="false">
    <atom:title>beans.xml</atom:title>
    <atom:link href="http://localhost:8080/s-ramp/core/XmlDocument/
a3f9d4d7-0f95-4219-85f6-84df445ef270/media"
      rel="alternate" type="application/xml" />
    <atom:link href="http://localhost:8080/s-ramp/core/XmlDocument/
a3f9d4d7-0f95-4219-85f6-84df445ef270"
      rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/XmlDocument/
a3f9d4d7-0f95-4219-85f6-84df445ef270/media"
      rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link href="http://localhost:8080/s-ramp/core/XmlDocument/
a3f9d4d7-0f95-4219-85f6-84df445ef270"
      rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2010:model"
 term="core" />
    <atom:updated>2013-07-22T12:19:21.498-04:00</atom:updated>
    <atom:id>a3f9d4d7-0f95-4219-85f6-84df445ef270</atom:id>
    <atom:published>2013-07-22T12:19:21.376-04:00</atom:published>
    <atom:author>
      <atom:name>eric</atom:name>
    </atom:author>
```

```
    <atom:content src="http://localhost:8080/s-ramp/core/XmlDocument/
a3f9d4d7-0f95-4219-85f6-84df445ef270/media"
      type="application/xml" />
  </atom:entry>
</atom:feed>
```

8.12. API: Batch Processing

```
  /s-ramp
```

Performs an S-RAMP query and returns an Atom feed containing the matching artifacts.

HTTP Method	Request	Response
POST	multipart/form-data	Atom Feed

This endpoint is used to perform an S-RAMP query and return an Atom Feed of the results. Ordering and paging is supported. The query and other parameters are passed as form data params in the request body. The feed contains Atom summary Entries - one for each artifact in the feed.

Example Request

```
POST XX_TBD_XX HTTP/1.1
```

Example Response

```
HTTP/1.1 200 OK

XX_TBD_XX
```

8.13. API: Add Ontology

```
  /s-ramp/ontology
```

Adds a new ontology (*.owl file) to the repository. This allows artifacts to be classified using the classes defined in the ontology.

HTTP Method	Request	Response
POST	application/rdf+xml	Atom Entry

This endpoint is used to add an ontology to the repository. The body of the request must be the OWL Lite formatted ontology (see the S-RAMP specification for more details). The response is an Atom Entry containing meta-data about the ontology, most importantly the UUID of the ontology (which can be later used to update or delete it).

Example Request

```
POST /s-ramp/ontology HTTP/1.1

<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xml:base="http://www.example.org/sample-ontology-1.owl">

    <owl:Ontology rdf:ID="SampleOntology1">
        <rdfs:label>Sample Ontology 1</rdfs:label>
        <rdfs:comment>A sample ontology.</rdfs:comment>
    </owl:Ontology>

    <owl:Class rdf:ID="All">
        <rdfs:label>All</rdfs:label>
    </owl:Class>

    <owl:Class rdf:ID="King">
        <rdfs:subClassOf rdf:resource="http://www.example.org/sample-
ontology-1.owl#All" />
        <rdfs:label>King</rdfs:label>
        <rdfs:comment>Feudal ruler.</rdfs:comment>
    </owl:Class>
    <owl:Class rdf:ID="Imperator">
        <rdfs:subClassOf rdf:resource="http://www.example.org/sample-
ontology-1.owl#All" />
        <rdfs:label>Imperator</rdfs:label>
        <rdfs:comment>Roman ruler.</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:ID="Baron">
        <rdfs:subClassOf rdf:resource="http://www.example.org/sample-
ontology-1.owl#King" />
        <rdfs:label>Baron</rdfs:label>
    </owl:Class>
    <owl:Class rdf:ID="Rex">
        <rdfs:subClassOf rdf:resource="http://www.example.org/sample-
ontology-1.owl#Imperator" />
        <rdfs:label>Imperator</rdfs:label>
    </owl:Class>

    <owl:Class rdf:ID="Knight">
        <rdfs:subClassOf rdf:resource="http://www.example.org/sample-
ontology-1.owl#Baron" />
        <rdfs:label>Knight</rdfs:label>
    </owl:Class>
```

```
    <owl:Class rdf:ID="Dux">
        <rdfs:subClassOf rdf:resource="http://www.example.org/sample-
ontology-1.owl#Rex" />
        <rdfs:label>Dux</rdfs:label>
    </owl:Class>

</rdf:RDF>
```

Example Response

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<atom:entry xmlns="http://www.w3.org/2000/01/rdf-schema#"
 xmlns:atom="http://www.w3.org/2005/Atom"
  xmlns:ns2="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:ns3="http://
www.w3.org/2002/07/owl#">
  <atom:title>Sample Ontology 1</atom:title>
  <atom:id>e8fe74f3-c9c3-4678-ba76-d71158141ddd</atom:id>
  <atom:author />
  <atom:summary>A sample ontology.</atom:summary>
  <ns2:RDF xml:base="http://www.example.org/sample-ontology-1.owl">
    <ns3:Ontology ns2:ID="SampleOntology1">
      <label>Sample Ontology 1</label>
      <comment>A sample ontology.</comment>
    </ns3:Ontology>
    <ns3:Class ns2:ID="All">
      <label>All</label>
    </ns3:Class>
    <ns3:Class ns2:ID="King">
      <subClassOf ns2:resource="http://www.example.org/sample-
ontology-1.owl#All" />
      <label>King</label>
      <comment>Feudal ruler.</comment>
    </ns3:Class>
    <ns3:Class ns2:ID="Imperator">
      <subClassOf ns2:resource="http://www.example.org/sample-
ontology-1.owl#All" />
      <label>Imperator</label>
      <comment>Roman ruler.</comment>
    </ns3:Class>
    <ns3:Class ns2:ID="Baron">
      <subClassOf ns2:resource="http://www.example.org/sample-
ontology-1.owl#King" />
      <label>Baron</label>
    </ns3:Class>
    <ns3:Class ns2:ID="Knight">
      <subClassOf ns2:resource="http://www.example.org/sample-
ontology-1.owl#Baron" />
```

```
      <label>Knight</label>
    </ns3:Class>
    <ns3:Class ns2:ID="Rex">
      <subClassOf ns2:resource="http://www.example.org/sample-
ontology-1.owl#Imperator" />
      <label>Imperator</label>
    </ns3:Class>
    <ns3:Class ns2:ID="Dux">
      <subClassOf ns2:resource="http://www.example.org/sample-
ontology-1.owl#Rex" />
      <label>Dux</label>
    </ns3:Class>
  </ns2:RDF>
</atom:entry>
```

8.14. API: List Ontologies

```
  /s-ramp/ontology
```

Retrieves, as an Atom feed, all ontologies currently known to the repository.

HTTP Method	Request	Response
GET	N/A	Atom Feed

This endpoint is used to retrieve all ontologies known to the repository as an Atom Feed of Entries, with one Entry for each ontology. Full information about the ontology can subsequently be retrieved by calling the Get Ontology endpoint.

Example Request

```
GET /s-ramp/ontology HTTP/1.1
```

Example Response

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<atom:feed xmlns:atom="http://www.w3.org/2005/Atom">
  <atom:title>S-RAMP ontology feed</atom:title>
  <atom:updated>2013-07-23T10:58:40.356-04:00</atom:updated>
  <atom:entry>
    <atom:title>Sample Ontology 1</atom:title>
    <atom:updated>2013-07-23T10:56:50.410-04:00</atom:updated>
    <atom:id>e8fe74f3-c9c3-4678-ba76-d71158141ddd</atom:id>
    <atom:published>2013-07-23T10:56:50.410-04:00</atom:published>
    <atom:author>
```

```
      <atom:name>eric</atom:name>
    </atom:author>
    <atom:source xml:base="http://www.example.org/sample-ontology-1.owl">
      <atom:id>SampleOntology1</atom:id>
    </atom:source>
    <atom:summary>A sample ontology.</atom:summary>
  </atom:entry>
  <atom:entry>
    <atom:title>Animal Kingdom</atom:title>
    <atom:updated>2013-07-23T10:58:37.737-04:00</atom:updated>
    <atom:id>fd0e5210-2567-409f-8df0-f851e1ce630d</atom:id>
    <atom:published>2013-07-23T10:58:37.737-04:00</atom:published>
    <atom:author>
      <atom:name>eric</atom:name>
    </atom:author>
    <atom:source xml:base="http://www.example.org/sample-ontology-2.owl">
      <atom:id>AnimalKingdom</atom:id>
    </atom:source>
    <atom:summary>Animal Kingdom Ontology</atom:summary>
  </atom:entry>
</atom:feed>
```

8.15. API: Update Ontology

```
  /s-ramp/ontology/{uuid}
```

Updates an existing ontology by its UUID.

HTTP Method	Request	Response
PUT	application/rdf+xml	N/A

This endpoint is used to update a single ontology in the repository. The request body must be a new version of the ontology in OWL Lite RDF format. Note that this might fail if the ontology changes in an incompatible way (e.g. a class is removed that is currently in use).

Example Request

```
PUT /s-ramp/ontology/fd0e5210-2567-409f-8df0-f851e1ce630d HTTP/1.1

<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xml:base="http://www.example.org/sample-ontology-1.owl">
```

```
    <owl:Ontology rdf:ID="SampleOntology1">
        <rdfs:label>Sample Ontology 1</rdfs:label>
        <rdfs:comment>A sample ontology.</rdfs:comment>
    </owl:Ontology>

    <owl:Class rdf:ID="All">
        <rdfs:label>All</rdfs:label>
    </owl:Class>

    <owl:Class rdf:ID="King">
        <rdfs:subClassOf rdf:resource="http://www.example.org/sample-
ontology-1.owl#All" />
        <rdfs:label>King</rdfs:label>
        <rdfs:comment>Feudal ruler.</rdfs:comment>
    </owl:Class>
    <owl:Class rdf:ID="Imperator">
        <rdfs:subClassOf rdf:resource="http://www.example.org/sample-
ontology-1.owl#All" />
        <rdfs:label>Imperator</rdfs:label>
        <rdfs:comment>Roman ruler.</rdfs:comment>
    </owl:Class>

    <owl:Class rdf:ID="Baron">
        <rdfs:subClassOf rdf:resource="http://www.example.org/sample-
ontology-1.owl#King" />
        <rdfs:label>Baron</rdfs:label>
    </owl:Class>
    <owl:Class rdf:ID="Rex">
        <rdfs:subClassOf rdf:resource="http://www.example.org/sample-
ontology-1.owl#Imperator" />
        <rdfs:label>Imperator</rdfs:label>
    </owl:Class>

    <owl:Class rdf:ID="Knight">
        <rdfs:subClassOf rdf:resource="http://www.example.org/sample-
ontology-1.owl#Baron" />
        <rdfs:label>Knight</rdfs:label>
    </owl:Class>
    <owl:Class rdf:ID="Dux">
        <rdfs:subClassOf rdf:resource="http://www.example.org/sample-
ontology-1.owl#Rex" />
        <rdfs:label>Dux</rdfs:label>
    </owl:Class>

</rdf:RDF>
```

Example Response

```
HTTP/1.1 200 OK
```

8.16. API: Get Ontology

```
/s-ramp/ontology/{uuid}
```

Returns the OWL representation of an ontology (wrapped in an Atom Entry).

HTTP Method	Request	Response
GET	N/A	Atom Entry

This endpoint is used to get the full ontology (by its UUID) in OWL Lite (RDF) format, wrapped in an Atom Entry. The response body is an Atom Entry with a single extension element that is the ontology RDF. This will fail if no ontology exists with the given UUID.

Example Request

```
GET /s-ramp/ontology/fd0e5210-2567-409f-8df0-f851e1ce630d HTTP/1.1
```

Example Response

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<atom:entry xmlns="http://www.w3.org/2000/01/rdf-schema#"
 xmlns:atom="http://www.w3.org/2005/Atom"
  xmlns:ns2="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:ns3="http://
www.w3.org/2002/07/owl#">
  <atom:title>Animal Kingdom</atom:title>
  <atom:updated>2013-07-23T10:58:37.737-04:00</atom:updated>
  <atom:id>fd0e5210-2567-409f-8df0-f851e1ce630d</atom:id>
  <atom:published>2013-07-23T10:58:37.737-04:00</atom:published>
  <atom:author>
    <atom:name>eric</atom:name>
  </atom:author>
  <atom:summary>Animal Kingdom Ontology</atom:summary>
  <ns2:RDF xml:base="http://www.example.org/sample-ontology-2.owl">
    <ns3:Ontology ns2:ID="AnimalKingdom">
      <label>Animal Kingdom</label>
      <comment>Animal Kingdom Ontology</comment>
    </ns3:Ontology>
    <ns3:Class ns2:ID="Animal">
      <label>Animal</label>
      <comment>All animals.</comment>
    </ns3:Class>
    <ns3:Class ns2:ID="UnicellularAnimal">
      <subClassOf ns2:resource="http://www.example.org/sample-
ontology-2.owl#Animal" />
      <label>Unicellular Animal</label>
```

```
      <comment>Single-celled animal.</comment>
    </ns3:Class>
    <ns3:Class ns2:ID="MulticellularAnimal">
      <subClassOf ns2:resource="http://www.example.org/sample-
ontology-2.owl#Animal" />
      <label>Multicellular Animal</label>
      <comment>Multi-celled animal.</comment>
    </ns3:Class>
    <ns3:Class ns2:ID="Protozoa">
      <subClassOf ns2:resource="http://www.example.org/sample-
ontology-2.owl#UnicellularAnimal" />
      <label>Protozoa</label>
    </ns3:Class>
    <ns3:Class ns2:ID="Metazoa">
      <subClassOf ns2:resource="http://www.example.org/sample-
ontology-2.owl#MulticellularAnimal" />
      <label>Metazoa</label>
    </ns3:Class>
    <ns3:Class ns2:ID="Invertebrate">
      <subClassOf ns2:resource="http://www.example.org/sample-
ontology-2.owl#Metazoa" />
      <label>Invertebrate</label>
    </ns3:Class>
    <ns3:Class ns2:ID="Vertebrate">
      <subClassOf ns2:resource="http://www.example.org/sample-
ontology-2.owl#Metazoa" />
      <label>Vertebrate</label>
    </ns3:Class>
  </ns2:RDF>
</atom:entry>
```

8.17. API: Delete Ontology

```
  /s-ramp/ontology/{uuid}
```

Deletes an ontology from the repository.

HTTP Method	Request	Response
DELETE	N/A	N/A

This endpoint is used to delete a single ontology from the repository. This might fail if the ontology is currently in-use (at least one artifact is classified by at least one class defined by the ontology).

Example Request

```
DELETE /s-ramp/ontology/fd0e5210-2567-409f-8df0-f851e1ce630d HTTP/1.1
```

Example Response

```
HTTP/1.1 200 OK
```

8.18. API: Get Artifact Audit History

```
  /s-ramp/audit/artifact/{artifactUuid}
```

Retrieves an Atom feed containing all of the audit entries for a single artifact.

HTTP Method	Request	Response
GET	N/A	Atom Feed

This endpoint is used to get a feed of the audit history of a single artifact. The request URL can include standard paging parameters. The response is an Atom Feed where each Entry in the feed represents a single audit event in the history of the artifact. A followup call must be made to the Get Artifact Audit Entry endpoint in order to retrieve full detail information about the audit event. This call might fail if no artifact exits with the given UUID.

Example Request

```
GET /s-ramp/audit/artifact/b086c558-58d6-4837-bb38-6c3da760ae80 HTTP/1.1
```

Example Response

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<atom:feed xmlns:atom="http://www.w3.org/2005/Atom" xmlns:s-ramp="http://
docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
  s-ramp:itemsPerPage="100" s-ramp:provider="JBoss Overlord" s-
ramp:startIndex="0" s-ramp:totalResults="2">
  <atom:title>S-RAMP Audit Feed</atom:title>
  <atom:subtitle>All Audit Entries for Artifact</atom:subtitle>
  <atom:updated>2013-07-23T11:14:07.189-04:00</atom:updated>
  <atom:id>bff03dd5-e55c-4528-b1aa-ee1eb471b899</atom:id>
  <atom:entry>
    <atom:title>artifact:update</atom:title>
    <atom:updated>2013-07-23T11:14:03.225-04:00</atom:updated>
    <atom:id>2947f90e-0f5a-4099-b3dc-29124c96c7d0</atom:id>
    <atom:published>2013-07-23T11:14:03.225-04:00</atom:published>
    <atom:author>
      <atom:name>eric</atom:name>
    </atom:author>
    <atom:summary />
  </atom:entry>
  <atom:entry>
    <atom:title>artifact:add</atom:title>
```

```
    <atom:updated>2013-07-23T11:13:28.513-04:00</atom:updated>
    <atom:id>e41404b3-9ec6-43f5-a6d8-aa6089bc6704</atom:id>
    <atom:published>2013-07-23T11:13:28.513-04:00</atom:published>
    <atom:author>
      <atom:name>eric</atom:name>
    </atom:author>
    <atom:summary />
  </atom:entry>
</atom:feed>
```

8.19. API: Get User Audit History

```
  /s-ramp/audit/user/{username}
```

Retrieves an Atom feed containing all of the audit entries for a specific user.

HTTP Method	Request	Response
GET	N/A	Atom Feed

This endpoint is used to get a feed of the audit history for a single user. The request URL can include standard paging parameters. The response is an Atom Feed where each Entry in the feed represents a single audit event in the history of the artifact. A followup call must be made to the Get Artifact Audit Entry endpoint in order to retrieve full detail information about the audit event. This call might fail if no user exists with the given username.

Example Request

```
GET /s-ramp/audit/user/eric HTTP/1.1
```

Example Response

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<atom:feed xmlns:atom="http://www.w3.org/2005/Atom" xmlns:s-ramp="http://
docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0"
  s-ramp:itemsPerPage="100" s-ramp:provider="JBoss Overlord" s-
ramp:startIndex="0" s-ramp:totalResults="2">
  <atom:title>S-RAMP Audit Feed</atom:title>
  <atom:subtitle>All Audit Entries for Artifact</atom:subtitle>
  <atom:updated>2013-07-23T11:16:00.545-04:00</atom:updated>
  <atom:id>d49057a2-2f84-48aa-9c79-078b1e86680a</atom:id>
  <atom:entry>
    <atom:title>artifact:update</atom:title>
    <atom:updated>2013-07-23T11:14:03.225-04:00</atom:updated>
    <atom:id>2947f90e-0f5a-4099-b3dc-29124c96c7d0</atom:id>
```

```
    <atom:published>2013-07-23T11:14:03.225-04:00</atom:published>
    <atom:author>
      <atom:name>eric</atom:name>
    </atom:author>
    <atom:summary />
  </atom:entry>
  <atom:entry>
    <atom:title>artifact:add</atom:title>
    <atom:updated>2013-07-23T11:13:28.513-04:00</atom:updated>
    <atom:id>e41404b3-9ec6-43f5-a6d8-aa6089bc6704</atom:id>
    <atom:published>2013-07-23T11:13:28.513-04:00</atom:published>
    <atom:author>
      <atom:name>eric</atom:name>
    </atom:author>
    <atom:summary />
  </atom:entry>
</atom:feed>
```

8.20. API: Add Artifact Audit Entry

```
  /s-ramp/audit/artifact/{artifactUuid}
```

Adds a user-defined (custom) audit entry to an artifact.

HTTP Method	Request	Response
POST	application/auditEntry+xml	Atom Entry

This endpoint is used to add a custom audit entry to a particular artifact. The request must be a POST of an XML document conforming to the audit schema type *auditEntry*. This call may fail if no artifact exists with the given UUID.

Example Request

```
POST /s-ramp/audit/artifact/b086c558-58d6-4837-bb38-6c3da760ae80 HTTP/1.1
```

Example Response

```
HTTP/1.1 200 OK

<auditEntry type="custom:foo" uuid="" when="" who="">
  <auditItem type="custom:item-type-1">
    <property name="my-property-1" value="some-value" />
    <property name="my-property-2" value="other-value" />
  </auditItem>
  <auditItem type="custom:item-type-2" />
</auditEntry>
```

8.21. API: Get Artifact Audit Entry

```
/s-ramp/audit/artifact/{artifactUuid}/{auditEntryUuid}
```

Retrieves full detailed information about a single audit entry.

HTTP Method	Request	Response
GET	N/A	Atom Entry

This endpoint is used to get the full details for a single audit event for a particular artifact. The particulars of the detailed information are specific to the type of audit entry, so *artifact create* detail information might be different from *artifact update* detail information. In addition, there is the possibility that the detail information might be from a custom audit entry added by an end user. This call might fail if the artifact does not exist or the audit entry does not exist.

Example Request

```
GET /s-ramp/audit/artifact/b086c558-58d6-4837-
bb38-6c3da760ae80/2947f90e-0f5a-4099-b3dc-29124c96c7d0 HTTP/1.1
```

Example Response

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<atom:entry xmlns="http://downloads.jboss.org/overlord/sramp/2013/
auditing.xsd" xmlns:atom="http://www.w3.org/2005/Atom">
  <atom:title>artifact:update</atom:title>
  <atom:updated>2013-07-23T11:14:03.225-04:00</atom:updated>
  <atom:id>2947f90e-0f5a-4099-b3dc-29124c96c7d0</atom:id>
  <atom:published>2013-07-23T11:14:03.225-04:00</atom:published>
  <atom:author>
    <atom:name>eric</atom:name>
  </atom:author>
  <atom:summary />
  <auditEntry type="artifact:update" uuid="2947f90e-0f5a-4099-
b3dc-29124c96c7d0" when="2013-07-23T11:14:03.225-04:00"
    who="eric">
    <auditItem type="property:added">
      <property name="sramp-properties:foo" value="bar" />
      <property name="sramp-properties:hello" value="world" />
    </auditItem>
    <auditItem type="property:changed" />
    <auditItem type="property:removed" />
  </auditEntry>
</atom:entry>
```

Chapter 9. The S-RAMP Browser (UI)

9.1. Overview

The Overlord S-RAMP project comes with a user interface that allows end users (or more likely business admins) to browse all of the artifacts in the S-RAMP repository. This UI is capable of viewing and manipulating all S-RAMP artifacts in a very generic way, supporting all aspects of the S-RAMP specification (properties, classifiers, relationships, etc).

The browser is a web based application built using GWT and Errai, and is compatible with all modern web browsers. Additionally, it is capable of scaling the interface down to a size that is useful on a smart phone.

9.2. Configuration

The UI can be configured via an external properties file named **sramp-ui.properties** located in the application server's configuration directory. This configuration file can contain UI specific configuration such as:

```
# The location of the S-RAMP server's Atom API
s-ramp-ui.atom-api.endpoint
# Whether or not to validate the S-RAMP server endpoint when connecting to
 it
s-ramp-ui.atom-api.validating
# The authentication provider to use when connecting
s-ramp-ui.atom-api.authentication.provider
# BASIC auth username/password
s-ramp-ui.atom-api.authentication.basic.user
s-ramp-ui.atom-api.authentication.basic.password
```

Alternatively, a configuration file location can be provided by setting a Java system property (e.g. JAVA_OPTS) with the following name:

sramp-ui.config.file.name

9.3. Configuration (EAP)

When running in JBoss EAP this same configuration information is instead stored in the **JBOSS/standalone/configuration/standalone.xml** file under the **urn:jboss:domain:overlord-configuration:1.0** subsystem. For example:

```
    <subsystem xmlns="urn:jboss:domain:overlord-configuration:1.0">
      <configurations>
```

```
        <configuration name="sramp-ui">
          <properties>
            <property name="s-ramp-ui.atom-api.endpoint"
 value="${overlord.baseUrl}/s-ramp-server" />
            <property name="s-ramp-ui.atom-api.authentication.provider"
 value="org.overlord.sramp.ui.server.api.SAMLBearerTokenAuthenticationProvider" /
>
            <property name="s-ramp-ui.atom-api.authentication.saml.issuer"
 value="/s-ramp-ui" />
            <property name="s-ramp-ui.atom-api.authentication.saml.service"
 value="/s-ramp-server" />
            <property name="s-ramp-ui.atom-api.authentication.saml.sign-
assertions" value="true" />
            <property name="s-ramp-ui.atom-api.authentication.saml.keystore"
 value="${overlord.auth.saml-keystore}" />
            <property name="s-ramp-ui.atom-api.authentication.saml.keystore-
password" value="${overlord.auth.saml-keystore-password}" />
            <property name="s-ramp-ui.atom-api.authentication.saml.key-
alias" value="${overlord.auth.saml-key-alias}" />
            <property name="s-ramp-ui.atom-api.authentication.saml.key-
password" value="${overlord.auth.saml-key-alias-password}" />
          </properties>
        </configuration>
      </configurations>
    </subsystem>
```

9.3.1. Security (Authentication)

The S-RAMP Browser is protected using web application security mechanisms configured in the web.xml.

By default, the UI uses SAML based single-sign-on (SSO) as the actual authentication mechanism. The SSO is provided via an Overlord SAML IDP web appliation (which is shared across all Overlord UI projects).

The Overlord SAML IDP is a simple web application that leverages the web app container's FORM authentication support and then implements the SAML SSO Web Browser protocol. This allows other web applications (called Service Providers) to leverage the IDP as a source of authentication.

Overlord SSO is enabled in the S-RAMP Browser UI web application by configuring a SAML SP servlet filter to protect the UI's html host page. This filter implements the Service Provider side of the SAML SSO protocol and is configured in the web.xml like so:

```
  <filter>
    <filter-name>AuthenticationFilter</filter-name>
    <filter-class>org.picketlink.identity.federation.web.filters.SPFilter</
filter-class>
    <init-param>
      <param-name>ROLES</param-name>
```

```
      <param-value>overlorduser</param-value>
    </init-param>
  </filter>
  <filter-mapping>
    <filter-name>AuthenticationFilter</filter-name>
    <url-pattern>/</url-pattern>
  </filter-mapping>
  <filter-mapping>
    <filter-name>AuthenticationFilter</filter-name>
    <url-pattern>/index.html</url-pattern>
  </filter-mapping>
```

This protects the UI's host page (index.html) and requires that any user logging in must have the *overlorduser* role.

This SPFilter class works together with the IDP web application to provide web based single-sign-on. This functionality is provided by the PicketLink project, which also requires one final bit of configuration via a *picketlink.xml* file also included in the WEB-INF folder of the Service Provider web application:

```
<PicketLink xmlns="urn:picketlink:identity-federation:config:2.1">
  <PicketLinkSP xmlns="urn:picketlink:identity-federation:config:2.1"
 ServerEnvironment="tomcat"
    BindingType="REDIRECT" RelayState="someURL">
    <IdentityURL>${overlord-idp.url::/overlord-idp/}</IdentityURL>
    <ServiceURL>${dtgov-ui.url::/dtgov-ui/}</ServiceURL>
  </PicketLinkSP>
  <Handlers xmlns="urn:picketlink:identity-federation:handler:config:2.1">
    <Handler
 class="org.picketlink.identity.federation.web.handlers.saml2.SAML2LogOutHandler" /
>
    <Handler
 class="org.picketlink.identity.federation.web.handlers.saml2.SAML2AuthenticationHandler" /
>
    <Handler class="org.overlord.commons.auth.handlers.RoleCachingHandler" /
>
  </Handlers>
</PicketLink>
```

All of the above configuration enables SAML based SSO for the Browser web app. It's worth pointing out that the IDP web app is configured to use the application container's local mechanism for authentication and authorization. So you should look up the details of this for whatever platform you are running.

When running in JBoss EAP, the Overlord IDP is configured to use the following security domain (configured in standalone.xml):

```
<security-domain name="overlord-idp" cache-type="default">
```

```
  <authentication>
    <login-module code="RealmDirect" flag="required">
      <module-option name="password-stacking" value="useFirstPass"/>
    </login-module>
  </authentication>
</security-domain>
```

This security domain passes through authentication to the JBoss Application security realm. By default, the Application Realm is configured like this:

```
<security-realm name="ApplicationRealm">
    <authentication>
        <local default-user="$local" allowed-users="*" skip-group-
loading="true"/>
        <properties path="application-users.properties" relative-
to="jboss.server.config.dir"/>
    </authentication>
    <authorization>
        <properties path="application-roles.properties" relative-
to="jboss.server.config.dir"/>
    </authorization>
</security-realm>
```

This uses simple property files to configure the users and groups. It is recommended for most production systems to replace this with some other approach, such as an LDAP configuration.

9.3.2. Security (Authorization)

Currently the S-RAMP Browser UI does not support any sort of fine grained authorization. As a result, the user must simply have the following role in order to log in and use the UI:

```
overlorduser
```

Chapter 10. Overlord S-RAMP Command Line

Using the S-RAMP cmdline tool `s-ramp.sh`

In the `bin` directory of the distribution you can find the `s-ramp.sh`. Run this command to fire up the shell

```
./s-ramp.sh
************************************************************************
          _____        ______  ___ ___  ________
         /  ___|      | ___ \/ _ \|  \/  | ___ \
         \ `--. ______| |_/ / /_\ \ .  . | |_/ /
          `--. \______|    /|  _  | |\/| |  __/
         /\__/ /      | |\ \| | | | |  | | |
         \____/       \_| \_\_| |_|_|  |_|_|

  JBoss S-RAMP Kurt Stam and Eric Wittmann, Licensed under the
  Apache License, V2.0, Copyright 2012
************************************************************************
s-ramp>
```

The shell supports auto-completion and keeps a command history for duration of the session.

10.1. Connecting to S-RAMP server

To connect the shell to the server type `connect` and hit the tab key. It should auto-complete to say `s-ramp:connect http://localhost:8080/s-ramp-server` and when hitting the return key you will be prompted for user credentials. If everything is successful, the cursor should go from red to green. Of course you will need to update the server and port information if your S-RAMP repository runs elsewhere.

10.2. Browsing the S-RAMP repository

To browse the artifacts in the repository run the following query:

```
s-ramp> s-ramp:query /s-ramp
Querying the S-RAMP repository:
    /s-ramp
Atom Feed (9 entries)
  Idx                    Type Name
  ---                    ---- ----
    1           ImageDocument user-properties.png
    2                Document overlord.demo.CheckDeployment-taskform.flt
```

```
    3         BrmsPkgDocument SRAMPPackage.pkg
    4           ImageDocument overlord.demo.SimpleReleaseProcess-image.png
    5           ImageDocument run-build-install.png
    6                Document overlord.demo.SimpleReleaseProcess-
taskform.flt
    7           ImageDocument audio-input-microphone-3.png
    8            BpmnDocument overlord.demo.SimpleReleaseProcess.bpmn
    9            TextDocument HttpClientWorkDefinitions.wid
```

To obtain the metaData of `overlord.demo.SimpleReleaseProcess.bpmn`, which is number 8 in the list, issue

```
s-ramp> s-ramp:getMetaData feed:8
Meta Data for: 31b3acbc-cda8-4856-9e34-d3e645283035
--------------
  -- Core S-RAMP Info --
  Type: BpmnDocument
  Model: ext
  UUID: 31b3acbc-cda8-4856-9e34-d3e645283035
  Name: overlord.demo.SimpleReleaseProcess.bpmn
  Derived: false
  Created By: <anonymous>
  Created On: 2013-03-08T14:00:37.036-05:00
  Modified By: <anonymous>
  Modified On: 2013-03-18T14:58:46.328-04:00
s-ramp>
```

10.3. Updating artifact MetaData

10.3.1. Properties

To update a property on the artifact use `s-ramp:property set` and hit the `tab` key

```
s-ramp> s-ramp:property set
description       name           version
```

this shows a list of properties that can be updated. To add a description to this artifact use

```
s-ramp> s-ramp:property set description "BPMN2 artifact representing the
 SimpleReleaseProcess"
Successfully set property description.
s-ramp> s-ramp:updateMetaData
Successfully updated artifact overlord.demo.SimpleReleaseProcess.bpmn.
```

To verify issue

```
s-ramp> s-ramp:getMetaData feed:8
```

```
Meta Data for: 31b3acbc-cda8-4856-9e34-d3e645283035
--------------
  -- Core S-RAMP Info --
  Type: BpmnDocument
  Model: ext
  UUID: 31b3acbc-cda8-4856-9e34-d3e645283035
  Name: overlord.demo.SimpleReleaseProcess.bpmn
  Derived: false
  Created By: <anonymous>
  Created On: 2013-03-08T14:00:37.036-05:00
  Modified By: <anonymous>
  Modified On: 2013-03-18T16:09:56.879-04:00
  -- Description --
BPMN2 artifact representing the SimpleReleaseProcess
```

and you can see the added description at the bottom of the printout.

10.3.2. Custom Properties

To add a `custom property` called `kurt` with value `stam` you can run

```
s-ramp> s-ramp:property set kurt stam
Successfully set property kurt.
s-ramp> s-ramp:updateMetaData
Successfully updated artifact overlord.demo.SimpleReleaseProcess.bpmn.
```

and to verify that the `custom property` was added issue

```
s-ramp> s-ramp:getMetaData feed:8
Meta Data for: 31b3acbc-cda8-4856-9e34-d3e645283035
--------------
  -- Core S-RAMP Info --
  Type: BpmnDocument
  Model: ext
  UUID: 31b3acbc-cda8-4856-9e34-d3e645283035
  Name: overlord.demo.SimpleReleaseProcess.bpmn
  Derived: false
  Created By: <anonymous>
  Created On: 2013-03-08T14:00:37.036-05:00
  Modified By: <anonymous>
  Modified On: 2013-03-18T16:21:16.119-04:00
  -- Description --
BPMN2 artifact representing the SimpleReleaseProcess
  -- Custom Properties --
  kurt: stam
s-ramp>
```

When hitting the `tab` key on `s-ramp:property set` results in

```
s-ramp> s-ramp:property set
description    kurt           name           version
```

which now had the added `custom property` `kurt`.

10.3.3. Classifications

To add a `classification` of `deployment-status` to your artifact use

```
s-ramp> s-ramp:classification add "http://www.jboss.org/overlord/deployment-
status.owl#Dev"
Successfully added classification 'http://www.jboss.org/overlord/deployment-
status.owl#Dev'.
s-ramp> s-ramp:updateMetaData
Successfully updated artifact overlord.demo.SimpleReleaseProcess.bpmn.
```

and to verify that it was added

```
s-ramp> s-ramp:getMetaData feed:8
Meta Data for: 31b3acbc-cda8-4856-9e34-d3e645283035
--------------
  -- Core S-RAMP Info --
  Type: BpmnDocument
  Model: ext
  UUID: 31b3acbc-cda8-4856-9e34-d3e645283035
  Name: overlord.demo.SimpleReleaseProcess.bpmn
  Derived: false
  Created By: <anonymous>
  Created On: 2013-03-08T14:00:37.036-05:00
  Modified By: <anonymous>
  Modified On: 2013-03-18T16:30:42.641-04:00
  -- Description --
BPMN2 artifact representing the SimpleReleaseProcess
  -- Classifications --
  Classified By: http://www.jboss.org/overlord/deployment-status.owl#Dev
  -- Custom Properties --
  kurt: stam
s-ramp>
```

10.4. Querying the S-RAMP Repository using XPath2 Syntax

S-RAMP supports an XPath2 Syntax for querying. For example to obtain all WSDL models in the repository use

```
s-ramp> s-ramp:query /s-ramp/wsdl/WsdlDocument
```

```
Querying the S-RAMP repository:
    /s-ramp/wsdl/WsdlDocument
Atom Feed (1 entries)
  Idx                   Type Name
  ---                   ---- ----
    1           WsdlDocument OrderService.wsdl
s-ramp>
```

When this WSDL file was uploaded `derived` information was extracted from it and stored a WSDL model. TO see the various data structures it derived simply hit the tab on `s-ramp:query /s-ramp/wsdl`

```
s-ramp> s-ramp:query /s-ramp/wsdl/
Binding                 BindingOperation         BindingOperationFault
 BindingOperationInput   BindingOperationOutput
Fault                   Message                  Operation
 OperationInput          OperationOutput
Part                    Port                     PortType
 WsdlDocument            WsdlExtension
WsdlService
s-ramp>
```

Note that derived data is `read only`, and cannot be updated by the user.

To obtain all Operations in this WSDL use

```
s-ramp:query /s-ramp/wsdl/Operation
Querying the S-RAMP repository:
    /s-ramp/wsdl/Operation
Atom Feed (1 entries)
  Idx                   Type Name
  ---                   ---- ----
    1              Operation submitOrder
s-ramp>
```

You can narrow this query down even more by adding that the name needs to start with `submit`

```
s-ramp:query "/s-ramp/wsdl/Operation[xp2:matches(@name, 'submit.*')]"
Querying the S-RAMP repository:
    /s-ramp/wsdl/Operation[xp2:matches(@name, 'submit.*')]
Atom Feed (1 entries)
  Idx                   Type Name
  ---                   ---- ----
    1              Operation submitOrder
s-ramp>
```

don't forget to use the surrounding quotes, and a `.` after `submit` as required by XPath2.

To obtain all the artifacts that were derived from an artifact you can use

```
/s-ramp[relatedDocument[@uuid = '<uuid>'
```

In this case we use the uuid of a wsdl and get all the artifacts derived from the wsdl

```
s-ramp:query "/s-ramp[relatedDocument[@uuid = '15a94308-a088-4a03-ad83-
e60239af74e4']]"
Querying the S-RAMP repository:
 /s-ramp[relatedDocument[@uuid = '15a94308-a088-4a03-ad83-e60239af74e4']]
Atom Feed (16 entries)
  Idx                   Type Name
  ---                   ---- ----
    1          OperationInput submitOrder
    2             WsdlService OrderService
    3             SoapAddress soap:address
    4   BindingOperationInput wsdl:input
    5             SoapBinding soap:binding
    6                    Part parameters
    7                 Binding OrderServiceBinding
    8  BindingOperationOutput wsdl:output
    9                 Message submitOrderResponse
   10         OperationOutput submitOrderResponse
   11        BindingOperation submitOrder
   12                 Message submitOrder
   13               Operation submitOrder
   14                    Port OrderServicePort
   15                    Part parameters
   16                PortType OrderService
```

To get a list of all artifacts that were extracted from another archive use

```
s-ramp:query "/s-ramp[expandedFromDocument[@uuid = '<uuid>']]"
```

let's say we uploaded a jar file containing switchyard artifacts, with uddi *67c6f2d3-0f10-4f0d-ada6-d85f92f02a33*:

```
s-ramp:query "/s-ramp[expandedFromDocument[@uuid = '67c6f2d3-0f10-4f0d-ada6-
d85f92f02a33']]"
Querying the S-RAMP repository:
 /s-ramp[expandedFromDocument[@uuid = '67c6f2d3-0f10-4f0d-ada6-
d85f92f02a33']]
Atom Feed (3 entries)
  Idx                   Type Name
  ---                   ---- ----
    1             XmlDocument switchyard.xml
    2             XmlDocument beans.xml
```

```
3             XmlDocument faces-config.xml
```

For more information about querying the repository see the *S-RAMP Query Language* section of this guide.

10.5. Extending the S-RAMP CLI

The S-RAMP CLI has a number of built-in commands that are ready to be used. However, it is also possible to extend the CLI with new custom commands. This section describes how to do it.

New CLI commands are contributed by creating a class that implements the *ShellCommandProvider* interface. The provider will indicate a namespace for its commands along with a Map of commands (command name → command). The provider and command implementations should be packaged up into a JAR along with a file named:

```
META-INF/services/org.overlord.sramp.shell.api.ShellCommandProvider
```

The JAR must be made available to the S-RAMP CLI, either by putting it on the classpath, or else by putting it in the following directory:

```
~/.s-ramp/commands
```

For a working example of a custom S-RAMP CLI command, there is a demo in the S-RAMP distribution called **s-ramp-demos-shell-command**.

10.6. Running Commands in Batch

An interesting thing you can do with the S-RAMP CLI is to use it as a batch processor. To do this, simply create a text file with all of the commands you wish to run in a batch (one per line) and then ask the S-RAMP CLI to execute the batch. For example, a batch of commands may look like this:

```
# Connect to S-RAMP
connect http://localhost:8080/s-ramp-server admin admin123!

# Upload an ontology
ontology:upload /path/to/data/my-ontology.owl

# Add some artifact content
upload /path/to/artifact-content.ext
property set property-foo Bar
updateMetaData
```

To execute the batch, simply do:

```
sramp.sh -f /path/to/cli-commands.txt
```

10.7. Batch File Property Interpolation

Note that it is possible to use Ant style property replacements within your S-RAMP CLI batch file. The CLI will look for property values as System Properties, or by passing in the path to a Java Properties file to the CLI via a "-propertiesFile" option.

We support simply property replacement as well as property replacement with defaults. For example:

```
# Connect to S-RAMP
connect ${sramp.endpoint:http://localhost:8080/s-ramp-server}
 ${sramp.username:admin} ${sramp.password:admin123!}
upload ${resource.path}
```

The above batch file allows whoever is using it (via the S-RAMP CLI) to set the following properties either via System Properties or via a passed-in properties file:

- resource.path - (required)
- sramp.endpoint - (optional, defaults to *http://localhost:8080/s-ramp-server*)
- sramp.username - (optional, defaults to admin)
- sramp.password - (optional, defaults to admin123!)

10.8. Log-to-File

Rather than creating batch files by hand, the S-RAMP CLI includes a "log-to-file" option. All commands executed during the CLI session will be logged to a file, directly usable as a batch file in the future.

```
sramp.sh -l /path/to/cli-commands.txt
```

Chapter 11. Overlord S-RAMP Maven Integration

11.1. Overview

A key feature of the Overlord S-RAMP project is its integration with Maven. Currently there are several mechanisms provided to integrate with Maven. The first mechanism is a custom S-RAMP Maven Wagon that adds support for the S-RAMP Atom based REST API protocol. The second mechanism is an HTTP servlet which acts as a facade in front of the S-RAMP repository. Finally, there is a "maven" namespace in the S-RAMP Shell (CLI) providing integration between the CLI and Maven.

The S-RAMP Maven Wagon can be used to upload deployable artifacts directly from Maven into a compliant S-RAMP repository. It allows a number of options including specifying the Artifact Type and creating an ArtifactGrouping (more on these later). Additionally, artifacts from the S-RAMP repository can be used as dependencies in a Maven project.

The S-RAMP Maven HTTP Facade currently allows only basic integration with Maven, but in a way that does not require the use of a custom Wagon in your pom.xml. The facade does not currently support the same set of optional features that the wagon implements. However, for relatively simple integrations it is a very easy solution to get working.

11.2. Enabling the S-RAMP Wagon

In order to use the S-RAMP Wagon in a maven project, it must first be enabled in the pom.xml build section. Here's how it's done:

```
<build>
  <extensions>
    <extension>
      <groupId>org.overlord.sramp</groupId>
      <artifactId>s-ramp-wagon</artifactId>
      <version>${s-ramp-wagon.version}</version>
    </extension>
  </extensions>
</build>
```

11.3. Deploying to S-RAMP

Once the wagon is enabled, then URLs with a schema of "sramp" can be used in the pom.xml's distributionManagement section. For example:

```
<distributionManagement>
```

```
  <repository>
    <id>local-sramp-repo</id>
    <name>S-RAMP Releases Repository</name>
    <url>sramp://localhost:8080/s-ramp-server/</url>
  </repository>
  <snapshotRepository>
    <id>local-sramp-repo-snapshots</id>
    <name>S-RAMP Snapshots Repository</name>
    <url>sramp://localhost:8080/s-ramp-server/</url>
  </snapshotRepository>
</distributionManagement>
```

With these settings, maven deployments will be sent directly to the S-RAMP repository using the S-RAMP API. Note that artifacts will be added to the S-RAMP repository with an artifact type based on the maven type of the project. This behavior can be overridden by adding a query parameter to the repository URL in the pom.xml. For example:

```
<distributionManagement>
  <repository>
    <id>local-sramp-repo</id>
    <name>S-RAMP Releases Repository</name>
    <url>sramp://localhost:8080/s-ramp-server/?
artifactType=SwitchYardApplication</url>
  </repository>
</distributionManagement>
```

The above example will cause the maven artifact to be uploaded with an S-RAMP artifact type of "SwitchYardApplication" whenever a maven deployment or release build is performed.

For example, the following maven command could be run to deploy the maven artifact directly into s-ramp:

```
mvn clean deploy
```

11.4. Adding S-RAMP Artifacts as Dependencies

Additionally (after enabling the wagon [see above]), artifacts from the S-RAMP repository can be used as dependencies in your maven project.

First, the S-RAMP repository must be configured in the maven project as a maven repository. This can be done with the following markup in the pom.xml.

```
<repositories>
  <repository>
    <id>local-sramp-repo</id>
    <name>Local S-RAMP Repository</name>
    <url>sramp://localhost:8080/s-ramp-server</url>
```

```
    <layout>default</layout>
  </repository>
</repositories>
```

Once the repository is configured, an S-RAMP artifact can be referenced as a dependency in two ways. First, if the artifact was added to S-RAMP using the maven integration to deploy it, then the artifact in S-RAMP will contain maven specific properties, allowing it to be referenced as a dependency using those maven specific properties. In this case, simply add the dependency as you normally would in a maven project. For example:

```
<dependency>
  <groupId>org.overlord.sramp.wiki</groupId>
  <artifactId>s-ramp-wiki-example</artifactId>
  <version>1.0</version>
</dependency>
```

However, even if an artifact was added to the S-RAMP repository in some other way (and therefore does not have any maven specific properties) it can be used as a dependency. In this case, you can reference the dependency by using its S-RAMP artifact model, type, and UUID. The model and type are used to make up a maven groupId, while the UUID becomes the maven artifactId. The version information is not used (but still required in the pom.xml). For example, if a JAR is added to the S-RAMP repository and you wish to use it as a dependency, your pom.xml might contain the following dependency.

```
<dependency>
  <groupId>ext.JavaArchive</groupId>
  <artifactId>8744-437487-4734525-382345-923424</artifactId>
  <version>1.0</version>
</dependency>
```

11.5. Leveraging the S-RAMP Maven HTTP Facade

A less feature-rich (currently) but easier to configure maven integration option is the S-RAMP Maven HTTP facade. This HTTP servlet can be accessed (by default) from the following URL:

```
  http://localhost:8080/s-ramp-server/maven/repository
```

This URL can be treated as the root of a standard Maven repository both for deploying artifacts to the S-RAMP repository and also for getting artifacts back out again as dependencies. You can use standard Maven configuration of your "repositories" (for GETs) and "distributionManagement" (for PUTs) within your pom.xml. There is no need to configure a wagon or any other maven extension.

An example configuration in your pom.xml for this mechanism might be:

```

  <repositories>
```

```
    <repository>
      <id>local-sramp-repo</id>
      <name>Local S-RAMP Repository</name>
      <url>http://localhost:8080/s-ramp-server/maven/repository</url>
      <layout>default</layout>
      <releases>
        <enabled>true</enabled>
        <updatePolicy>never</updatePolicy>
      </releases>
      <snapshots>
        <enabled>true</enabled>
        <updatePolicy>daily</updatePolicy>
      </snapshots>
    </repository>
  </repositories>

  <distributionManagement>
    <repository>
      <id>local-sramp-repo</id>
      <name>Local S-RAMP Releases Repository</name>
      <url>http://localhost:8080/s-ramp-server/maven/repository</url>
    </repository>
    <snapshotRepository>
      <id>local-sramp-repo-snapshots</id>
      <name>Local S-RAMP Snapshots Repository</name>
      <url>http://localhost:8080/s-ramp-server/maven/repository</url>
    </snapshotRepository>
  </distributionManagement>
```

Once this configuration is complete, you should be able to both deploy to the S-RAMP repository (requires authentication - see below) and pull in dependencies from the S-RAMP repository (does not require authentication).

11.6. A Note About Authentication

Whenever the S-RAMP Maven integration features are used, it is likely that you will need to provide valid authentication credentials. There are two available mechanisms to provide these credentials. First, you may provide the S-RAMP repository username and password in the *Maven settings.xml* [http://maven.apache.org/settings.html] file. If no credentials are found there, then you will be prompted to enter them when they are needed during the build.

An example of providing credentials in the settings.xml file:

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
                      http://maven.apache.org/xsd/settings-1.0.0.xsd">
  <servers>
```

```
    <server>
      <id>local-sramp-repo</id>
      <username>admin</username>
      <password>ADMIN_PASSWORD</password>
    </server>
    <server>
      <id>local-sramp-repo-snapshots</id>
      <username>admin</username>
      <password>ADMIN_PASSWORD</password>
    </server>
  </servers>
</settings>
```

11.7. Maven Integration in the CLI

Note: For more general information about the S-RAMP Shell please see the S-RAMP CLI chapter in this guide.

Another available mechanism for integrating with maven is the S-RAMP CLI's "maven" command namespace. For help on the maven commands in the CLI, run the S-RAMP shell (sramp.sh) and type the following from the resulting prompt:

```
help maven
```

Using the maven CLI commands is often a good choice if you wish to incorporate maven related S-RAMP operations into a script of some kind.

Chapter 12. S-RAMP Samples

Chapter 13. Overlord S-RAMP Server Configuration

Out-of-the-box, Overlord S-RAMP provides a useful, default server configuration. However, if you'd like to mold it into an existing setup, here are a few areas that can be modified.

13.1. Datasource

By default, S-RAMP uses a simple file-based H2 datasource. However, any existing datasource can be used. Swap-out the datasource's JNDI name in the following locations, based on your platform:

13.1.1. JBoss EAP 6

- Standalone modes: JBOSS_HOME/standalone/configuration/standalone*.xml
- Domain mode: JBOSS_HOME/domain/configuration/domain.xml (note that the cache-container will need updated for ALL profiles)

```
      ...
      <cache-container name="modeshape">
        <local-cache name="sramp">
          <locking isolation="NONE"/>
          <transaction mode="NON_XA"/>
          <string-keyed-jdbc-store purge="false" passivation="false"
datasource="[DATASOURCE JNDI NAME]">
            <string-keyed-table prefix="ispn_bucket">
              <id-column type="VARCHAR(500)" name="id"/>
              <data-column type="VARBINARY(60000)" name="datum"/>
              <timestamp-column type="BIGINT" name="version"/>
            </string-keyed-table>
          </string-keyed-jdbc-store>
        </local-cache>
      </cache-container>
      ...
```

13.1.2. Tomcat 7

TOMCAT_HOME/conf/sramp-modeshape.json:

```
    ...
    },"storage": {
        "cacheName": "sramp",
```

```
        "cacheConfiguration" : "infinispan-configuration-webapp.xml",
        "binaryStorage": {
            "type"  : "database",
            "dataSourceJndiName" : "[DATASOURCE JNDI NAME]",
            "minimumBinarySizeInBytes" :
 "${application.min.binary.size:4096}"
        }
    },
    ...
```

13.1.3. Jetty 8

JETTY_HOME/etc/sramp-modeshape.json:

```
    ...
    },"storage": {
        "cacheName": "sramp",
        "cacheConfiguration" : "infinispan-configuration-webapp.xml",
        "binaryStorage": {
            "type"  : "database",
            "dataSourceJndiName" : "[DATASOURCE JNDI NAME]",
            "minimumBinarySizeInBytes" :
 "${application.min.binary.size:4096}"
        }
    },
    ...
```

13.1.4. JBoss Fuse 6.1

- Standalone mode: FUSE_HOME/etc/sramp-modeshape.json
- Fuse Fabric: FUSE_HOME/fabric/import/fabric/configs/versions/1.0/profiles/overlord/sramp.profile/sramp-modeshape.json

```
    ...
    },"storage": {
        "cacheName": "sramp",
        "cacheConfiguration" : "infinispan-configuration-webapp.xml",
        "binaryStorage": {
            "type"  : "database",
            "dataSourceJndiName" : "osgi:service/javax.sql.DataSource/
(osgi.jndi.service.name=[DATASOURCE JNDI NAME])",
            "minimumBinarySizeInBytes" :
 "${application.min.binary.size:4096}"
        }
    },
    ...
```

13.2. WARNINGS

- Between runs, the ModeShape repository name cannot be changed. The name is used on multiple paths used for binary storage, search indexes, etc. Attempting to do so, without changing all necessary paths, will result in ModeShape failures during startup.
- Similarly, between runs, the Infinispan "local-cache" name (and accompanying "cacheName" in the ModeShape config) cannot be changed. ModeShape includes a hash of the cache name within its node IDs, so changing the name is guaranteed to break your repository.

Bibliography

Books

[walsh-muellner] Norman Walsh & Leonard Muellner. *DocBook - The Definitive Guide*. O'Reilly & Associates. 1999. ISBN 1-56592-580-7.