Artificer User Guide

Chapter 1. Introduction to Artificer

1.1. Why?

All individuals, teams, and organizations tend to have a tangled mess of "stuff". That bucket can include bits of information, logical metadata, and physical files. Those "artifacts" are almost never isolated in nature. They're all connected and inter-dependent, but the relationships can be difficult to understand.

In the software development world, this is an especially important problem to solve. The development process often spews out a huge amount of artifacts, needed for future analysis and actions. Without the ability to analyze how the information, artifacts, and content within the artifacts are connected, development processes become difficult, at best, or nearly impossible, at worst. Further, it's not enough to simply know how the bits are related. How do you correlate the artifacts/metadata with, for example, service endpoints, the responsible teams/individuals, and change histories?

Both the publisher and the consumer need help!

1.2. Overview

In steps Artificer. Artificer is a software artifact, logical metadata, and information repository. It consists of a common data model, multiple interfaces, useful tools, and exensibility. In less words? It's a powerful platform that **untangles everything**.

1.3. Use Cases

More specifically, here are a few brief examples of how Artificer can be used. Obviously, this is not exhaustive!

- Software artifact and metadata repository (publish and consume)

- Software lifecycle management and workflows

- Impact analysis

- Discovery and reuse

- Relationship comprehension and hierarchical analysis

- Common configuration and policy management

- Data auditing

- Other "out of the box" uses:

  - "Getting Things Done"

- Reference system

- Project and action lists (with hierarchical classifications)

For more details on specific use cases, take a look at Artificer's available *blog posts* [https://developer.jboss.org/en/artificer/blog]. We plan to periodically provide in-depth walkthroughs.

1.4. Benefits

Although all the features are described in detail, further in the guide, here's a brief overview of what Artificer provides. The bottom line is that Artificer is **not** "just another software repository"!

- Automatically extracts content from archives (ZIPs, JARs, WARs, etc.)

- Automatically derives content from documents — examples:

  - Type declarations in XSDs

  - Messages/ports/endpoints in WSDLs

  - Many others

- Automatically detects content type

- Powerful query language

- Many interfaces & integrations

  - REST/Atom

  - UI

  - Java client

  - CLI

  - Maven (+ Gradle and other Maven dependency based systems)

  - EJB

  - JMS

  - Others under evaluation

- Hierarchical classifiers/ontologies

- Powerful & rich built-in/modeled metadata and relationships

- Logical information models (services, organizations, etc.)

- Ad-hoc metadata and relationships

- Model does not have to be defined at compile time — can be completely ad-hoc

- Full-blown auditing of historical changes

- Easily extended

- Performance — supports enterprise-level usage

1.5. The S-RAMP Specification

Much of Artificer's capabilities are defined by the OASIS S-RAMP specification. S-RAMP is the *SOA Repository Artifact Model and Protocol* [https://www.oasis-open.org/committees/s-ramp/charter.php], a newer specification provided by an 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*

Two of the developers on the project currently participate in the Technical Committee, one of which is the co-chair.

It's important to note that Artificer is not simply an S-RAMP implementation! The capabilities extend well beyond what's required by the spec. Further, although historically Artificer was 100% compliant with the spec, we've recently begun to break compliance where necessary. In our opinion, flexibility, extension, and well-thought-out rules trump spec compliancy, at least in this unique case.

1.6. Core Properties

All artifacts in Artificer 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).

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.

1.7. Custom Properties

An artifact may have additional properties set 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.

1.8. Classifiers

Another type of metadata found on artifacts are "classifiers". Classifiers are a lot like keywords or tags except that they are **hierarchical**. Classifiers are defined in the repository through the OWL Lite format, a subset of the Web Ontology Language.

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.

1.9. Relationships

The final bit of metadata that can be found on an artifact is relationships. These are uni-directional links between a source artifact and a target artifact. Artificer automatically defines a handful of useful relationships on its own, but artifacts can also have arbitrary, client-defined, ad hoc relationships defined during runtime. All relationships have both 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 Artificer application is written in Java. To get started make sure your system has the following:

- Java JDK 1.7 or newer

- Maven 3.0.3 or newer to build and run the examples

- EE application sever: WildFly (*http://www.wildfly.org/downloads*) or JBoss EAP (*http://www.jboss.org/jbossas/downloads*)

2.2. Download

The `artificer-<version>.zip` (or tar.gz) archive can be downloaded from the *website* [http://artificer.jboss.org/downloads.html] website. Grab the latest and extract it.

2.3. Artificer Installation

From the unpacked distribution, run:

```
./install.sh
or
install.bat
```

Alternatively, if you have Apache Ant 1.7 (or later) installed, simply run *ant*.

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

- WildFly 8

- JBoss EAP 6.4

Simply follow the installer instructions to install onto the platform of your choice. We recommend installing into a clean version of whatever 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

Finally, please make sure the JBoss 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.

2.4. User Management

The Artificer WARs are protected using web application security mechanisms configured in the web.xml. By default, Artificer uses single-sign-on (SSO) as the actual authentication mechanism. The SSO is provided via integration with the Keycloak framework, a powerful out-of-the-box auth solution. The actual web.xml configuration uses a standard basic security-context, but SSO is provided under-the-hood.

The Artificer distribution ships with a **artificer-realm.json** file that's completely pre-configured and can be directly imported into Keycloak. Startup WildFly/EAP and visit **localhost:8080/auth/admin/master/console/#/create/realm**. The initial administrator account uses "admin" for both the username and password. There, you can upload the **artificer-realm.json** file and tweak the realm/accounts. Note that you can also import the realm the first time you start up WildFly/EAP. Simply include the following argument:

```
bin/standalone.sh -c standalone-full.xml -Dkeycloak.import=[ARTIFICER HOME]/
artificer-realm.json
```

By default, the realm import creates an "admin" user (password: "artificer1!"). This user is used to access the Artificer server, UI, shell, and other tools. Again, the user is configurable through the Keycloak admin console.

Feel free to manually create a realm for scratch. However, there are a few requirements (see **artificer-realm.json** for example values):

1. The realm must be named **artificer**

2. Defined applications must include **artificer-server** and **artificer-ui**

3. To use our themed login page, use **artificer** for the **loginTheme**

4. Each user must have at least **admin** and **user** values for the **realmRole**

2.5. DDL

Artificer uses RDBMS (through JPA) for all persistence. The distribution ships with DDL SQL files for the "big 5" databases: Postgres, MySQL, Oracle, SQL Server, and DB2/IBM. In addition, H2 is included (the default DB in WildFly/EAP).

During installation, a simple, file-based H2 datasource is automatically deployed. (see `$JBOSS_HOME/standalone/deployments/artificer-h2-ds.xml`). If you use something

other than that datasource, be sure to update the relevant connection properties in `artificer.properties`. See the "Artificer Server Configuration" section for more details.

To install the DDL, you have two options. Obviously, manually importing it, using your favorite SQL client, is great. We're fans of *SQuirreL* [http://www.squirrelsql.org/]). As an example, if you install SQuirreL and its H2 plugin, then start WildFly/EAP with Artificer installed, you could connect to its H2 datasource with the following connection URL. Use it to import the DDL SQL.

```
jdbc:h2:[JBOSS_HOME]/standalone/data/h2/artificer;mvcc=true
```

Alternatively, Artificer will automatically install the DDL for you. On startup, it checks to see if the necessary tables exist. If not, it will automatically execute the DDL SQL for the DB you've selected in `artificer.properties`.

2.6. Startup

Once Artificer is installed and configured on your preferred platform, you should be able to go ahead and start it up.

WildFly & EAP (standalone-full is required)

```
bin/standalone.sh -c standalone-full.xml
```

2.7. Check your Installation

To make sure your installation works you can fire up the *artificer-ui* [http://localhost:8080/artificer-ui]. By default, the username is "admin" and the password "artificer1!". 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 artificer-ui.

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

Alternatively you can fire up the `artificer shell` in the `bin` directory of the distribution:

```
./bin/artificer.sh
```

To connect the shell to the server type `connect` and hit the tab key. It should auto-complete to say `connect http://localhost:8080/artificer-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:

```
artificer> 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. 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, as well as additional models provided by Artificer.

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.

3.1. 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

3.2. 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

3.3. 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

3.4. 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>	

3.5. 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		
Choreography		
ChoreographyProcess		
Collaboration		
CollaborationProcess		
Composition		
Effect		
Element		
Event		
InformationType		
Orchestration		
OrchestrationProcess		
Policy		
PolicySubject		
Process		
Service		
ServiceContract		
ServiceComposition		
ServiceInterface		
System		
Task		

3.6. 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

Artifact Type	Parent Type	Properties
ServiceOperation		end, url

3.7. Custom/Extension Data Models (ext)

Clients can define their own ad-hoc data models, during **runtime**, by using the "ext" model space. This allows clients to add both logical and document artifacts with custom artifact types. For example, a client can add an artifact to /ext/MyDocumentType. Clients could also specify "MyDocumentType" when creating an artifact through the CLI, UI, etc. — the "/ext" space is automatically assumed.

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 an explicitly definition of the model - it is up to the client to properly conform to their own implicit model.

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

3.8. Java Data Model

The Artificer 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

3.9. KIE Data Model (Knowledge is Everything)

The Artificer 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	
BpmnDocument	ExtendedDocument	bpmn	
DroolsDocument	ExtendedDocument	dtl	

3.10. SwitchYard Data Model

The Artificer server includes 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

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

3.11. 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,

Artifact Type	Parent Type	Properties
		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, 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
TeiidVdbDataPolicy	TeiidVdbDataPolicy	TeiidVdbPermission	1 to many

Relationship Type	Source Type	Target Type	Multiplicity
TeiidVdbPermissionDataPolicy	TeiidVdbPermission	TeiidVdbDataPolicy	1 to 1
TeiidVdbSchemaSources	TeiidVdbSchema	TeiidVdbSource	1 to 1
TeiidVdbSchemaValidationErrors	TeiidVdbSchema	TeiidVdbValidationError	1 to many
TeiidVdbSourceSchema	TeiidVdbSource	TeiidVdbSchema	1 to many
TeiidVdbSourceTranslator	TeiidVdbSource	TeiidVdbTranslator	1 to 1
TeiidVdbTranslatorSources	TeiidVdbTranslator	TeiidVdbSource	1 to many
TeiidVdbValidationSource	TeiidVdbValidation	TeiidVdbSource	1 to 1
expandedFromDocument	TeiidVdbDataPolicy, TeiidVdbEntry, TeiidVdbSchema, TeiidVdbTranslator, TeiidVdbImportVdb	TeiidVdbManifest	many to 1

Chapter 4. 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 *Full-text searches on both metadata and content

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[@derived = *false* and @expandedFromArchive=*false*]	All "original" artifacts that were not expanded from an archive artifact.
/s-ramp[expandedFromArchive[@uuid='12345']]	All artifacts that were expanded from an archive artifact with UUID *12345*.
/s-ramp/wsdl/Message[xp2:matches(@name, 'get.*')]/part[element]	Element style WSDL parts from WSDL messages with names starting with *get*.

In addition, Artificer provides extensions to the S-RAMP query language, providing extra capabilities.

Query	What It Selects
/s-ramp/xsd/XsdDocument[xp2:matches(., '.**foo**.')]	Full-text search: XSD artifacts where "foo" appears in the metadata or content
/s-ramp/xsd/ XsdDocument[someRelationship[s-ramp:getRelationshipAttribute(., *someName*)]]	XSD artifacts that contain a generic relationship named *someRelationship*, and that relationship has an *otherAttribute* named *someName*.
/s-ramp/xsd/ XsdDocument[someRelationship[s-ramp:getRelationshipAttribute(., *someName*) = *someValue*]]	XSD artifacts that contain a generic relationship named *someRelationship*, and that relationship has an *otherAttribute* named *someName* with a *someValue* value.
/s-ramp/xsd/ XsdDocument[someRelationship[s-ramp:getTargetAttribute(., *someName*)]]	XSD artifacts that contain a generic relationship named *someRelationship*, and that relationship's target has an *otherAttribute* named *someName*.
/s-ramp/xsd/ XsdDocument[someRelationship[s-ramp:getTargetAttribute(., *someName*) = *someValue*]]	XSD artifacts that contain a generic relationship named *someRelationship*, and that relationship's target has an *otherAttribute* named *someName* with a *someValue* value.

Chapter 5. Artificer REST API Endpoints

The intent of the S-RAMP specification is to outline the 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.

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 the spec's notional format, as well as Artificer-specific extensions. But it bears repeating that for any given server implementation, a client should first query the *Atom service document*.

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.

The following table lists the endpoints available in the Artificer 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*
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/query	*Create Stored Query*
PUT /s-ramp/query/{queryName}	*Update Stored Query*
GET /s-ramp/query	*List Stored Queries*
GET /s-ramp/query/{queryName}	*Get Stored Query*
GET /s-ramp/query/{queryName}/results	*Execute Stored Query*

Endpoint	Name
DELETE /s-ramp/query/{queryName}	*Delete Stored Query*
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*

5.1. API: Get Service Document

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

Retrieves the service document.

HTTP Method	Request	Response
GET	N/A	Atom Service Document

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.

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:2013:type" term="Document"/>
```

```
                <atom:category label="XML Document" scheme="urn:x-s-
    ramp:2013: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:2013: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:2013: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.

5.2. Artifacts

5.2.1. API: Publish Artifact

Publishes a new artifact into the repository. 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).

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

5.2.1.1. 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:2013:type"
  term="Document" />
```

```
  <atom:category label="Document" scheme="x-s-ramp:2013: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>
```

5.2.1.2. 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:2013:type"
 term="MavenPom" />
  <atom:category label="Extended Document" scheme="x-s-ramp:2013: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>
```

5.2.1.3. 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:2013:type"
 term="Document" />
  <atom:category label="Document" scheme="x-s-ramp:2013: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>
```

5.2.2. 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>
```

5.2.3. 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/artificer-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/artificer-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/artificer-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:2013:type" term="ComplexTypeDeclaration"/>
<atom:category label="XML Schema Complex Type Declaration" scheme="x-s-
ramp:2013: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>
```

5.2.4. 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.
```

5.2.5. 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
```

5.2.6. Queries

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.

5.2.6.1. 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="Artificer" 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:2013:type"
 term="Document" />
    <atom:category label="Document" scheme="x-s-ramp:2013: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:2013:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2013: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:2013:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2013: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:2013:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2013: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:2013:type"
 term="XmlDocument" />
```

```
    <atom:category label="XML Document" scheme="x-s-ramp:2013: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>
```

5.2.6.2. 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="Artificer" 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:2013:type"
 term="Document" />
    <atom:category label="Document" scheme="x-s-ramp:2013: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:2013:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2013: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:2013:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2013: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:2013:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2013: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:2013:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2013: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>
```

5.2.6.3. API: Query (GET)

```
    /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="Artificer" 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:2013:type"
  term="Document" />
    <atom:category label="Document" scheme="x-s-ramp:2013: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:2013:type"
 term="XmlDocument" />
      <atom:category label="XML Document" scheme="x-s-ramp:2013: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:2013:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2013: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:2013:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2013: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:2013:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2013: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>
```

5.2.6.4. API: Query (POST)

```
   /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="Artificer" 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:2013:type"
 term="Document" />
    <atom:category label="Document" scheme="x-s-ramp:2013: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:2013:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2013: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:2013:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2013: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:2013:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2013: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:2013:type"
 term="XmlDocument" />
    <atom:category label="XML Document" scheme="x-s-ramp:2013: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>
```

5.2.6.5. 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
```

5.2.7. Stored Queries

S-RAMP queries can be utilized through the "stored query" concept. The query is persisted within the S-RAMP repository and can be repeatedly executed by name.

5.2.7.1. API: Create Stored Query

```
/s-ramp/query
```

Creates a new stored query in the repository. The body of the request must be the stored query, wrapped as an Atom Entry. The response is also an Atom Entry containing additional, server-generated meta-data.

HTTP Method	Request	Response
POST	Atom Entry	Atom Entry

Example Request

```
POST /s-ramp/query 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">
  <s-ramp:storedQueryData>
    <s-ramp:queryName>FooQuery</s-ramp:queryName>
    <s-ramp:queryString>/s-ramp/ext/FooType</s-ramp:queryString>
    <s-ramp:propertyName>importantProperty1</s-ramp:propertyName>
    <s-ramp:propertyName>importantProperty2</s-ramp:propertyName>
  </s-ramp:storedQueryData>
</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">
  <atom:title>Stored Query: FooQuery</atom:title>
  <atom:link href="http://localhost:9093/artificer-server/s-ramp/query/
FooQuery"
    rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
  <atom:link href="http://localhost:9093/artificer-server/s-ramp/query/
FooQuery"
    rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
  <atom:link
    href="http://localhost:9093/artificer-server/s-ramp/query/FooQuery/
results"
    rel="urn:x-s-ramp:2013:query:results" type="application/atom
+xml;type=&quot;feed&quot;" />
  <atom:category label="Stored Query Entry" scheme="urn:x-s-ramp:2013:type"
    term="query" />
  <atom:id>urn:uuid:FooQuery</atom:id>
  <atom:content>Stored Query Entry</atom:content>
  <s-ramp:storedQueryData>
    <s-ramp:queryName>FooQuery</s-ramp:queryName>
    <s-ramp:queryString>/s-ramp/ext/FooType</s-ramp:queryString>
    <s-ramp:propertyName>importantProperty1</s-ramp:propertyName>
    <s-ramp:propertyName>importantProperty2</s-ramp:propertyName>
  </s-ramp:storedQueryData>
</atom:entry>
```

5.2.7.2. API: Update Stored Query

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

Updates the given stored query in the repository. The body of the request is the same Atom Entry as in *Create Stored Query*.

HTTP Method	Request	Response
PUT	Atom Entry	N/A

5.2.7.3. API: List Stored Queries

```
/s-ramp/query
```

Retrieves all stored queries from the repository.

HTTP Method	Request	Response
GET	N/A	Atom Feed

Example Response

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<atom:feed 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>S-RAMP stored queries feed</atom:title>
  <atom:updated>2014-09-25T16:45:10.133-04:00</atom:updated>
  <atom:entry>
    <atom:title>Stored Query: FooQuery</atom:title>
    <atom:link
      href="http://localhost:9093/artificer-server/s-ramp/query/FooQuery"
      rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link
      href="http://localhost:9093/artificer-server/s-ramp/query/FooQuery"
      rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link
      href="http://localhost:9093/artificer-server/s-ramp/query/FooQuery/
results"
      rel="urn:x-s-ramp:2013:query:results" type="application/atom
+xml;type=&quot;feed&quot;" />
    <atom:category label="Stored Query Entry" scheme="urn:x-s-
ramp:2013:type"
      term="query" />
    <atom:id>urn:uuid:FooQuery</atom:id>
    <atom:content>Stored Query Entry</atom:content>
    <s-ramp:storedQueryData>
      <s-ramp:queryName>FooQuery</s-ramp:queryName>
      <s-ramp:queryString>/s-ramp/ext/FooType</s-ramp:queryString>
      <s-ramp:propertyName>importantProperty1</s-ramp:propertyName>
      <s-ramp:propertyName>importantProperty2</s-ramp:propertyName>
    </s-ramp:storedQueryData>
  </atom:entry>
  <atom:entry>
    <atom:title>Stored Query: FooQuery2</atom:title>
    <atom:link
      href="http://localhost:9093/artificer-server/s-ramp/query/FooQuery2"
      rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link
      href="http://localhost:9093/artificer-server/s-ramp/query/FooQuery2"
      rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
```

```
        <atom:link
          href="http://localhost:9093/artificer-server/s-ramp/query/FooQuery2/
results"
          rel="urn:x-s-ramp:2013:query:results" type="application/atom
+xml;type=&quot;feed&quot;" />
        <atom:category label="Stored Query Entry" scheme="urn:x-s-
ramp:2013:type"
          term="query" />
        <atom:id>urn:uuid:FooQuery2</atom:id>
        <atom:content>Stored Query Entry</atom:content>
        <s-ramp:storedQueryData>
          <s-ramp:queryName>FooQuery2</s-ramp:queryName>
          <s-ramp:queryString>/s-ramp/ext/FooType</s-ramp:queryString>
        </s-ramp:storedQueryData>
    </atom:entry>
</atom:feed>
```

5.2.7.4. API: Get Stored Query

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

Retrieves the given stored query from the repository.

HTTP Method	Request	Response
GET	N/A	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">
  <atom:title>Stored Query: FooQuery</atom:title>
  <atom:link href="http://localhost:9093/artificer-server/s-ramp/query/
FooQuery"
    rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
  <atom:link href="http://localhost:9093/artificer-server/s-ramp/query/
FooQuery"
    rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
  <atom:link
    href="http://localhost:9093/artificer-server/s-ramp/query/FooQuery/
results"
    rel="urn:x-s-ramp:2013:query:results" type="application/atom
+xml;type=&quot;feed&quot;" />
  <atom:category label="Stored Query Entry" scheme="urn:x-s-ramp:2013:type"
```

```
    term="query" />
  <atom:id>urn:uuid:FooQuery</atom:id>
  <atom:content>Stored Query Entry</atom:content>
  <s-ramp:storedQueryData>
    <s-ramp:queryName>FooQuery</s-ramp:queryName>
    <s-ramp:queryString>/s-ramp/ext/FooType</s-ramp:queryString>
    <s-ramp:propertyName>importantProperty1</s-ramp:propertyName>
    <s-ramp:propertyName>importantProperty2</s-ramp:propertyName>
  </s-ramp:storedQueryData>
</atom:entry>
```

5.2.7.5. API: Execute Stored Query

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

Similar to a normal, ad-hoc *Query*, this returns an Atom Feed of artifact entries.

HTTP Method	Request	Response
GET	N/A	Atom Feed

Example Request

```
GET /artificer-server/s-ramp/query/FooQuery/results?
startIndex=0&count=20&orderBy=name&ascending=true HTTP/1.1
```

Note that the paging and ordering parameters are optional and have reasonable defaults (defaults are shown values).

Example Response

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<atom:feed 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:provider="Artificer" s-ramp:itemsPerPage="20"
  s-ramp:startIndex="0" s-ramp:totalResults="2">
  <atom:title>S-RAMP Feed</atom:title>
  <atom:subtitle>Ad Hoc query feed</atom:subtitle>
  <atom:updated>2014-09-25T17:03:32.504-04:00</atom:updated>
  <atom:id>b4746dcd-82b9-433a-9d61-e54a384ae4e6</atom:id>
  <atom:author>
    <atom:name>anonymous</atom:name>
  </atom:author>
  <atom:entry s-ramp:derived="false" s-ramp:extendedType="FooType">
    <atom:title>FooArtifact</atom:title>
```

```
    <atom:link
      href="http://localhost:9093/artificer-server/s-ramp/ext/FooType/
f57c30d3-9a9a-4df9-b362-8f0b0816af99"
      rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link
      href="http://localhost:9093/artificer-server/s-ramp/ext/FooType/
f57c30d3-9a9a-4df9-b362-8f0b0816af99/media"
      rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link
      href="http://localhost:9093/artificer-server/s-ramp/ext/FooType/
f57c30d3-9a9a-4df9-b362-8f0b0816af99"
      rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:category label="Extended Artifact Type" scheme="x-s-
ramp:2013:type"
      term="FooType" />
    <atom:category label="Extended Artifact Type" scheme="x-s-
ramp:2013:model"
      term="ext" />
    <atom:updated>2014-09-25T17:03:27.392-04:00</atom:updated>
    <atom:id>f57c30d3-9a9a-4df9-b362-8f0b0816af99</atom:id>
    <atom:published>2014-09-25T17:03:27.392-04:00</atom:published>
    <atom:author>
      <atom:name>asdf</atom:name>
    </atom:author>
    <s-ramp:artifact>
      <s-ramp:ExtendedArtifactType />
    </s-ramp:artifact>
  </atom:entry>
  <atom:entry s-ramp:derived="false" s-ramp:extendedType="FooType">
    <atom:title>FooArtifact2</atom:title>
    <atom:link
      href="http://localhost:9093/artificer-server/s-ramp/ext/FooType/
e8b6aaf2-d787-45d0-a534-b9205d6e8815"
      rel="self" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link
      href="http://localhost:9093/artificer-server/s-ramp/ext/FooType/
e8b6aaf2-d787-45d0-a534-b9205d6e8815/media"
      rel="edit-media" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:link
      href="http://localhost:9093/artificer-server/s-ramp/ext/FooType/
e8b6aaf2-d787-45d0-a534-b9205d6e8815"
      rel="edit" type="application/atom+xml;type=&quot;entry&quot;" />
    <atom:category label="Extended Artifact Type" scheme="x-s-
ramp:2013:type"
      term="FooType" />
    <atom:category label="Extended Artifact Type" scheme="x-s-
ramp:2013:model"
      term="ext" />
    <atom:updated>2014-09-25T17:03:29.580-04:00</atom:updated>
```

```
      <atom:id>e8b6aaf2-d787-45d0-a534-b9205d6e8815</atom:id>
      <atom:published>2014-09-25T17:03:29.580-04:00</atom:published>
      <atom:author>
        <atom:name>asdf</atom:name>
      </atom:author>
      <s-ramp:artifact>
        <s-ramp:ExtendedArtifactType />
      </s-ramp:artifact>
    </atom:entry>
  </atom:feed>
```

5.2.7.6. API: Delete Stored Query

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

Deletes the given stored query from the repository.

HTTP Method	Request	Response
DELETE	N/A	N/A

5.3. Ontologies

5.3.1. 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>
```

5.3.2. 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>
```

5.3.3. 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
```

5.3.4. 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>
```

5.3.5. 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
```

5.4. Auditing

5.4.1. 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="Artificer" 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>
```

5.4.2. 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="Artificer" 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>
```

5.4.3. 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>
```

5.4.4. 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/artificer/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 6. Clients

6.1. Java Client

The Artificer implementation provides a full-featured 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.

6.1.1. Basic Usage

The Artificer 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.artificer</groupId>
  <artifactId>artificer-client</artifactId>
  <version>${artificer.client.version}</version>
</dependency>
```

Once the library is included in your project, you can use the client by instantiating the **ArtificerAtomApiClient** 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 Artificer.

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

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

```
ArtificerAtomApiClient client = new ArtificerAtomApiClient(urlToArtificer);
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.

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

Retrieve artifact content.

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

Query the Artificer repository (by artifact name).

```
ArtificerAtomApiClient client = new ArtificerAtomApiClient(urlToArtificer);
String artifactName = getArtifactName();
QueryResultSet rset = client.buildQuery("/s-ramp/xsd/XsdDocument[@name
 = ?]")
        .parameter(artifactName)
        .count(10)
        .query();
```

Query the Artificer repository using a stored query.

```
ArtificerAtomApiClient client = new ArtificerAtomApiClient(urlToArtificer);

StoredQuery storedQuery = new StoredQuery();
storedQuery.setQueryName("FooQuery");
storedQuery.setQueryExpression("/s-ramp/ext/FooType");
storedQuery.getPropertyName().add("importantProperty1");
storedQuery.getPropertyName().add("importantProperty2");
client.createStoredQuery(storedQuery);

QueryResultSet rset =
 client.queryWithStoredQuery(storedQuery.getQueryName());
```

6.1.2. Extended Feature: Ontologies

Although the S-RAMP specification is silent on how the API should support the management of ontologies, the Artificer 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 Artificer 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 Artificer repository.

6.1.3. Extended Feature: Auditing

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

6.1.4. Important Notes

6.1.4.1. WildFly and EAP

When using the Java client from within a webapp running on WildFly or EAP, the *artificer-client* Maven dependency carries transitive dependencies that conflict with modules on the app server. At this time, the worst known offender is *jaxb-impl* (causes a PropertyException related to RESTEasy's use of *NamespacePrefixMapper*). It's vital to do the following in your webapp's POM.

Exclude *jaxb-impl* from the *artificer-client* dependency:

```
<dependency>
  <groupId>org.artificer</groupId>
  <artifactId>artificer-client</artifactId>
  <version>[version]</version>

  <exclusions>
    <exclusion>
      <groupId>com.sun.xml.bind</groupId>
      <artifactId>jaxb-impl</artifactId>
    </exclusion>
  </exclusions>
</dependency>
```

Include the *com.sun.xml.bind* module in your *maven-war-plugin manifestEntries*:

```
<plugin>
  <artifactId>maven-war-plugin</artifactId>
  <version>2.3</version>
  <configuration>
    <archive>
      <manifestEntries>
        <Dependencies>com.sun.xml.bind,...</Dependencies>
      </manifestEntries>
    </archive>
  </configuration>
</plugin>
```

6.2. EJB Client

The logic and actions that back all of the REST services are available for direct use through EJB, for both local server and remote client use. This should be the top choice for client interactivity where performance is a major consideration, as it removes the typical REST bottlenecks.

To use it, you'll need to add the following dependencies:

```
<dependency>
  <groupId>org.artificer</groupId>
  <artifactId>artificer-server-api</artifactId>
  <version>[ARTIFICER VERSION]</version>
</dependency>
<dependency>
  <groupId>org.wildfly</groupId>
  <artifactId>wildfly-ejb-client-bom</artifactId>
  <version>8.2.0.Final</version>
  <type>pom</type>
  <scope>runtime</scope>
</dependency>
```

There are a couple of things to note with the dependencies. 1.) We're "cheating" and using the *wildfly-ejb-client-bom* to pull in quite a bit. With out it, you'll need the EJB API, JTA API, etc. 2.) xercesImpl is currently required during runtime, mostly due to XMLGregorianCalendarImpl use during (un)marshalling.

Then, interacting with Artificer is as simple as:

```
ExtendedArtifactType artifact = new ExtendedArtifactType();
artifact.setArtifactType(BaseArtifactEnum.EXTENDED_ARTIFACT_TYPE);
artifact.setExtendedType("FooArtifactType");
artifact.setName("Foo");
artifact.setDescription("I'm a Foo");

try {
    Properties jndiProps = new Properties();
    jndiProps.put(Context.INITIAL_CONTEXT_FACTORY,
 "org.jboss.naming.remote.client.InitialContextFactory");
    jndiProps.put(Context.PROVIDER_URL,"http-remoting://localhost:8080");
    jndiProps.put("jboss.naming.client.ejb.context", true);
    Context context = new InitialContext(jndiProps);

    final ArtifactService artifactService =  (ArtifactService)
 context.lookup(
            "artificer-server/ArtifactService!" +
 ArtifactService.class.getName());
    artifactService.login("artificer", "artificer1!");
    artifactService.create(artifact);

    final QueryService queryService =  (QueryService) context.lookup(
            "artificer-server/QueryService!" +
 QueryService.class.getName());
    queryService.login("artificer", "artificer1!");
    ArtifactSet artifactSet = queryService.query("/s-ramp/ext/
 FooArtifactType");
```

```
    Iterator<BaseArtifactType> iterator = artifactSet.iterator();
    while (iterator.hasNext()) {
        BaseArtifactType artifactResult = iterator.next();
        System.out.println(artifactResult.getName());
    }
} catch (Exception e) {
    e.printStackTrace();
}
```

The complete list of services include the following. Have a look at their javadocs — the capabilities are fairly extensive. * org.artificer.server.core.api.ArtifactService * org.artificer.server.core.api.AuditService * org.artificer.server.core.api.BatchService * org.artificer.server.core.api.OntologyService * org.artificer.server.core.api.QueryService

Note that you must call #login for each service, using the EJB/JMS username and password that you provided during installation!

6.3. JMS Client

Artificer publishes JMS messages to both topics and queues for several types of events. The type of event is designated by the JMSType header field. All events carry the relevant object marshalled into a JSON payload.

6.3.1. Installation and Setup

The *artificer.properties* configuration file contains multiple properties relevant to the JMS setup:

```
# Artificer will automatically attempt to discover a JMS ConnectionFactory
 through the literal JNDI name
# "ConnectionFactory".  However, that name can be overridden here.
artificer.config.events.jms.connectionfactory = ConnectionFactory
# By default, Artificer publishes events through the "artificer/events/
topic" JMS topic name (JNDI).  But, it will also publish
# to any other names listed here (comma-delimited).
artificer.config.events.jms.topics = artificer/events/topic
# In addition to the above topics, Artificer will also publish non-expiring
 events to any JMS queue names (JNDI)
# listed here (comma-delimited).
artificer.config.events.jms.queues =
```

Artificer supports two JMS environments:

- When Artificer is installed in WildFly/EAP by using our installation script, JMS is configured automatically. The existing HornetQ configuration is modified to add the default topic, described above, and all necessary credentials. Users can add additional topics/queues to their framework, then add them to *artificer.properties* (see above). IMPORTANT: In order for HornetQ to work properly, the standalone-full profile must be used (*bin/standalone.sh -c standalone-full.xml*)! Without it, errors are guaranteed to occur during startup!

• For other EE platforms, Artificer will always attempt to discover a JMS *ConnectionFactory* and all configured topics/queues through JNDI. If found, it will simply use that existing framework and setup. Users can add additional topics/queues to their framework, then add them to *artificer.properties* (see above).

6.3.2. Authorization

During installation, you were prompted for a password. This set up a standard WildFly/EAP **admin** user (including the **artificer** role used by the HornetQ configuration in standalone*.xml). These credentials must be used when connecting to the JMS topics/queues as a subscriber!

6.3.3. Artifact JMS Events

Event	JMSType Header	Payload
Artifact Created	artificer:artifactCreated	Artifact JSON
Artifact Updated	artificer:artifactUpdated	Old/New Artifacts JSON
Artifact Deleted	artificer:artifactDeleted	Artifact JSON

These events carry the artifacts, marshalled into JSON, as payloads. Note that these can be easily unmarshalled back into the artificer-api module's Java bindings. Here's a brief example using Jackson:

```
// The TextMessage is received through a typical JMS MessageListener.
TextMessage textMessage = ...;
ObjectMapper mapper = new ObjectMapper();
ExtendedArtifactType eventArtifact = mapper.readValue(textMessage.getText(),
 ExtendedArtifactType.class);
```

Example Artifact Created JSON

```
{
  "classifiedBy":[

  ],
  "relationship":[

  ],
  "property":[

  ],
  "artifactType":"EXTENDED_ARTIFACT_TYPE",
```

```
    "name":"Foo",
    "description":"created",
    "createdBy":"admin",
    "version":null,
    "uuid":"cd0d16c6-cee0-41fa-ad53-47d4e48947fb",
    "createdTimestamp":1411744515668,
    "lastModifiedTimestamp":1411744515668,
    "lastModifiedBy":"admin",
    "otherAttributes":{
      "{http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0}derived":"false",
      "{http://docs.oasis-open.org/s-ramp/ns/s-ramp-
  v1.0}contentType":"application/xml"
    },
    "extendedType":"FooArtifactType"
  }
```

artifactUpdated takes the payload a step further and includes both the original and the revised artifacts.

Example Artifact Updated JSON

```
  {
    "updatedArtifact":{
      "@class":"org.oasis_open.docs.s_ramp.ns.s_ramp_v1.ExtendedArtifactType",
      "classifiedBy":[

      ],
      "relationship":[

      ],
      "property":[

      ],
      "artifactType":"EXTENDED_ARTIFACT_TYPE",
      "name":"Foo",
      "description":"updated",
      "createdBy":"admin",
      "version":null,
      "uuid":"cd0d16c6-cee0-41fa-ad53-47d4e48947fb",
      "createdTimestamp":1411744515668,
      "lastModifiedTimestamp":1411744516142,
      "lastModifiedBy":"admin",
      "otherAttributes":{
        "{http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0}derived":"false",
        "{http://docs.oasis-open.org/s-ramp/ns/s-ramp-
  v1.0}contentType":"application/xml"
      },
      "extendedType":"FooArtifactType"
    },
```

```
    "oldArtifact":{
        "@class":"org.oasis_open.docs.s_ramp.ns.s_ramp_v1.ExtendedArtifactType",
        "classifiedBy":[

        ],
        "relationship":[

        ],
        "property":[

        ],
        "artifactType":"EXTENDED_ARTIFACT_TYPE",
        "name":"Foo",
        "description":"created",
        "createdBy":"admin",
        "version":null,
        "uuid":"cd0d16c6-cee0-41fa-ad53-47d4e48947fb",
        "createdTimestamp":1411744515668,
        "lastModifiedTimestamp":1411744515668,
        "lastModifiedBy":"admin",
        "otherAttributes":{
          "{http://docs.oasis-open.org/s-ramp/ns/s-ramp-v1.0}derived":"false",
          "{http://docs.oasis-open.org/s-ramp/ns/s-ramp-
v1.0}contentType":"application/xml"
        },
        "extendedType":"FooArtifactType"
    }
}
```

6.3.4. Ontology JMS Events

Event	JMSType Header	Payload
Ontology Created	artificer:ontologyCreated	Ontology JSON
Ontology Updated	artificer:ontologyUpdated	Old/New Ontologies JSON
Ontology Deleted	artificer:ontologyDeleted	Ontology JSON

These events work similarly to Artifacts, but carry the ontology payload using the artificer-api module's binding: RDF.

Example Ontology Created JSON

```
{
```

```
  "ontology":{
    "label":"Color",
    "comment":null,
    "id":"Color"
  },
  "clazz":[
    {
      "subClassOf":null,
      "label":"Red",
      "comment":null,
      "id":"Red"
    },
    {
      "subClassOf":null,
      "label":"Blue",
      "comment":null,
      "id":"Blue"
    }
  ],
  "otherAttributes":{
    "{http://www.w3.org/XML/1998/namespace}base":"foo"
  }
}
```

Example Ontology Updated JSON

```
{
  "updatedOntology":{
    "ontology":{
      "label":"ColorUpdated",
      "comment":null,
      "id":"Color"
    },
    "clazz":[
      {
        "subClassOf":null,
        "label":"Red",
        "comment":null,
        "id":"Red"
      },
      {
        "subClassOf":null,
        "label":"Blue",
        "comment":null,
        "id":"Blue"
      }
    ],
    "otherAttributes":{
      "{http://www.w3.org/XML/1998/namespace}base":"foo"
```

```
        }
      },
      "oldOntology":{
        "ontology":{
          "label":"Color",
          "comment":null,
          "id":"Color"
        },
        "clazz":[
          {
            "subClassOf":null,
            "label":"Red",
            "comment":null,
            "id":"Red"
          },
          {
            "subClassOf":null,
            "label":"Blue",
            "comment":null,
            "id":"Blue"
          }
        ],
        "otherAttributes":{
          "{http://www.w3.org/XML/1998/namespace}base":"foo"
        }
      }
    }
```

Chapter 7. Artificer Command Line

Using the Artificer 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

```
./artificer.sh
```

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

7.1. Connecting to Artificer server

To connect the shell to the server type `connect` and hit the tab key. It should auto-complete to say `connect http://localhost:8080/artificer-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 Artificer repository runs elsewhere.

7.2. Browsing the Artificer repository

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

```
artificer> 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

```
artificer> 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
  artificer>
```

7.3. Updating artifact MetaData

7.3.1. Properties

To update a property on the artifact use `property set` and hit the `tab` key

```
  artificer> property set
  description        name              version
```

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

```
  artificer> property set description "BPMN2 artifact representing the
   SimpleReleaseProcess"
  Successfully set property description.
  artificer> updateMetaData
  Successfully updated artifact overlord.demo.SimpleReleaseProcess.bpmn.
```

To verify issue

```
  artificer> 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.

7.3.2. Custom Properties

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

```
artificer> property set kurt stam
Successfully set property kurt.
artificer> updateMetaData
Successfully updated artifact overlord.demo.SimpleReleaseProcess.bpmn.
```

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

```
artificer> 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
artificer>
```

When hitting the `tab` key on `property set` results in

```
artificer> property set
description     kurt            name            version
```

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

7.3.3. Classifications

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

```
artificer> classification add "http://www.jboss.org/overlord/deployment-
status.owl#Dev"
Successfully added classification 'http://www.jboss.org/overlord/deployment-
status.owl#Dev'.
```

```
artificer> updateMetaData
Successfully updated artifact overlord.demo.SimpleReleaseProcess.bpmn.
```

and to verify that it was added

```
artificer> 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
artificer>
```

7.4. Querying the Artificer Repository using XPath2 Syntax

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

```
artificer> 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
artificer>
```

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 `query /s-ramp/wsdl`

```
artificer> query /s-ramp/wsdl/
Binding                 BindingOperation        BindingOperationFault
  BindingOperationInput    BindingOperationOutput
```

```
Fault                    Message                    Operation
 OperationInput           OperationOutput
Part                     Port                       PortType
 WsdlDocument             WsdlExtension
WsdlService
artificer>
```

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

To obtain all Operations in this WSDL use

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

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

```
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
artificer>
```

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

```
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

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

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

```
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.

7.4.1. Stored Queries

The above queries can also be executed using Stored Queries:

```
artificer> createStoredQuery FooQuery /s-ramp/ext/FooType
artificer> executeStoredQuery FooQuery
Querying the S-RAMP repository:
    /s-ramp/ext/FooType
Atom Feed (1 entries)
  Idx                Type Name
  ---                ---- ----
```

```
        1                       FooType FooArtifact
```

7.5. Running Commands in Batch

An interesting thing you can do with the Artificer 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 Artificer CLI to execute the batch. For example, a batch of commands may look like this:

```
# Connect to Artificer
connect http://localhost:8080/artificer-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:

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

7.6. Batch File Property Interpolation

Note that it is possible to use Ant style property replacements within your Artificer 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 Artificer
connect ${artificer.endpoint:http://localhost:8080/artificer-server}
 ${artificer.username:admin} ${artificer.password:admin123!}
upload ${resource.path}
```

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

- resource.path - (required)

- artificer.endpoint - (optional, defaults to *http://localhost:8080/artificer-server*)

- artificer.username - (optional, defaults to admin)

- artificer.password - (optional, defaults to admin123!)

7.7. Log-to-File

Rather than creating batch files by hand, the Artificer 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.

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

Chapter 8. Artificer Maven Integration

8.1. Overview

A key feature of the Artificer project is its integration with Maven. Currently there are two mechanisms provided to integrate with Maven. First, the project provides an HTTP servlet which acts as a facade in front of the Artificer repository. This can be used for dependency retrieval and artifact deployments, just as any other Maven repository. Second, there is a "maven" namespace in the Artificer Shell (CLI) providing integration between the CLI and Maven.

8.2. Deploying to Artificer

The Artificer URL can be directly used in the distributionManagement section. For example:

```
<distributionManagement>
  <repository>
    <id>local-artificer-repo</id>
    <name>Artificer Releases Repository</name>
    <url>http://localhost:8080/artificer-server/maven/repository</url>
  </repository>
  <snapshotRepository>
    <id>local-artificer-repo-snapshots</id>
    <name>Artificer Snapshots Repository</name>
    <url>http://localhost:8080/artificer-server/maven/repository</url>
  </snapshotRepository>
</distributionManagement>
```

With these settings, maven deployments will be sent directly to the Artificer repository using the Artificer API. Note that artifacts will be added to the Artificer 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-artificer-repo</id>
    <name>Artificer Releases Repository</name>
    <url>http://localhost:8080/artificer-server/maven/repository?
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 Artificer:

```
mvn clean deploy
```

8.3. Adding Artificer Artifacts as Dependencies

Additionally, artifacts from the Artificer repository can be used as dependencies in your maven project.

First, the Artificer 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-artificer-repo</id>
    <name>Local Artificer Repository</name>
    <url>http://localhost:8080/artificer-server/maven/repository</url>
    <layout>default</layout>
  </repository>
</repositories>
```

Once the repository is configured, an Artificer artifact can be referenced as a dependency in two ways. First, if the artifact was added to Artificer using the maven integration to deploy it, then the artifact in Artificer 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.artificer.wiki</groupId>
  <artifactId>artificer-wiki-example</artifactId>
  <version>1.0</version>
</dependency>
```

However, even if an artifact was added to the Artificer 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 Artificer 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 Artificer 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>
```

8.4. Authentication

Whenever the Artificer Maven integration features are used, you will need to provide valid authentication credentials in the *Maven settings.xml* [http://maven.apache.org/settings.html] file. However, the typical "username" and "password" values are not sufficient, since they are ignored during artifact retrieval (ie, GET calls to the repo). Instead, you must explicitly define the BASIC authentication header value. Unfortunately, this also means you have to manually Base64 encode the value. For example, "Basic admin:artificer1!" becomes "Basic YWRtaW46YXJ0aWZpY2VyMSE=".

It's a pain, but at least as of Maven 3.0.5, it's the best option we could find (PLEASE correct us if we're wrong!).

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-artificer-repo</id>
      <configuration>
        <httpHeaders>
          <property>
            <name>Authorization</name>
            <value>Basic YWRtaW46YXJ0aWZpY2VyMSE=</value>
          </property>
        </httpHeaders>
      </configuration>
    </server>
    <server>
      <id>local-artificer-repo-snapshots</id>
      <configuration>
        <httpHeaders>
          <property>
            <name>Authorization</name>
            <value>Basic YWRtaW46YXJ0aWZpY2VyMSE=</value>
          </property>
        </httpHeaders>
      </configuration>
    </server>
  </servers>
</settings>
```

8.5. Maven Integration in the CLI

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

Another available mechanism for integrating with maven is the Artificer CLI's "maven" command namespace. For help on the maven commands in the CLI, run the Artificer shell (artificer.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 Artificer operations into a script of some kind.

Chapter 9. The Artificer Browser (UI)

9.1. Overview

The Artificer project comes with a user interface that allows end users (or more likely business admins) to browse all of the artifacts in the Artificer repository. This UI is capable of viewing and manipulating all Artificer 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.

Unresolved directive in Guide.asciidoc - include::GuideUserManagement.asciidoc[]

Chapter 10. Artificer Extension Points

Artificer includes multiple "extension point" contracts that allow users to provide additional, custom functionality. The contracts currently include the following:

10.1. ArtifactBuilder

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 Artificer 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 ArtifactBuilder contract allows integrations to define custom properties on the primary artifact, generate derived artifacts, and resolve relationships between the whole set. The Artificer implementation includes ArtifactBuilders 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 ArtifactBuilders for their own artifact types. This is done by performing the following steps:

1. Write a custom ArtifactBuilder Java class - it must implement **ArtifactBuilder**

2. Create an ArtifactBuilderProvider (a class that implements **ArtifactBuilderProvider**) - used to map artifact types to implementations of ArtifactBuilder

3. Provide a text file named **org.artificer.integration.artifactbuilder.ArtifactBuilderProvider** 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 **artificer.extension.customDir**.

The Artificer distribution comes with an example of how to write and package a custom builder - the demo is named **custom-artifact-builder**. Also, see the Javadocs for the relevant classes.

10.2. ArtifactTypeDetector

When an artifact is deployed **without** an explicit model and type, we must attempt to automatically detect them. The ArtifactTypeDetector contract allows integrations to automatically detect an artifact's type, given various contextual-clues. The clues include the path/filename, the artifact's context within an archive, and the artifact's full content. That last bit is the most powerful part, allowing integrations to, for example, parse files and introduce complex business logic.

As with ArtifactBuilder, Artificer provides several built-in detectors. However, it also provides a mechanism that allows users to provide their own ArtifactTypeDetectors . This is done by performing the following steps:

1. Write a custom ArtifactTypeDetector Java class - it must implement **ArtifactTypeDetector**

2. Provide a text file named **org.artificer.integration.artifacttypedetector.ArtifactTypeDetector** 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 #1

3. Package everything up into a JAR and make it available either on the classpath or in an external directory configured by setting property **artificer.extension.customDir**.

The implementations form a prioritized chain, most-specialized at the top and most-generic at the bottom (the contract includes a #getPriority method). It's important to note that execution is **exclusive** and ends processing for the rest of the chain. So, custom implementations **extending** built-in detectors isn't really necessary — simply return null and they'll be executed eventually.

Chapter 11. Artificer Implementation

11.1. Overview

The Artificer 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.

11.2. Server

11.2.1. 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. JPA (Hibernate) + Hibernate Search + RDBMS - used as the persistence engine, where all Artificer data is stored. Artifacts and ontologies are both stored as typical JPA entities. All Artificer queries are mapped to JSQL queries for processing by the JPA API. Performant searching is provided by Hibernate Search, while the backing storage utilizes H2 (by default). Note that all of this is fully configurable and replaceable!

2. JAX-RS (RESTEasy) - used to provide the Artificer Atom based REST API. The S-RAMP specification documents an Atom based REST API that implementations should make available. The Artificer 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).

11.2.2. Security

The Artificer Browser is protected using web application security mechanisms configured in the WARs' web.xml.

By default, the UI uses single-sign-on (SSO) as the actual authentication mechanism. The SSO is provided via integration with the Keycloak framework. The actual web.xml configuration uses a standard basic security-context, but SSO is provided under-the-hood.

The security domain is configured to accept either a username and password (standard BASIC authentication) or a bearer token. 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 the bearer token as a request

header ("Authorization", "Bearer " + bearerToken). For example, the Artificer CLI application uses BASIC authentication when invoking the Artificer Atom API. The Artificer Browser (a web application) requires the user be authenticated into it, and thus is able to use the bearer tokens rather than propagate user credentials.

The app uses a Keycloak realm named **artificer**, which you'll see used in **standalone-full.xml**'s Keycloak subsystem resources. See **Getting Started** for more info.

Chapter 12. Artificer Server Configuration

Out-of-the-box, Artificer 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.

12.1. Hibernate

Out of the box, we provide a fairly standard set of Hibernate configuration defaults. However, for power users, note that **any** Hibernate property may be set, either in `artificer.properties`, environment variables, or System properties. More specifically, literally **any** property prefixed by *hibernate.* will be handed to Hibernate during Artificer startup. More specifics are below:

12.2. Datasource

See the "Getting Started" section for information on Artificer's provided DDL. We currently support Postgres, MySQL, Oracle, SQL Server, DB2/IBM, and H2.

By default, Artificer installs a simple, file-based H2 datasource (see `$JBOSS_HOME/standalone/deployments/artificer-h2-ds.xml`). However, any other WildFly/EAP datasource can be used. Just edit the following in `artificer.properties`:

```
...
hibernate.dialect = org.hibernate.dialect.H2Dialect
hibernate.connection.driver_class = org.h2.Driver
hibernate.connection.datasource = java:jboss/datasources/artificerH2
hibernate.connection.username = sa
hibernate.connection.password = sa
...
```

Note that a datasource is **not** required, although we typically recommend them. Plain JDBC connection URLs, including external instances, are also fully supported (use `hibernate.connection.url`, `hibernate.connection.username`, and `hibernate.connection.password`). If a connection URL is used, Artificer will automatically wrap it with HikariCP, a lightweight and extremely performant connection pool library.

Also note that, due to licensing, we only include the JDBC driver for H2. For all other supported databases, you'll need to ensure that their JDBC driver JAR(s) are available on the classpath, typically through a WildFly/EAP module.

12.3. File Content

Artificer supports storing artifacts' file content on the filesystem or in JDBC Blobs. By default, we use the Blob approach. However, this is configurable at runtime. See the *artificer.file.storage*

property in artificer.properties (values: *blob* or *filesystem*). If you use the filesystem, also include a path with the *artificer.file.storage.filesystem.path* property — all content will be stored there.

Although JDBC Blobs are the default (purely because they're convenient), most databases have fairly restrictive size limits. Even more importantly, there are many vertical and horizontal performance considerations when dealing with larger files. For most use cases, we'd actually recommend using the filesystem.

12.4. Hibernate Configuration

For detailed information of what's possible, please see the Hibernate docs themselves. However, a few things to highlight:

- Query caching and Infinispan second-level entity caching are both enabled by default. This drastically reduces processing time for repeated queries, relationship creation (when uploading large archives of artifacts or highly derived documents), etc. We essentially lean on the default "hibernate" cache available in EAP and Wildfly. However, as mentioned above, feel free to tweak the *hibernate.cache.** settings in `artificer.properties` as needed.

- By default, we use Lucene filesystem indexes with Hibernate Search. This is also highly configurable and can use any number of backends. See the *hibernate.search.** properties in artificer.properties. Note one property in particular: *hibernate.search.default.indexBase*. By default, we use a relative path: *lucene/indexes*. However, that's completely dependent on the location from which you start the application server. Most users will probably need to change this path to something more permanent (and one which you have adequate permissions).

- Also with Hibernate Search, many usages could benefit from hibernate-search-infinispan, which introduces in-memory index caching (clusterable, etc.). This is currently disabled by default, but can be set with the correct properties and including the hibernate-search-infinispan JAR. See *https://docs.jboss.org/author/display/WFLY8/JPA+Reference +Guide#JPAReferenceGuide-UsingtheInfinispansecondlevelcache* for more info.

12.5. Running the Server and Web UI Separately

The Artificer server and web UI are purposefully separated into two WARs, allowing you to run each on separate app servers. If you do so, you'll need to update the "artificer-ui.atom-api.endpoint" property in `artificer-ui.properties`. By default, it assumes co-location and "localhost".

12.6. Remote Connections

If you'd like to allow remote, non-localhost connections to Artificer, you'll need to change two items in `standalone-full.xml`:

- In <interface name="public">, change the inet-address from "127.0.0.1" to "0.0.0.0".

• In the Keycloak subsystem, change the "auth-server-url" from "localhost" to your IP.

12.7. WARNINGS

Chapter 13. Migration Guides

13.1. 0.x # 1.0

- Support for Jetty, Tomcat, Fuse/Fabric/OSGi, and EAP 6.0-6.3 has been dropped. Attention is now solely on WildFly 8 and EAP 6.4. (If any of these are truly a need by community users, we'd love to *hear from you* [https://developer.jboss.org/en/artificer]!)

- We've switched from JCR storage to RDBMS (JPA). Therefore, out of the box, Artificer 1.0 will not work with existing Overlord S-RAMP 0.x repository data. An automated migration script is relatively easy to create. Please *contact us* [https://developer.jboss.org/en/artificer] if this would be helpful!

- All modules renamed from s-ramp-* to artificer-*. Any dependencies on these modules will need updated.

- All Java packages renamed from org.overlord.sramp.* to org.artificer.*

- URL endpoints changed: /s-ramp-server $\rightarrow$ /artificer-server and /s-ramp-ui $\rightarrow$ /artificer-ui

- Names and paths have been changed in many of the configuration settings. If an automated migration utility is needed in this area, please *contact us* [https://developer.jboss.org/en/artificer].

- sramp.properties and sramp-ui.properties have been replaced by artificer.properties and artificer-ui.properties. All configuration keys have been changed to reflect "artificer".

- The default JMS topic has changed to "artificer/events/topic".

- The old "Deriver" and "Expander" contracts have been replaced by "ArtifactBuilder". For more info, see the "Artificer Extension Points" chapter.

- The Artificer CLI no longer uses the default "s-ramp:" command namespace. Core commands can be typed without a ns.

- The Artificer Exception architecture has changed somewhat dramatically. Uses of the Java Client or EJB may require modifications to Exception catching.

- s-ramp-wagon has been completely replaced by a "Maven Facade" servlet in the Artificer Server. This can be used like any other Maven repository. For more info, see the "Artificer Maven Integration" chapter.