Escenic Content Engine

Integration Guide

6.1.3-1

escenic

A CCI COMPANY

Table of Contents

1 Introduction

This manual describes the Escenic Content Engine's open web service, and provides basic advice on how you can use it to integrate the Content Engine with other applications and web services. Client applications can communicate with the web service by means of a REST API. This manual assumes that you are already familiar with the characteristics of REST (Representational State Transfer) APIs, and the principles on which they are based. If you are not, you can find out more about REST here:

- http://en.wikipedia.org/wiki/Representational_State_Transfer

The Content Engine's web service uses the ATOM publishing protocol, and has the following characteristics:

- All operations are performed using HTTP **`GET`**, **`PUT`**, **`POST`** and **`DELETE`** commands
- All resources provided by the web service are identified by URIs.
- Given the initial URI of the web service, all the resources it can provide are then discoverable via hypertext links.
- Resources returned by the web service fall into three categories (as defined by the ATOM publishing protocol):
  - **Feed resources** (that is, ATOM feeds containing a list of resource links)
  - **Entry resources** (single ATOM entry elements containing, for example, content items)
  - **Media resources** (binary resources such as images or video clips)
- Wherever possible, Standard ATOM constructs are used in the entry resources returned by the web service. However, some extensions are used. These fall into two categories:
  - Some RDF (Dublin Core) constructs are used where suitable standard ATOM constructs are not available. See chapter 4 for details.
  - A small number of proprietary extensions are used where necessary.

Here is a complete list of the ATOM standards on which the web service is based:

- RFC 4287 — Atom Syndication Format
- RFC 5023 — Atom Publishing Protocol
- RFC 4685 — Atom Threading Extensions
- RFC 5005 — Feed Paging and Archiving
- RFC 5829 — Version Navigation Link Relations
- RFC 6573 — The Item and Collection Link Relations
- RFC 6903 — Additional Link Relation Types
- OpenSearch — OpenSearch search descriptions are used to describe how to search in an Atom collection.
- draft-snell-atompub-tombstones-06 (moving forward to RFC soon) — Atom Tombstones
- draft-divilly-atom-hierarchy-03 — Atom Hierarchy Relations

Since the web service API is largely based on these open, documented standards, the reference information in this guide is limited to the proprietary extensions needed to provide a comprehensive service.

1.1 Conventions Used in This Manual

The Content Engine's REST API is completely language-independent: it is based entirely on the use of URIs, and all resources returned by the web service either consist of XML documents or are binary objects described in accompanying XML documents. You can therefore write client applications using more or less any programming language you choose (although some languages provide better tools for communicating via the HTTP protocol and handling XML documents than others).

In order not to require knowledge of a particular programming language all of the examples shown in this manual are based on the use of **`curl`**. **`curl`** (see http://curl.haxx.se/) is a command line utility for transferring data using URI syntax, and supports all the HTTP operations needed to communicate with the Content Engine web service. It is a freely-available open source application and is available for all major computing platforms.

1.2 Backwards Compatibility

A primary objective of the Content Engine REST API is stability. Changes are only introduced when they are needed either to correct errors or to add new, required functionality. When changes are necessary, every effort is made to ensure backwards compatibility and minimize disruption to users of the API.

For historical reasons, the REST API makes use of URIs containing the domain name of a former owner of Escenic, **`www.vizrt.com`**. To replace these URIs without compromising backwards compatibility would be both complicated and risky. We consider this to be an unnecessary risk to take for what is essentially a cosmetic issue and will therefore not replace these URIs now or in the future.

1.3 How The Web Service Works

This section contains a brief general description of how a client application is expected to make use of the Content Engine's web service.

Assuming the client has no information about the web service or what it can provide, it can initiate contact by sending an HTTP **`GET`** request to the web service's "start" URI. This is usually

```
http://host-ip-address/webservice/escenic/section/ROOT/subsections
```

where *host-ip-address* is the host name or IP address of the Content Engine. This "start address" (plus a user name and password) is the minimum information a client needs to be able to interact at some level with the web service. The following **`curl`** command, therefore, will initiate contact with a Content Engine web service:

```
curl -u user:password -X GET http://host-ip-address/webservice/escenic/section/ROOT/
subsections
```

The web service responds to such a request by returning a **feed resource**: an ATOM feed document that looks something like this:

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <author>
    <name>Escenic Content Engine</name>
```

```
  </author>
  <id>http://host-ip-address/webservice/escenic/section/ROOT/subsections</id>
  <link rel="self"
        href="http://host-ip-address/webservice/escenic/section/ROOT/subsections"
        type="application/atom+xml"/>
  <updated>2010-06-23T16:51:06.721Z</updated>
  <title type="text">Publication root sections you are authorized to.</title>
  <link rel="changelog"
        href="http://host-ip-address/webservice/escenic/changelog"
        type="application/atom+xml"/>
  <entry>
    <id>http://host-ip-address/webservice/escenic/section/1</id>
    <title type="text">frontpage</title>
    <updated>2010-06-22T10:16:46.309Z</updated>
    <category term="directory" scheme="http://www.vizrt.com/types"/>
    <link rel="edit-media"
          type="application/vnd.escenic.content+xml;type=com.escenic.section"
          href="http://host-ip-address/webservice/escenic/section/1"/>
    <link xmlns:thr="http://purl.org/syndication/thread/1.0"
          rel="down"
          href="http://host-ip-address/webservice/escenic/section/1/subsections"
          type="application/atom+xml"
          thr:count="number-of-subsections"/>
    <link rel="http://www.vizrt.com/types/relation/inboxes"
          href="http://host-ip-address/webservice/escenic/section/1/inboxes"
          type="application/atom+xml"/>
    <link rel="http://www.vizrt.com/types/relation/lists"
          href="http://host-ip-address/webservice/escenic/section/1/lists"
          type="application/atom+xml"/>
    <link rel="http://www.vizrt.com/types/relation/pages"
          href="http://host-ip-address/webservice/escenic/section/1/pages"
          type="application/atom+xml"/>
    <link rel="http://www.vizrt.com/types/relation/content-items"
          href="http://host-ip-address/webservice/escenic/section/1/content-items"
          type="application/atom+xml"/>
    <link rel="http://www.vizrt.com/types/relation/changelog"
          href="http://host-ip-address/webservice/escenic/changelog/section/1"
          type="application/atom+xml"/>
  </entry>
</feed>
```

The document contains a series of **`entry`** elements representing the root sections of all the Content Engine's publications. Each entry contains a sequence of link elements representing the resources to which the requesting user has access. In the example, above, for example, the requesting user has access rights to many resources in the publication "Escenic Times", but not to "frontpage Test".

A **`link`** element has three attributes:

`href`
The URI of a resource.

`type`
This attribute contains a MIME type id identifying the type of data the resource contains. The MIME type **`application/atom+xml`**, for example, indicates that the resource is another ATOM feed document. All the types used by the Content Engine web service are described in chapter 2.

`rel`
This attribute describes the meaning of the resource in terms of its relationship to the current resource. All the link relations used by the Content Engine web service are described in chapter 3.

These three attributes provide the client application with sufficient information to determine:

- Which resource (if any) it needs to access next
- The URL needed to access the resource
- What kind of data it can expect to find in the resource.

The client can construct a new **`GET`** request from the information returned in this document and obtain a new feed resource. It can, for example obtain a feed containing links to all the top-level sections in the Escenic Times publication by submitting the following request:

```
curl -u user:password -X GET http://host-ip-address/webservice/escenic/section/22/
subsections
```

By recursively retrieving feed resources in this way, it is possible to navigate down through the section tree, then from a section to a list of in-boxes, for example, from there to a specific in-box. At various points along this route you can also retrieve feed resources that list links to actual content items (The content items in a section, for example, or the content items in a list or in-box). By following these links you can retrieve entry resources that represent actual content items, and media resources such as image files.

The web service allows entry and media resources to be modified by users with sufficient privileges. This is done by submitting requests using HTTP **`PUT`**, **`POST`** and **`DELETE`** operations:

- The client can update an entry or media resource by submitting a PUT request containing a modified resource and the URI of the resource to be updated.
- The client can create a new entry or media resource by submitting a POST request containing a new resource and the URI of the Atom collection (i.e feed resource) to which it is to be added.
- The client can delete an entry or media resource by submitting a DELETE request containing the URI of the resource to be deleted.

1.4 Change Logs

Most of the resources returned by the web service in response to HTTP **`GET`** requests correspond to standard Content Engine structures that are well-known and described elsewhere: sections, content items, section pages, lists and inboxes. **Change logs**, however, are new structures that requires some explanation.

A change log is a record of all the changes made to a set of content items. The entries in a change log are ordered by change date and are in reverse chronological order (that is, the most recent change is listed first). All change logs are maintained in real time: every time a content item is created, deleted or modified, the change is inserted at the front of all appropriate change logs.

Change logs are maintained for **all** the content items managed by the Content Engine. Special change logs are also maintained for all of a Content Engine's person records (that is, content items

containing information about all registered users of the Content Engine - editors, authors, publication administrators and registered site users).

The Content Engine maintains a number of different change logs, in order to ensure that users can only access change log entries for content items to which they have read access. Specifically, the Content Engine maintains one content item change log for each of its **protection domains**. A protection domain is a set of sections governed by an **explicitly defined** set of access permissions. What this means in practice is that there is always at least one protection domain for each of a Content Engine's publications, since there will always be an explicit set of access permissions associated with each publication's root section. A publication will have additional protection domains for any subsection that does not simply inherit access controls from its parent section, but has explicitly defined access controls of its own.

In addition to a change log for each protection domain, the Content Engine maintains:

- A person record change log for each publication
- An aggregated change log for all changes made to content items belonging to the publication. (A content item belongs to the publication that contains its home section.)

Access to the change logs is governed as follows:

- Access to the person record change log is granted to any user with **`journalist`** access to any part of any publication.
- Access to a protection domain change log is granted to any user with read access to the relevant protection domain.
- Access to an aggregated publication change log is granted to any user with global read access to the relevant publication.

A change log is returned as an Atom **paged collection**. The first entry in the first Atom feed returned represents the most recently changed content item in the protection domain represented by the change log. The next entry represents the second most recent change, and so on. You can get the next page, a feed containing an older set of changes by following this feed's **`next`** link.

Change logs can be accessed in two ways:

- Via a link in the web service's "start" feed resource
- Via a link in a section's "content items" feed resource

> For advice on how to use the Content Engine's change logs in the most efficient way, and how to make use of server-sent events (SSE), see section 6.1.

1.4.1 Accessing Change Logs From the "Start" Feed

The highlighted link in the following "start feed":

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <author>
    <name>Escenic Content Engine</name>
  </author>
  <id>http://host-ip-address/webservice/escenic/section/ROOT/subsections</id>
  <link rel="self"
        href="http://host-ip-address/webservice/escenic/section/ROOT/subsections"
        type="application/atom+xml"/>
```

```
  <updated>2010-05-31T12:45:38.144Z</updated>
  <title type="text">Publication root sections you are authorized to.</title>
  <link rel="http://www.vizrt.com/types/relation/protection-domains"
        href="http://host-ip-address/webservice/escenic/changelog"
        type="application/atom+xml"/>
...
</feed>
```

returns a feed resource containing a list of all the change logs the requesting user is allowed to access:

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <id>http://host-ip-address/webservice/escenic/changelog</id>
  <title type="text">Change logs accessible to demo_admin</title>
  <updated>2010-06-23T17:54:09.173Z</updated>
  <author>
    <name>Escenic Content Engine</name>
  </author>
  <link rel="self"
        href="http://host-ip-address/webservice/escenic/changelog"
        type="application/atom+xml"/>
  <entry>
    <id>http://host-ip-address/webservice/escenic/publication/demo</id>
    <title type="text">Change log for persons in publication 'demo'</title>
    <link rel="http://www.vizrt.com/types/relation/changelog"
          href="http://host-ip-address/webservice/escenic/changelog/person/1"
          type="application/atom+xml"/>
  </entry>
  <entry>
    <id>http://localhost:8080/webservice/escenic/publication/demo</id>
    <title type="text">Change log for publication 'demo'</title>
    <link rel="http://www.vizrt.com/types/relation/changelog"
          href="http://host-ip-address/webservice/escenic/changelog/publication/1"
          type="application/atom+xml"/>
  </entry>
  <entry>
    <id>http://localhost:8080/webservice/escenic/publication/demo/structural</id>
    <title type="text">Change log of structural changes for publication 'demo'</title>
    <link rel="http://www.vizrt.com/types/relation/changelog"
          href="http://host-ip-address/webservice/escenic/changelog/publication/1/
structural"
          type="application/atom+xml"/>
  </entry>
  <entry>
    <id>http://host-ip-address/webservice/escenic/section/4</id>
    <title type="text">Change log for content in protection domain 'New Articles'</
title>
    <link rel="http://www.vizrt.com/types/relation/changelog"
          href="http://host-ip-address/webservice/escenic/changelog/section/4"
          type="application/atom+xml"/>
  </entry>
...
</feed>
```

There are two different types of change log: "normal" change logs for reporting changes in content, and structural change logs for reporting changes in structure. A structural change log has the suffix **`/structural`** on the end of its URL. So in the example shown above, **`http://`***host-ip-address***`/webservice/escenic/changelog/publication/1`** is the content change log for

the demo publication, and **`http://`***host-ip-address*`/webservice/escenic/changelog/publication/1/structural`** is the structural change log for the same publication.

Real changes to content items and sections are reported in the "normal" content change log. The structural change log is used to report changes that are not directly applied to a content item but are simply consequences of structural changes elsewhere in the publication. If, for example, a content item's home section is moved, then that change will be reported for the content item in the structural change log.

1.4.2 Accessing a Section's Change Log

You can access the change log that contains all the changes in a specific section from the section's feed resource:

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <author>
    <name>Escenic Content Engine</name>
  </author>
  <id>http://host-ip-address/webservice/escenic/section/1/subsections</id>
  <link rel="self"
        href="http://host-ip-address/webservice/escenic/section/1/subsections"
        type="application/atom+xml"/>
  <updated>2010-06-23T17:35:45.180Z</updated>
  <title type="text">Subsections for Section with id=1</title>
  <entry>
    <id>http://host-ip-address/webservice/escenic/section/4</id>
    <title type="text">New Articles</title>
    <updated>2010-06-22T10:16:46.637Z</updated>
    <category term="directory" scheme="http://www.vizrt.com/types"/>
...
    <link rel="http://www.vizrt.com/types/relation/changelog"
          href="http://host-ip-address/webservice/escenic/changelog/section/4"
          type="application/atom+xml"/>
  </entry>
</feed>
```

Note that the change log accessed in this way is guaranteed to contain all the changes made to the content items in the section, but it may also contain changes made in other sections, since it is the change log for the protection domain to which the section belongs.

1.4.3 Accessing Tag Structure Change Logs

You can access a change log that contains all the changes made to the tags in a tag structure from the tag structure's feed resource:

```
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:opensearch="http://a9.com/-/spec/
opensearch/1.1/"
       xmlns:thr="http://purl.org/syndication/thread/1.0">
  <id>http://host-ip-address/webservice/escenic/classification/
tag:book@escenic.com,2011</id>
  <title type="text">Book</title>
  <link href="http://host-ip-address/webservice/escenic/classification/
tag:book@escenic.com,2011" rel="self"/>
  <author>
    <name>Escenic Classification Web Service</name>
  </author>
  <updated>2012-06-20T18:00:00.000Z</updated>
```

```
  <link href="http://host-ip-address/webservice/escenic/changelog/classification/
tag:book@escenic.com,2011"
          rel="http://host-ip-address/types/relation/changelog"/>
  <opensearch:Url
     template="http://host-ip-address/webservice/escenic/classification/tag/search?
searchTerms={searchTerms}&parent=tag%3Abook%40escenic.com%2C2011"
     type="application/atom+xml" rel="parent"/>
...
</feed>
```

1.4.4 Accessing Inbox/List Change Logs

You can access a change log that contains all the changes made to an inbox or any content items in it from the inbox's feed. Similarly, you can access a change log that contains all the changes made to a list or any content items in it from the list's feed. Here is an inbox feed, with the change log link highlighted.

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <author>
    <name>Escenic Content Engine</name>
  </author>
  <id>http://host-ip-address/webservice/escenic/section/1/inboxes</id>
  <link rel="self" href="http://host-ip-address/webservice/escenic/section/1/inboxes"
 type="application/atom+xml"/>
  <updated>2012-10-11T12:51:09.969Z</updated>
  <entry>
    <id>http://host-ip-address/webservice/content/com.escenic.inbox/2</id>
    <title type="text"/>
    <link href="http://host-ip-address/webservice/content/com.escenic.list-pool/2"
 rel="http://www.vizrt.com/types/relation/list-pool" type="application/xhtml+xml"/>
    <link rel="http://www.vizrt.com/types/relation/changelog" href="http://host-ip-
address/webservice/escenic/changelog/inbox/2" type="application/atom+xml"/>
    <link rel="edit-media" type="application/vnd.escenic.content+xml;
 type=com.escenic.inbox" href="http://host-ip-address/webservice/content/
com.escenic.inbox/2" title=""/>
  </entry>
</feed>
```

1.5 Searching

The Content Engine web service includes a search service based on **OpenSearch**. OpenSearch is a standard for:

- Describing how to search in Atom collections
- Annotating search results returned in Atom feeds

For details see http://www.opensearch.org/Home. Using OpenSearch helps to ensure that the search service complies with REST principles, and can be searched by clients with no prior knowledge of the Content Engine or its internal structures.

Every **http://www.vizrt.com/types/relation/subsection** entry feed returned by the web service contains an **http://www.vizrt.com/types/relation/content-items** link:

```
...
```

```
    <link rel="http://www.vizrt.com/types/relation/content-items"
          href="http://host-ip-address/webservice/escenic/section/22/content-items"
          type="application/atom+xml"/>
...
</feed>
```

This link returns an empty feed resource representing the collection of all content items in the section. It also, however, contains a **`search`** link:

```
<link rel="search"
     href="http://host-ip-address/webservice/open-search/escenic/22/content-search-
description.xml"
     type="application/opensearchdescription+xml"/>
```

This link returns an OpenSearch document containing URI templates that can be used to search for content items in the section:

```
<OpenSearchDescription xmlns="http://a9.com/-/spec/opensearch/1.1/">
  <ShortDescription>Escenic search</ShortDescription>
  <Description/>
  <Url type="application/atom+xml"
       template="http://host-ip-address/webservice/publication/publication-name/
search/escenic/22/
{searchTerms}/?pw={startPage?}&amp;c={count?}&amp;tag={tagIdentifier?}"/>
  <Url type="text/html"
       template="http://host-ip-address/webservice/publication/publication-name/
search/escenic/22/
{searchTerms}/?
pw={startPage?}&amp;c={count?}&amp;tag={tagIdentifier?}&amp;format=html"/>
  <Contact>http://www.escenic.com/</Contact>
  <Tags/>
  <LongName>Escenic Content Engine Search</LongName>
  <Image height="16" width="16" type="image/x-icon">http://host-ip-address/webservice/
images/ece.ico</Image>
  <Query role="example" searchTerms="cat"/>
  <Developer>Escenic AS</Developer>
  <Attribution/>
  <SyndicationRight>private</SyndicationRight>
  <AdultContent>false</AdultContent>
  <OutputEncoding>UTF-8</OutputEncoding>
  <InputEncoding>UTF-8</InputEncoding>
</OpenSearchDescription>
```

One of the templates returns results in a paged Atom feed, the other returns the results as HTML. The client can use one of them to construct and submit a search request:

```
curl -u user:password -X GET http://host-ip-address/webservice/publication/pub-name/
search/escenic/22/Obama/?pw=1&c=10
```

If the Atom feed template (**`type="application/atom+xml"`**) is used, then a paged Atom feed is returned in which the actual results are wrapped in an **`OpenSearch`** element and presented in elements belonging to a proprietary, **undocumented and deprecated** Escenic format identified by the namespace **`http://www.escenic.com/2007/content-engine`**. The use of this format is temporary and you should not invest any effort in attempting to make use of it: it will be replaced with a properly documented and supported format in a future version of the Content Engine.

2 Supported MIME Types

Every **`link`** element in the feed resources returned by the Content Engine web service has a **`type`** attribute. This attribute contains a MIME type identifier that identifies the type of data in the resource referenced by the link. This chapter lists all the **`type`** URIs that client applications can expect to find in feed resources returned by the Content Engine web service, and descriptions of their meaning.

The MIME type URIs found in Content Engine feed resources fall into two categories:

- Standard MIME type identifiers defined in the IANA MIME type registry (http://www.iana.org/assignments/media-types/). The descriptions of these relations consist of a reference to the appropriate specification and a brief description of how it is used by the Content Engine web service.
- Proprietary MIME types defined by Escenic.

> Note that the MIME type in a link is only intended to provide client applications with a hint about what kind of data to expect. It is not a contract and cannot be relied upon to always be correct. The resource referenced by the link may, for example, no longer exist. The MIME type's primary purpose is to provide information that the client application can use when selecting the link to follow. Client applications should therefore always use the HTTP header of the resource returned by the link to determine what kind of resource it actually is and how it should be processed.

2.1 application/atom+xml

This standard MIME type indicates that the referenced resource is an Atom feed, collection or entry.

The Content Engine web service uses this MIME type for:

- Subsection feeds
- Content item feeds
- Search result feeds
- Change log feeds
- Section list feeds
- Section in-box feeds
- Section page feeds
- Content item entries

2.2 application/vnd.vizrt.payload+xml

This proprietary MIME type indicates that the referenced resource is a Viz Data Format (VDF) payload document.

2.3 application/vnd.vizrt.model+xml

This proprietary MIME type indicates that the referenced resource is a Viz Data Format (VDF) model document.

2.4 application/vnd.escenic.content+xml

This proprietary MIME type indicates that the referenced resource has content in an undocumented, deprecated format identified by the namespace **`http://www.escenic.com/2007/content-engine`**.

The Content Engine uses this MIME type for various kinds of user content. The MIME type id will usually be followed by one of the following type specifications in order to indicate precisely what kind of user content is to be expected:

com.escenic.section
: The VDF document contains an Escenic section definition.

com.escenic.inbox
: The VDF document contains an Escenic inbox definition.

com.escenic.list
: The VDF document contains an Escenic inbox definition.

com.escenic.page
: The VDF document contains an Escenic inbox definition.

The use of this format is temporary and you should not invest any effort in attempting to make use of it: it will be replaced with a properly documented and supported format in a future version of the Content Engine.

2.5 application/opensearchdescription+xml

This standard MIME type indicates that the referenced resource is an OpenSearch search description document.

The Content Engine uses this MIME type to describe the URL format that must be used to search for content in a section. For more information about searching, see section 1.5; for an example of how to search, see section 5.3.

3 Supported Link Relations

Every **`link`** element in the feed resources returned by the Content Engine web service has a **`rel`** attribute. This attribute contains a **link relation**, a name that identifies the relationship between the current resource and the resource referenced by the link. This section lists all the link relations that client applications can expect to find in feed resources returned by the Content Engine web service, and descriptions of their meaning.

The link relations found in Content Engine feed resources fall into two categories:

- Standard link relations defined in one of the ATOM specifications. The descriptions of these relations consist of a reference to the appropriate specification and a brief account of how they are used by the Content Engine web service.
- Proprietary relations defined by Escenic. The descriptions of these relations consist of a brief account of their meaning and use.

3.1 Standard Link Relations

3.1.1 about

This link relation is defined in http://tools.ietf.org/html/rfc6903.

It is used by the Content Engine web service to represent the relationship between a list or in-box item and the content item it references. You can retrieve the content item referenced by a list/in-box item by following the entry's **`about`** link.

The following example shows a list item's list pool handle, which contains an **`about`** link to the content item it represents.

```
<entry xmlns="http://www.w3.org/2005/Atom">
  <id>http://host-ip-address/webservice/escenic/list-pool/handle/583347</id>
  <title type="text">(type=1 ,id=204697 ,pubId=-1)</title>
  ...
  <link
    rel="about"
    href="http://host-ip-address/webservice/escenic/content/204697"
    type="application/atom+xml"/>
  ...
</entry>
```

3.1.2 collection

This link relation is defined in http://www.ietf.org/rfc/rfc6573.txt.

It is used by the Content Engine web service to represent the relationship between a list or in-box item and its owning list or in-box.

The following example shows a list item's list pool handle, which contains a **`collection`** link to the parent list pool.

```
<entry xmlns="http://www.w3.org/2005/Atom">
  <id>http://host-ip-address/webservice/escenic/list-pool/handle/583347</id>
  <title type="text">(type=1 ,id=204697 ,pubId=-1)</title>
  ...
  <link
    rel="collection"
    href="http://host-ip-address/webservice/escenic/list-pool/423"
    type="text/uri-list"/>
</entry>
```

3.1.3 down

This link relation is defined in http://tools.ietf.org/html/draft-divilly-atom-hierarchy-03.

It is used by the Content Engine web service to represent hierarchical relationships. A section's subsection's, for example, are represented by **down** links. You can navigate down through a publication's section hierarchy by simply following **down** links. You can also use **down** links to navigate down a tag hierarchy.

The following example shows a link of this type:

```
<entry>
  ...
  <link xmlns:thr="http://purl.org/syndication/thread/1.0"
        rel="down"
        href="http://host-ip-address/webservice/escenic/section/3461/subsections"
        type="application/atom+xml"
        thr:count="number-of-entries"/>
  ...
</entry>
```

The **count** attribute is defined in RFC 4685. If present, it specifies the number of entries that will be returned by following the link.

3.1.4 edit

This link relation is defined in RFC 5023.

It is used by the Content Engine web service to represent an entry containing an actual content item that can be edited by the client application and re-submitted.

The following example shows a link of this type in a change log entry:

```
<entry xmlns:app="http://www.w3.org/2007/app"
       xmlns:metadata="http://xmlns.escenic.com/2010/atom-metadata"
       xmlns:dcterms="http://purl.org/dc/terms/">
...
  <link href="http://host-ip-address/webservice/escenic/content/4" rel="edit"/>
...
</entry>
```

3.1.5 next

This link relation is defined in IETF RFC-5005.

Some of the ATOM collections returned by the web service (for example, search results and change log resources) may be **paged** - divided into a series of linked feed resources. A **`next`** link points to the next page (the next feed resource in the series).

The following example shows a change log resource containing a link of this type:

```
<feed xmlns="http://www.w3.org/2005/Atom">
...
  <link rel="self"
        href="http://host-ip-address/webservice/escenic/changelog/publication/12/
before/9425007?count=10"/>
  <link rel="previous"
        href="http://host-ip-address/webservice/escenic/changelog/publication/12/
after/9425006?count=10"/>
  <link rel="next"
        href="http://host-ip-address/webservice/escenic/changelog/publication/12/
before/9424970?count=10"/>
...
</feed>
```

Note that many Atom feeds are ordered by date, with the newest entry first. This means that the **`next`** page of a change log will contain links to **older** resources, not **newer** ones.

3.1.6 previous

This link relation is defined in IETF RFC-5005.

Some of the ATOM collections returned by the web service (for example, search results and change log resources) may be **paged** - divided into a series of linked feed resources. A **`previous`** link points to the previous page (the previous feed resource in the series).

The following example shows a change log resource containing a link of this type:

```
<feed xmlns="http://www.w3.org/2005/Atom">
...
  <link rel="self"
        href="http://host-ip-address/webservice/escenic/changelog/publication/12/
before/9425007?count=10"/>
  <link rel="previous"
        href="http://host-ip-address/webservice/escenic/changelog/publication/12/
after/9425006?count=10"/>
  <link rel="next"
        href="http://host-ip-address/webservice/escenic/changelog/publication/12/
before/9424970?count=10"/>
...
</feed>
```

Note that many Atom feeds are ordered by date, with the newest entry first. This means that the **`previous`** page of a change log will contain links to **newer** resources, not **older** ones.

3.1.7 working-copy

This link relation is defined in IETF RFC-5829.

Some content items returned by the web service may have an unpublished **staged** copy. A **`working-copy`** link points from a content item to such a working copy.

The following example shows a content item entry resource containing a link of this type:

```
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:app="http://www.w3.org/2007/app"
 xmlns:metadata="http://xmlns.escenic.com/2010/atom-metadata" xmlns:dcterms="http://
purl.org/dc/terms/">
...
  <link
        href="http://href="host-ip-address/webservice/escenic/content/6" rel="working-
copy"/>
...
</entry>
```

3.1.8 working-copy-of

This link relation is defined in IETF RFC-5829.

This is the opposite relation to **working-copy**. A **working-copy-of** link points from a working copy to the published content item on which it is based.

The following example shows a content item entry resource containing a link of this type:

```
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:app="http://www.w3.org/2007/app"
 xmlns:metadata="http://xmlns.escenic.com/2010/atom-metadata" xmlns:dcterms="http://
purl.org/dc/terms/">
...
  <link
        href="http://href="host-ip-address/webservice/escenic/content/6/published"
 rel="working-copy-of"/>
...
</entry>
```

3.1.9 search

This link relation is defined in http://www.opensearch.org/Specifications/OpenSearch/1.1.

It identifies a link to an **OpenSearch** specification. When a **search** link appears in a feed resource returned by the web service, it indicates that the collection is searchable. The resource referenced by the **search** link is an OpenSearch document describing the format of the search requests that may be submitted to search the collection. Links of this type occur in feed resources representing content items collections.

The following example shows a feed resource containing a link of this type:

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <author>
    <name>Escenic Content Engine</name>
  </author>
  <id>http://host-ip-address/webservice/escenic/section/22/content-items</id>
  <link rel="self"
        href="http://host-ip-address/webservice/escenic/section/22/content-items"
        type="application/atom+xml"/>
  <updated>2010-06-01T07:08:10.427Z</updated>
  <author/>
  <title type="text">Content items for section with dbid="22"</title>
  <link rel="search"
        href="http://host-ip-address/webservice/open-search/escenic/22/content-search-
description.xml"
```

```
        type="application/opensearchdescription+xml"/>
</feed>
```

3.2 Proprietary Link Relations

3.2.1 http://www.escenic.com/types/relation/model

This link relation identifies a link to the content model for the content item.

The following example shows a link of this type:

```
<entry>
  ...
    <link
        href="http://host-ip-address/webservice/escenic/publication/demo/model/
content-type/news"
        rel="http://www.escenic.com/types/relation/model"
        type="application/vnd.vizrt.model+xml" title="news"/>
  ...
</entry>
```

3.2.2 http://www.escenic.com/types/relation/summary-model

This link relation identifies a link to the content summary model for the content item. Links of this type occur in entry resources representing content items.

The following example shows a link of this type:

```
<entry>
  ...
    <link
        href="http://host-ip-address/webservice/escenic/publication/demo/model/
content-summary/news"
        rel="http://www.escenic.com/types/relation/summary-model"
        type="application/vnd.vizrt.model+xml" title="news"/>
  ...
</entry>
```

3.2.3 http://www.vizrt.com/types/relation/changelog

This is a proprietary link relation. It identifies a link to an Escenic **change log** (see section 1.4). A change log is a paged collection (that is, a chained series of feed resources) containing links to all the changes made in a particular protection domain, ordered by modification date. For a detailed description of what a change log is, see section 1.4. Links of this type can be found in

- The protection domain feed resource returned from a **`http://www.vizrt.com/types/relation/protection-domains`** link (see section 3.2.17).
- The sub-section feed resources returned from **`down`** links (see section 3.1.3).
- Change log feeds. In this case, the type is used to identify a link to a persistent server-sent event connection (see section 6.1.2.

The following example shows a protection domain feed resource containing links of this type:

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <id>http://host-ip-address/webservice/escenic/changelog</id>
  <title type="text">Protection domains accessible to demo_admin</title>
  <updated>2010-06-23T17:54:09.173Z</updated>
  <author>
    <name>Escenic Content Engine</name>
  </author>
  <link rel="self"
        href="http://host-ip-address/webservice/escenic/changelog"
        type="application/atom+xml"/>
  <entry>
    <id>http://host-ip-address/webservice/escenic/publication/demo</id>
    <title type="text">demo</title>
    <link rel="http://www.vizrt.com/types/relation/changelog"
          href="http://host-ip-address/webservice/escenic/changelog/person/1"
          type="application/atom+xml"/>
  </entry>
  <entry>
    <id>http://host-ip-address/webservice/escenic/publication/demo</id>
    <title type="text">demo</title>
    <link rel="http://www.vizrt.com/types/relation/changelog"
          href="http://host-ip-address/webservice/escenic/changelog/publication/1"
          type="application/atom+xml"/>
  </entry>
  <entry>
    <id>http://host-ip-address/webservice/escenic/section/3</id>
    <title type="text">topicRoot</title>
    <link rel="http://www.vizrt.com/types/relation/changelog"
          href="http://host-ip-address/webservice/escenic/changelog/section/3"
          type="application/atom+xml"/>
  </entry>
  <entry>
    <id>http://host-ip-address/webservice/escenic/section/4</id>
    <title type="text">New Articles</title>
    <link rel="http://www.vizrt.com/types/relation/changelog"
          href="http://host-ip-address/webservice/escenic/changelog/section/4"
          type="application/atom+xml"/>
  </entry>
...
</feed>
```

A change log can contain either:

- Changes to ordinary content items, in which case the URI in the **`href`** attribute will end with **`section/`***id*, or
- Changes to person records (users, authors, registered site visitors and so on), in which case the URI in the **`href`** attribute will end with **`person/`***id*
- All changes in one publication, in which case the URI in the **`href`** attribute will end with **`publication/`***id*. Access to this change log is restricted to users with global **`editor`**, **`journalist`**, or **`reader`** access rights.

3.2.4 http://www.vizrt.com/types/relation/content-items

This link relation identifies a link to a collection of Escenic content items. Links of this type occur in feed resources representing publication sections.

Currently, the feed resource returned by a link of this type is always empty (that is, the feed contains no **`entry`** elements). In future, it is the intention that this will not be the case, such links will return paged collections. In the mean time, however, a feed resource of this type is still useful:

- It provides a target for HTTP **`POST`** operations, allowing content items to be added to the section it represents
- It contain a **`search`** link, enabling the section to be searched (see section 3.1.9).

The following example shows a link of this type:

```
<entry>
  ...
  <link rel="http://www.vizrt.com/types/relation/content-items"
        href="http://host-ip-address/webservice/escenic/section/22/content-items"
        type="application/atom+xml"/>
  ...
</entry>
```

3.2.5 http://www.vizrt.com/types/relation/home-section

This link relation identifies a link to a content item's home section. Links of this type occur in entry resources representing content items.

The following example shows a link of this type:

```
<entry>
  ...
    <link rel="http://www.vizrt.com/type/relation/home-section"
          href="http://host-ip-address/webservice/escenic/section/4"
          title="New Articles"
          type="application/atom+xml; type=entry"/>
  ...
</entry>
```

3.2.6 http://www.vizrt.com/types/relation/inboxes

This link relation identifies a link to a collection of Escenic in-boxes. Links of this type occur in feed resources representing publication sections.

The following example shows a link of this type:

```
<entry>
  ...
  <link rel="http://www.vizrt.com/types/relation/inboxes"
        href="http://host-ip-address/webservice/escenic/section/22/inboxes"
        type="application/atom+xml"/>
  ...
</entry>
```

3.2.7 http://www.vizrt.com/types/relation/lists

This link relation identifies a link to a collection of Escenic lists. Links of this type occur in feed resources representing publication sections.

The following example shows a link of this type:

```
<entry>
  ...
  <link rel="http://www.vizrt.com/types/relation/lists"
        href="http://host-ip-address/webservice/escenic/section/22/lists"
        type="application/atom+xml"/>
  ...
</entry>
```

3.2.8 http://www.vizrt.com/types/relation/list-pool

This link relation identifies a link to a **list pool** - a feed containing links to all the entries in a list or inbox. Links of this type occur in feed resources representing a section's lists or inboxes.

The following example shows a link of this type:

```
<entry>

...
  <link href="http://host-ip-address/webservice/content/com.escenic.list-pool/423"
        rel="http://www.vizrt.com/types/relation/list-pool"
        type="application/atom+xml"/>
...
</entry>
```

3.2.9 http://www.vizrt.com/types/relation/lock

This link relation identifies a link that can be used to lock a content item or a specific fragment of the content item. Links of this type occur in entry resources representing content items.

The following example shows a link of this type:

```
<entry>
  ...
  <link rel="http://www.vizrt.com/types/relation/lock"
        href="http://host-ip-address/webservice/escenic/lock/article/4"/>
  ...
</entry>
```

3.2.10 http://www.vizrt.com/types/relation/log

This link relation identifies a link to a content item edit history log. Links of this type occur in entry resources representing content items.

The following example shows a link of this type:

```
<entry>
  ...
  <link href="http://host-ip-address/webservice/escenic/content/1337/log"
        rel="http://www.vizrt.com/types/relation/log"/>
  ...
</entry>
```

3.2.11 http://www.vizrt.com/types/relation/merge

This link relation identifies a link that can be used to merge two or more tags. Links of this type occur in resources representing tags.

The following example shows a link of this type:

```
<entry>
  ...
  <link rel="http://www.vizrt.com/types/relation/merge"
        href="http://host-ip-address/webservice/escenic/classification/merge>
  ...
</entry>
```

3.2.12 http://www.vizrt.com/types/relation/pages

This link relation identifies a link to a collection of Escenic section pages. Links of this type occur in feed resources representing publication sections.

> Currently, links with this relation will not work: they will return an **`HTTP 404`** error.

The following example shows a link of this type:

```
<entry>
  ...
  <link rel="http://www.vizrt.com/types/relation/pages"
        href="http://host-ip-address/webservice/escenic/section/22/pages"
        type="application/atom+xml"/>
  ...
</entry>
```

3.2.13 http://www.vizrt.com/types/relation/parent

This link relation identifies a hierarchical relationship to a parent. You can use links of this type to navigate upwards through a tag hierarchy, for example.

The following example shows a link of this type:

```
<entry>
  ...
    <link
      rel="http://www.vizrt.com/types/relation/parent"
      href="http://host-ip-address/webservice/escenic/classification/
tag:folksonomy.escenic.com,2002/tag:folksonomy.escenic.com,2002:db:1346"
      title="politics"/>
  ...
</entry>
```

3.2.14 http://www.vizrt.com/types/relation/person

This link relation identifies a link to a collection of Escenic persons. Links of this type occur in feed resources representing publications. One link of this type is included in the web service's "start" feed resource (**`http://`***host-ip-address*`/webservice/escenic/section/ROOT/subsections`**).

> Currently, the feed resource returned by a link of this type is always empty (that is, the feed contains no **`entry`** elements). In future, it is the intention that this will not be the case, such links will return paged collections. In the mean time, however, a feed resource of this type is still useful:
>
> - It provides a target for HTTP **`POST`** operations, allowing persons to be added to the publication it represents

- It contains an **`http://www.vizrt.com/types/relation/person-lookup`** link that you can use to search for persons

The following example shows a link of this type:

```
<feed>
  ...
  <title type="text">Publication root sections you are authorized to.</title>
  ...
  <link rel="http://www.vizrt.com/types/relation/person"
        href="http://host-ip-address/webservice/escenic/person"
        type="application/atom+xml"/>
  ...
</feed>
```

3.2.15 http://www.vizrt.com/types/relation/person-lookup

An **`http://www.vizrt.com/types/relation/person-lookup`** link is contained in the Atom feed returned by an **`http://www.vizrt.com/types/relation/person`** link (see section 3.2.14). A link of this type references an OpenSearch document describing the format of the search requests that may be submitted to search for persons.

```
</feed>
...
    <link rel="http://www.vizrt.com/types/relation/person-lookup"
          href="http://host-ip-address/webservice/open-search/person-lookup-
description.xml"
          type="application/opensearchdescription+xml"/>
...
</feed>
```

3.2.16 http://www.vizrt.com/types/relation/preview

This relation identifies a link to a preview of a content item. Links of this type occur in entry resources representing content items.

The following example shows a link of this type:

```
<entry>
  ...
  < link href="http://host-ip-address/webservice/escenic/content/preview"
        rel="http://www.vizrt.com/types/relation/preview"/>
  ...
</entry>
```

3.2.17 http://www.vizrt.com/types/relation/protection-domains

This link relation identifies a link to a feed resource listing all the **protection domains** the requesting user has access to. One link of this type is included in the web service's "start" feed resource (**`http://`***host-ip-address*`/webservice/escenic/section/ROOT/subsections`**).

The following example shows the "start" feed resource containing a link of this type:

```
<feed xmlns="http://www.w3.org/2005/Atom">
...
  <title type="text">Publication root sections you are authorized to.</title>
```

```
  <link rel="http://www.vizrt.com/types/relation/protection-domains"
        href="http://host-ip-address/webservice/escenic/changelog"
        type="application/atom+xml"/>
...
</feed>
```

3.2.18 http://www.vizrt.com/types/relation/publication

This link relation identifies a link to the publication to which a content item belongs. Links of this type occur in entry resources representing content items.

The following example shows a link of this type:

```
<entry>
  ...
    <link rel="http://www.vizrt.com/type/relation/publication"
          href="http://host-ip-address/webservice/escenic/publication/demo"
          title="demo"
          type="application/atom+xml; type=entry"/>
  ...
</entry>
```

3.2.19 http://www.vizrt.com/types/relation/section

This link relation identifies a link to one of the sections in which a content item appears. Links of this type occur in entry resources representing content items.

The following example shows a link of this type:

```
<entry>
  ...
    <link rel="http://www.vizrt.com/type/relation/section"
          href="http://host-ip-address/webservice/escenic/section/4"
          title="New Articles"
          type="application/atom+xml; type=entry"/>
  ...
</entry>
```

3.2.20 http://www.vizrt.com/types/relation/top

This link relation identifies a link to the top (or root) node of a hierarchy. You can use links of this type to navigate from a tag to its owning tag structure, for example.

The following example shows a link of this type:

```
<entry>
  ...
  <link
      href="http://host-ip-address/webservice/escenic/classification/
tag:folksonomy.escenic.com,2002"
      rel="http://www.vizrt.com/types/relation/top"
      title="Tags"/>
  ...
</entry>
```

3.2.21 http://www.vizrt.com/types/relation/more-info

In Content Studio if it sees a content having links with this relation, for each link it will use the URL and open it in the side panel.

```
> curl -I -X HEAD -u user:password http://server:port/webservice/escenic/content/35
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Cache-Control: no-transform, max-age=60
ETag: "13b2d5fa488"
Allow: OPTIONS, TRACE, GET, HEAD, POST, PUT, DELETE
Link: <http://server:port/webservice/publication/pub1/usage/article/35>; rel="usage";
 title="Incoming Links"
Link: <http://user:password@server:port/webservice/escenic/content/35>; rel="http://
www.vizrt.com/types/relation/more-info "; type="application/atom+xml;type=entry";
 title="Webservice view"
Link: <http://google.com>; rel="http://www.vizrt.com/types/relation/more-info ";
 type="application/atom+xml;type=entry"; title="Google example"
Content-Type: application/atom+xml; type=entry
Content-Length: 0
Date: Mon, 26 Dec 2111 12:20:16 GMT
```

From above example we can see there are two link relation with http://www.vizrt.com/types/relation/more-info for the content. When Content Studio opens this article it will open following URL in the side panel of the content editor.

```
http://user:password@server:port/webservice/escenic/content/35
https://www.google.com/search?q=35&qscrl=1
```

4 Standard Element Usage

When the Content Engine's web service returns an Atom feed containing a content item, it maps various content item properties and fields to standard Atom elements. In addition, some elements from other well-known standards are used where suitable standard Atom elements are not available.

This section contains descriptions of the mappings and RDF extensions used by the web service.

When the web service returns content items in an Atom feed (for example, in a change log feed), the following Atom elements are used to hold specific content item properties or field values:

`title`
: This Atom element holds the content item's title field (that is, the field that is nominated as the title field in the **`content-type`** resource - see title-field). Since the content item title is included in the Atom entry in this way, the title field is omitted from the entry's VDF payload.

`updated`
: This Atom element holds the content item's **`Published`** property.

`published`
: This Atom element holds the content item's **`First Published`** property.

`category`
: This Atom element holds the content item's **`tags`** property. A content item can contain an unlimited number of tags.

The following elements from the Atom Publishing Protocol (namespace URI **`http://www.w3.org/2007/app`**) and RDF Dublin Core (namespace URI **`http://purl.org/dc/terms/`**) are also used:

`app:edited`
: This Atom Publishing Protocol element holds the content item's **`Last Modified`** property.

`dcterms:created`
: This RDF Dublin Core element holds the content item's **`Created`** property.

`dcterms:identifier`
: This RDF Dublin Core element holds the content item's **`Id`** property.

`app:control`
: This Atom Publishing Protocol element holds the state of the content item.

`app:draft`
: This Atom Publishing Protocol element define whether current content item is draft or not.

Metadata items for which no suitable standard elements could be found are included in the entries in a proprietary namespace identified by the URI **`http://xmlns.escenic.com/2010/atom-metadata`**. These include:

`metadata:creator`
: The content item's **`Creator`** property. The content model of this element is an Atom person construct.

`metadata:reference`
: The content item's **`Reference`** property.

metadata:publication
The content item's **Publication** property.

metadata:deleted
This identifies whether a content item is deleted or not.

metadata:group
The name of a content item relation type group. It is used to classify **rel="relation"** links in entry resources. For an example of how this element is used, see section 5.12.

Atom Publishing Protocol for which no suitable standard elements could be found are included in the entries in a proprietary namespace identified by the URI **http://www.vizrt.com/atom-ext**. These include:

vaext:state
This Atom Protocol Extension elements represent any **app:control** identifies to represent the state of a resource. This will be present only when content item is not in **published** state and it's in either **approved** or **submitted** state. For example if a content item is in approved state he corresponding atom entry will contain the following XML fragment.

This element is used to represent additional content item states that cannot be represented by standard Atom or Atom publishing states. It appears as a child of the **app:control** element. For examples of how this element is used together with other state-related elements, see section 4.1.

4.1 Content Item State Representations

The Content Engine state concept is represented by a combination of two elements. The following examples show the combinations used to represent each state:

Draft

```
<app:control>
  <app:draft>yes</app:draft>
  <vaext:state name="draft"
               href="http://host-ip-address/webservice/escenic/content/state/
draft/editor"/>
</app:control>
```

Submitted

```
<app:control>
  <app:draft>yes</app:draft>
  <vaext:state name="submitted"
               href="http://host-ip-address/webservice/escenic/content/state/
submitted/editor"/>
</app:control>
```

Approved

```
<app:control>
  <app:draft>yes</app:draft>
  <vaext:state name="approved"
               href="http://host-ip-address/webservice/escenic/content/state/
approved/editor"/>
</app:control>
```

Published

```
<app:control>
  <app:draft>no</app:draft>
  <vaext:state name="published"
               href="http://host-ip-address/webservice/escenic/content/state/
published/editor"/>
</app:control>
```

Draft (with published)

```
<app:control>
  <app:draft>yes</app:draft>
  <vaext:state name="draft-published"
               href="http://host-ip-address/webservice/escenic/content/state/
draft-published/editor"/>
</app:control>
```

Submitted (with published)

```
<app:control>
  <app:draft>yes</app:draft>
  <vaext:state name="draft-submitted"
               href="http://host-ip-address/webservice/escenic/content/state/
draft-submitted/editor"/>
</app:control>
```

Approved (with published)

```
<app:control>
  <app:draft>yes</app:draft>
  <vaext:state name="draft-approved"
               href="http://host-ip-address/webservice/escenic/content/state/
draft-approved/editor"/>
</app:control>
```

Deleted

```
<app:control>
  <app:draft>yes</app:draft>
  <vaext:state name="deleted"
               href="http://host-ip-address/webservice/escenic/content/state/
deleted/editor"/>
</app:control>
```

The **`vaext:state`** element's **`href`** attribute contains a URL that can be used to retrieve a document containing allowed state transformations for the content item. For more information about this, see section 5.11.

5 How to...

This section contains examples showing the kinds of HTTP requests a client program can submit in order to achieve various common objectives, and the kinds of documents returned by the web service in response to these requests.

5.1 Get Started

No matter what your client program's purpose, the first thing it must always do is send an HTTP **GET** request to a suitable "start" URL, such as:

- The URL for the root sections of all your publications
- The URL of the specific publication you are interested in
- The tag hierarchy start URL (see section 5.18) if you want to work with your publications' tag hierarchy

5.1.1 Starting from the Root Sections URL

Submit an HTTP request like this:

```
curl -u user:password -X GET http://host-ip-address/webservice/escenic/section/ROOT/
subsections
```

This returns an Atom feed containing the root sections of all the publications to which your user has access. Your client program must then parse the feed and extract the URLs it needs. Below is an example of a root sections feed:

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <author>
    <name>Escenic Content Engine</name>
  </author>
  <category term="1234" scheme="http://myTaggingBackend/myTagStructure/"
 label="MyTag"/>
  <id>http://host-ip-address/webservice/escenic/section/ROOT/subsections</id>
  <link rel="self"
        href="http://host-ip-address/webservice/escenic/section/ROOT/subsections"
        type="application/atom+xml"/>
  <updated>2010-06-23T16:51:06.721Z</updated>
  <title type="text">Publication root sections you are authorized to.</title>
  <link rel="changelog"
        href="http://host-ip-address/webservice/escenic/changelog"
        type="application/atom+xml"/>
  <entry>
    <id>http://host-ip-address/webservice/escenic/section/1</id>
    <title type="text">frontpage</title>
    <updated>2010-06-22T10:16:46.309Z</updated>
    <category term="directory" scheme="http://www.vizrt.com/types"/>
    <link rel="edit-media"
          type="application/vnd.escenic.content+xml;type=com.escenic.section"
          href="http://host-ip-address/webservice/escenic/section/1"/>
    <link xmlns:thr="http://purl.org/syndication/thread/1.0"
          rel="down"
```

```
          href="http://host-ip-address/webservice/escenic/section/1/subsections"
          type="application/atom+xml"
          thr:count="8"/>
    <link rel="http://www.vizrt.com/types/relation/inboxes"
          href="http://host-ip-address/webservice/escenic/section/1/inboxes"
          type="application/atom+xml"/>
    <link rel="http://www.vizrt.com/types/relation/lists"
          href="http://host-ip-address/webservice/escenic/section/1/lists"
          type="application/atom+xml"/>
    <link rel="http://www.vizrt.com/types/relation/pages"
          href="http://host-ip-address/webservice/escenic/section/1/pages"
          type="application/atom+xml"/>
    <link rel="http://www.vizrt.com/types/relation/content-items"
          href="http://host-ip-address/webservice/escenic/section/1/content-items"
          type="application/atom+xml"/>
    <link rel="http://www.vizrt.com/types/relation/changelog"
          href="http://host-ip-address/webservice/escenic/changelog/section/1"
          type="application/atom+xml"/>
  </entry>
  ...[other entry elements representing other root sections omitted here]...
</feed>
```

Each entry represents the root section of a publication, and the links in each entry represent the operations available for that root section.

5.1.2 Starting from a Publication URL

Submit an HTTP request like this:

```
curl -u user:password -X GET http://host-ip-address/webservice/escenic/
publication/publication-name/
```

This returns a single Atom entry (not a feed) containing information about the requested publication. Your client program must then parse the entry and extract the URLs it needs. Below is an example of a publication entry:

```
<entry xmlns="http://www.w3.org/2005/Atom"
       xmlns:dcterms="http://purl.org/dc/terms/"
       xmlns:metadata="http://xmlns.escenic.com/2010/atom-metadata">
  <id>http://host-ip-address/webservice/escenic/publication/demopub/</id>
  <link rel="self"
        href="http://host-ip-address/webservice/escenic/publication/demopub/"
        type="application/atom+xml; type=entry"/>
  <link rel="edit"
        href="http://host-ip-address/webservice/escenic/publication/demopub/"
        type="application/atom+xml; type=entry"/>
  <link rel="http://www.vizrt.com/types/relation/lock"
        href="http://host-ip-address/webservice/escenic/lock/publication/9/e9319c09"
        type="application/atom+xml; type=entry"/>
  <dcterms:identifier>9</dcterms:identifier>
  <link href="http://host-ip-address/webservice/escenic/publication/demopub/"
        rel="http://www.vizrt.com/types/relation/top"
        type="application/atom+xml; type=entry"
        title="demopub"/>
  <link href="http://host-ip-address/webservice/escenic/publication/demopub/"
        rel="http://www.vizrt.com/types/relation/publication"
        type="application/atom+xml; type=entry" title="demopub"/>
```

```
  <metadata:publication href="http://host-ip-address/webservice/escenic/publication/
demopub/"
                        title="demopub"/>
  <content type="application/vnd.vizrt.payload+xml">
    <vdf:payload xmlns:vdf="http://www.vizrt.com/types"
                 model="http://host-ip-address/webservice/escenic/publication/demopub/
model/content-type/com.escenic.publication">
      <vdf:field name="com.escenic.publicationFeatures">
        <vdf:value>qualification.flaggedSectionUniqueName=flagged
usercontent.picture.file.maxsize=5242880
article.list.age.max=2880
article.list.age.default=2880
qualification.userQualificationEnabled=true
wf.conventionalInheritance=false
blueprint.active=blueprint
qualification.flaggingThreshold=5
multimedia.image.virtualVersions=true
        </vdf:value>
      </vdf:field>
    </vdf:payload>
  </content>
  <link href="http://demopub.host-ip-address/" rel="alternate"/>
  <link rel="down"
        href="http://host-ip-address/webservice/escenic/section/40"
        type="application/atom+xml; type=entry" title="Front Page"/>
  <title type="text">demopub</title>
</entry>
```

A publication entry carries a VDF payload containing the publication's feature settings. The entry's links represent the operations available for the publication. In particular, the publication entry's **down** link (highlighted) will return an entry for the publication's root section.

5.2 Navigate The Section Hierarchy

A client application can navigate the section hierarchy by recursively following all links with the **rel** attribute **down** (see section 3.1.3).

Root section 1 in the "start" Atom feed shown in section 5.1, for example, has a single **down** link that returns a new feed containing entries for all of section 1's subsections. So submitting the request:

```
curl -u user:password -X GET http://host-ip-address/webservice/escenic/section/1/
subsections
```

returns a new Atom feed like this:

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <author>
    <name>Escenic Content Engine</name>
  </author>
  <id>http://host-ip-address/webservice/escenic/section/1/subsections</id>
  <link rel="self"
        href="http://host-ip-address/webservice/escenic/section/1/subsections"
        type="application/atom+xml"/>
  <updated>2010-06-23T17:35:45.180Z</updated>
  <title type="text">Subsections for Section with id=1</title>
  <entry>
```

```
    <id>http://host-ip-address/webservice/escenic/section/22</id>
    <link rel="self"
          href="http://host-ip-address/webservice/escenic/section/22"
          type="application/atom+xml; type=entry"/>
    <link rel="edit"
          href="http://host-ip-address/webservice/escenic/section/22"
          type="application/atom+xml; type=entry"/>
    <link rel="http://www.vizrt.com/types/relation/lock"
          href="http://host-ip-address/webservice/escenic/lock/section/22"
          type="application/atom+xml; type=entry"/>
    <dcterms:identifier>22</dcterms:identifier>
    <link href="http://host-ip-address/webservice/escenic/publication/publication-id/"
          rel="http://www.vizrt.com/types/relation/publication"
          type="application/atom+xml; type=entry"
          title="publication-title"/>
    <metadata:publication href="http://host-ip-address/webservice/escenic/
publication/publication-id/" title="publication-title"/>
    <content type="application/vnd.vizrt.payload+xml">
      ...[section content omitted here]...
    </content>
    <title type="text">Escenic Times</title>
    <summary type="text">Escenic Times</summary>
    <app:edited>2012-08-06T14:03:30.527Z</app:edited>
    <link href="http://host-ip-address:8140/publication-id/" rel="alternate"/>
    <updated>2012-08-06T14:03:30.527Z</updated>
    <published>2007-08-31T05:47:44.797Z</published>
    <dcterms:created>2007-08-31T05:47:44.797Z</dcterms:created>
    <category term="directory" scheme="http://www.vizrt.com/types"/>
    <link xmlns:thr="http://purl.org/syndication/thread/1.0"
          rel="down"
          href="http://host-ip-address/webservice/escenic/section/22/subsections"
          type="application/atom+xml"
          thr:count="3"/>
    <link rel="http://www.vizrt.com/types/relation/inboxes"
          href="http://host-ip-address/webservice/escenic/section/22/inboxes"
          type="application/atom+xml"/>
    <link rel="http://www.vizrt.com/types/relation/lists"
          href="http://host-ip-address/webservice/escenic/section/22/lists"
          type="application/atom+xml"/>
    <link rel="http://www.vizrt.com/types/relation/pages"
          href="http://host-ip-address/webservice/escenic/section/22/pages"
          type="application/atom+xml"/>
    <link rel="http://www.vizrt.com/types/relation/content-items"
          href="http://host-ip-address/webservice/escenic/section/22/content-items"
          type="application/atom+xml"/>
    <link rel="http://www.vizrt.com/types/relation/changelog"
          href="http://host-ip-address/webservice/escenic/changelog/section/22"
          type="application/atom+xml"/>
    <link rel="http://www.vizrt.com/types/relation/active-page"
          href="http://host-ip-address/webservice/content/com.escenic.index-page/1362"
          type="application/vnd.escenic.content+xml; type=com.escenic.index-page"
          title="Daily News"/>
    <metadata:home-section>false</metadata:home-section>
  </entry>
  ...[other entry elements omitted here]...
</feed>
```

All the entries in this feed will contain a similar **down** link (highlighted in the above example) leading to further subsections that the client can follow. If a section has no subsections, then following the link will result in an empty feed like this:

```
<feed>
  <author>
    <name>Escenic Content Engine</name>
  </author>
  <id>http://host-ip-address/webservice/escenic/section/4161/subsections</id>
  <link rel="self"
        href="http://host-ip-address/webservice/escenic/section/4161/subsections"
        type="application/atom+xml"/>
  <updated>2012-09-11T12:37:38.172Z</updated>
  <title type="text">Subsections for Section with id=4161</title>
  <link href="http://host-ip-address/webservice/escenic/section/22/subsections"/>
</feed>
```

A client can therefore use these links to traverse the entire section tree of a publication.

5.3 Search For Content

Every entry in a subsection feed contains a link with the **rel** attribute **http://www.vizrt.com/types/relation/content-items**. This relationship indicates that the link points to a collection of all the content items belonging to the section. If a client application follows one of these links:

```
curl -u user:password -X GET http://host-ip-address/webservice/escenic/section/22/
content-items
```

what is returned is an empty feed that contains no actual content item entries, but does contain a link with the **rel** attribute **search**.

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <author>
    <name>Escenic Content Engine</name>
  </author>
  <id>http://host-ip-address/webservice/escenic/section/22/content-items</id>
  <link rel="self"
        href="http://host-ip-address/webservice/escenic/section/22/content-items"
        type="application/atom+xml"/>
  <updated>2010-06-03T12:19:57.058Z</updated>
  <author/>
  <title type="text">Content items for section with dbid="22"</title>
  <link rel="search"
        href="http://host-ip-address/webservice/open-search/escenic/22/content-search-
description.xml"
        type="application/opensearchdescription+xml"/>
</feed>
```

If the client application then follows this link:

```
curl -u user:password -X GET http://host-ip-address/webservice/open-search/escenic/22/
content-search-description.xml
```

The web service returns an OpenSearch document describing the URL format required to search through the section's content items:

```
<OpenSearchDescription xmlns="http://a9.com/-/spec/opensearch/1.1/">
  <ShortDescription>Escenic search</ShortDescription>
  <Description/>
  <Url type="application/atom+xml"
       template="http://host-ip-address/webservice/publication/publication-name/
search/escenic/22/
{searchTerms}/?pw={startPage?}&amp;c={count?}&amp;tag={tagIdentifier?}"/>
  <Url type="text/html"
       template="http://host-ip-address/webservice/publication/publication-name/
search/escenic/22/
{searchTerms}/?
pw={startPage?}&amp;c={count?}&amp;tag={tagIdentifier?}&amp;format=html"/>
  <Contact>http://www.escenic.com/</Contact>
  <Tags/>
  <LongName>Escenic Content Engine Search</LongName>
  <Image height="16" width="16" type="image/x-icon">http://host-ip-address/webservice/
images/ece.ico</Image>
  <Query role="example" searchTerms="cat"/>
  <Developer>Escenic AS</Developer>
  <Attribution/>
  <SyndicationRight>private</SyndicationRight>
  <AdultContent>false</AdultContent>
  <OutputEncoding>UTF-8</OutputEncoding>
  <InputEncoding>UTF-8</InputEncoding>
</OpenSearchDescription>
```

From this information the client can construct a URL that submits a query and specifies how the results are to be returned. Note that two different query templates are supplied: one that returns an Atom feed suitable for further processing, and one that returns HTML suitable for display. To obtain an Atom feed, for example, the client might submit the following request:

```
curl -u user:password -X GET
'http://host-ip-address/webservice/publication/publication-name/search/escenic/22/
Obama/?pw=1&count=10'
```

Tag searches

You can use the **`tag={tagIdentifier?}`** component of the search template to search for content items with a specific tag. Note that you must specify the identifier of the required tag, not its title.

5.4 Retrieve a Content Item

Once your client application has located a content item you are interested in, it can retrieve it very simply as follows:

```
curl -u user:password -X GET http://host-ip-address/webservice/escenic/content/43
```

The content item is returned as an Atom entry resource:

```
<entry xmlns="http://www.w3.org/2005/Atom"
       xmlns:app="http://www.w3.org/2007/app"
       xmlns:metadata="http://xmlns.escenic.com/2010/atom-metadata"
       xmlns:dcterms="http://purl.org/dc/terms/">
  <id>http://host-ip-address/webservice/escenic/content/43</id>
  <title type="text">Test</title>
  <app:edited>2010-06-23T09:09:50.654Z</app:edited>
```

```
  <dcterms:created>2010-06-22T10:22:20.000Z</dcterms:created>
  <author>
    <name>demo Administrator</name>
    <uri>http://host-ip-address/webservice/escenic/person/2</uri>
  </author>
  <dcterms:identifier>4</dcterms:identifier>
  <metadata:reference source="ece-auto-gen" sourceid="6d7203c9-27d5-4fce-b14a-
a466ead83875"/>
  <link rel="http://www.vizrt.com/types/relation/home-section"
        href="http://host-ip-address/webservice/escenic/section/4"
        title="New Articles"
        type="application/atom+xml; type=entry"/>
  <link href="http://wrk-ermo:12345/publication-id/incoming/article4.ece"
        rel="alternate"/>
  <link href="http://host-ip-address/webservice/escenic/lock/article/43"
        rel="http://www.vizrt.com/types/relation/lock"/>
  <link rel="http://www.vizrt.com/types/relation/publication"
        href="http://host-ip-address/webservice/escenic/publication/demo"
        title="demo"
        type="application/atom+xml; type=entry"/>
  <metadata:creator>
    <name>demo Administrator</name>
  </metadata:creator>
  <metadata:publication href="http://host-ip-address/webservice/escenic/publication/
demo">
    <link rel="http://www.vizrt.com/types/relation/home-section"
          href="http://host-ip-address/webservice/escenic/section/4"
          title="New Articles"
          type="application/atom+xml; type=entry"/>
    <link rel="http://www.vizrt.com/types/relation/section"
          href="http://host-ip-address/webservice/escenic/section/4"
          title="New Articles"
          type="application/atom+xml; type=entry"/>
  </metadata:publication>
  <link href="http://host-ip-address/webservice/escenic/content/43" rel="edit"/>
  <link href="http://host-ip-address/webservice/escenic/content/43" rel="self"/>
  <content type="application/vnd.vizrt.payload+xml">
    <vdf:payload xmlns:vdf="http://www.vizrt.com/types"
         model="http://host-ip-address/webservice/escenic/model/another">
      <vdf:field name="TITLE"><vdf:value>Test</vdf:value></vdf:field>
      <vdf:field name="BODY">
        <vdf:value>
            <div xmlns="http://www.w3.org/1999/xhtml">
              <p>This is a test</p>
            </div>
        </vdf:value>
      </vdf:field>
    </vdf:payload>
  </content>
</entry>
```

5.4.1 Retrieve the Published Copy of a Content Item

If staging is enabled then once a content item is published, it is effectively duplicated: there now exist two copies of the content item: a read-only published copy and an editable copy that exists in one of the various draft states (**`draft`**, **`submitted`** or **`approved`**). If your client application retrieves a content item using the standard content item URL, as described in section 5.4, then it will always be

the editable draft copy that is retrieved. You can, however, if required, retrieve the published copy by simply following the link with the relation **`rel="working-copy-of"`**. For example:

```
<link href="http://host-ip-address/webservice/escenic/content/6/published"
 rel="working-copy-of"/>
```

The returned Atom entry will have the same structure as if the default draft copy had been retrieved, but the state information in the **`app:control`** element will be different, and the content in the **`vdf:payload`** element may be different if the content item has been modified since publication. The entry will have a link with the relation **`rel="working-copy"`** that points back to the editable copy.

> The published copy of a content item is read-only: you cannot **`PUT`** (as described in section 5.6) to a **`working copy of`** URI.

5.5 Process a Content Item

Retrieved content items are embedded in Atom entry resources as Viz Data Format (VDF) payload documents:

```
<vdf:payload xmlns:vdf="http://www.vizrt.com/types"
     model="http://host-ip-address/webservice/escenic/model/another">
  <vdf:field name="TITLE"><vdf:value>Test</vdf:value></vdf:field>
  <vdf:field name="BODY">
    <vdf:value>
        <div xmlns="http://www.w3.org/1999/xhtml">
          <p>This is a test</p>
        </div>
    </vdf:value>
  </vdf:field>
</vdf:payload>
```

VDF is a proprietary format . Basically, content items are encoded as a sequence of **`vdf:field`** elements - one for each field in the content item. The exact structure of the VDF document therefore depends on the content type of the returned content item.

For simple cases where you know the structure of the content items to be retrieved, it may be sufficient for you to "hard-code" your client to deal with the expected content types. If, however, you want to make a generic client that can deal with any kind of content item, then you can do so by making use of the document's **model**.

The root element of a VDF payload document returned by the Content Engine always has a **`model`** attribute that contains a reference to a VDF model document. This document contains a schema defining the structure and content of the payload document. That is, it tells you what fields the payload document may contain, how they are organized and what data types they contain. Your client can therefore download this model document and use it as a guide to processing the contents of the payload document.

The field types defined in a Content Engine **`content-type`** resource correspond to VDF model field definitions as follows:

Escenic `field` definitions (in `content-type` resource)	Corresponding VDF model `fielddef`
`<field name="..." type="basic" mime-type="text/plain" .../>`	`<vdf:fielddef name="..." xsdtype="string" mediatype="text/plain"/>`
`<field name="..." type="basic" mime-type="application/xhtml+xml" .../>`	`<vdf:fielddef name="..." xsdtype="string" mediatype="application/xhtml+xml"/>`
`<field name="..." type="boolean" .../>`	`<vdf:fielddef name="..." xsdtype="boolean" mediatype="text/plain"/>`
`<field name="..." type="number" .../>`	`<vdf:fielddef name="..." xsdtype="decimal" mediatype="text/plain"/>`

`<field name="..." type="enumeration" multiple="false" .../>`	`<vdf:fielddef name="...">`
	`  <vdf:choice>`
	`  <vdf:alternative>...</vdf:alternative>`
	`  <vdf:alternative>...</vdf:alternative>`
	`  ...`
	`  </vdf:choice>`
	`</vdf:fielddef>`

`<field name="..." type="collection" mime-type="..." src="..." select="..." .../>`	`<vdf:fielddef name="..." xsdtype="string" mediatype="...">`
	`  <vdf:choice scope="limit">`
	`  <vdf:collection src="..." select="..."/>`
	`  </vdf:choice>`
	`</vdf:fielddef>`

`<field name="..." type="complex" ...>`
`  <complex>`
`  <field name="..." .../>`
`  <field name="..." .../>`
`  ...`
`  </complex>`
`</field>`
	`<vdf:fielddef name="...">`
	`  <vdf:fielddef name="..." .../>`
	`  <vdf:fielddef name="..." .../>`
	`  ...`
	`</vdf:fielddef>`

`<field`
`  name="..."`
`  type="number" ...`
`>`
`  <array`
`  default="3"`
`  max="10"`
`  />`
`</field>`
	`<vdf:fielddef name="...">`
	`  <vdf:listdef>`
	`  <vdf:schema>`
	`  <vdf:fielddef`
	`  name="..."`
	`  xsdtype="decimal"`
	`  mediatype="text/plain" />`
	`  </vdf:schema>`
	`  </vdf:listdef>`
	`  <ct:constraints>`

Escenic `field` definitions (in `content-type` resource)	**Corresponding VDF model `fielddef`**

	`<ct:minimum>0</ct:minimum>`
	`<ct:maximum>10</ct:maximum>`
	`<ct:default>3</ct:default>`
	`</ct:constraints>`
	`</vdf:fielddef>`

`<field name="..." type="link" .../>`	`<vdf:fielddef name="..." label="..." mediatype="`*mime-type-list*`" xsdtype="link" rel="edit-media"/>` where *mime-type-list* is a comma-separated list of allowed MIME types.
`<field name="..." type="schedule"/>`	not described in VDF model - see below.
`<field name="..." type="enumeration" multiple="true" .../>`	not described in VDF model - see below.

Schedule fields

Schedule fields are not described in the VDF model although they are included in the returned VDF payload document. They are encoded in Escenic syndication format, using the syndication format's **`schedule`** elements. For a detailed description of the **`schedule`** element and it's content model, see schedule:schedule.

The following example shows a content item Atom entry resource that contains a schedule field.

```
<entry xmlns="http://www.w3.org/2005/Atom"
       xmlns:app="http://www.w3.org/2007/app"
       xmlns:metadata="http://xmlns.escenic.com/2010/atom-metadata"
       xmlns:dcterms="http://purl.org/dc/terms/">
  <id>http://host-ip-address/webservice/escenic/content/43</id>
  ...
  <content type="application/vnd.vizrt.payload+xml">
    <vdf:payload xmlns:vdf="http://www.vizrt.com/types"
         model="http://host-ip-address/webservice/escenic/model/another">
      ...
      <vdf:field name="OPENING-HOURS">
        <vdf:value>
          <schedule xmlns="http://xmlns.escenic.com/2011/schedule" time-zone="Asia/
Almaty">
            <recurrence>
              <daily start-time="09:00:00" end-time="18:00:00"></daily>
              <range start-date="2011-05-13" end-date="2011-05-26"></range>
            </recurrence>
          </schedule>
        </vdf:value>
      <vdf:field>
    </vdf:payload>
  </content>
```

```
</entry>
```

Field options

Field options are included in the VDF payload document, embedded as sub-fields of the fields to which they apply. They are, however, currently not described in the VDF model. The following example shows a **`vdf:field`** element containing an embedded option field:

```
<vdf:field name="summary">
  <vdf:value>...</vdf:value>
  <vdf:field name="summary:css">
    <vdf:value>...</vdf:value>
  </vdf:field>
</vdf:field>
```

Note how the name of the embedded option field (**`summary:css`**) is formed by concatenating the name of the parent field and the name of the option, using a colon as a separator.

Multiple value enumeration fields

Multiple value enumeration fields are included in the VDF payload document, but they are not, however, currently described in the VDF model. The following example shows a multiple value enumeration field:

```
<vdf:field name="enum_multiple">
  <vdf:list>
    <vdf:payload>
      <vdf:field name="enum_multiple">
        <vdf:value>enum1</vdf:value>
      </vdf:field>
    </vdf:payload>
    <vdf:payload>
      <vdf:field name="enum_multiple">
        <vdf:value>enum2</vdf:value>
       </vdf:field>
    </vdf:payload>
  </vdf:list>
</vdf:field>
```

5.6 Change a Content Item

Once your client application has retrieved a content item, it can modify it and submit the changed version. The example in this section shows how to make the simplest of changes: a simple change to the title of the content item returned in section 5.4. If you are following this example using **`curl`** from the command line, you can simple copy the returned entry into a text editor and manually change the content of the **`<vdf:field name="title">`** element:

```
    <vdf:field name="title"><vdf:value>Edited Title</vdf:value></vdf:field>
```

and save the results to a file (called **`my-edited-item.xml`**, for example).

In order to submit the updated content item, you must **`PUT`** the file you have saved to the same URI from which it was retrieved:

```
curl --include -u user:password -X PUT -H "If-Match: *" -H "Content-Type: application/
atom+xml" \
> http://host-ip-address/webservice/escenic/content/4 --upload-file my-edited-
article.xml
HTTP/1.1 100 Continue

HTTP/1.1 200 OK
Date: Thu, 24 Jun 2010 09:23:24 GMT
Content-Type: application/atom+xml;type=entry
Content-Length: 0
Server: Jetty(6.1.19)
```

In order for the **PUT** operation to work, you must specify two HTTP headers as shown above:

Content-Type: application/atom+xml
: You must specify the content type of the data you are uploading.

If-Match: *
: The **If-Match** header value ***** is used here for reasons of simplicity. It is acceptable to use it for test and demonstration purposes, but should **never** be used in a production system. For more information about this header and what it does, see section 6.2.

If you're using **curl**, it's a good idea to specify **--include** with **PUT** operations: **curl** will then output the response header returned from the web service as shown above, and you can verify whether or not the operation was successful:

- A response code in the 2*xx* range indicates success.
- A response code in the 4*xx* range means that you made an invalid edit and the server won't accept your modification.
- A response code in the 5*xx* range means there is a server error.

> Submitting a change in this way may not work if the resource you are attempting to modify is already locked by another client application. For more information about this and about how locking works, see section 5.14.

5.6.1 Prevent Changes to The Last Modified Attribute

If you want to make a "silent" change to a content item that does not cause its **Last modified** attribute to be updated, you can do so by adding a **keep-last-modified="true"** attribute to the Atom entry's **app:control** element. A retrieved content item Atom entry always has an **app:control** element containing information about its current state. For example:

```
<app:control>
  <app:draft>yes</app:draft>
  <vaext:state name="draft-published"
               href="http://host-ip-address/webservice/escenic/content/state/draft-
published/editor"/>
</app:control>
```

All you have to do is add the attribute as follows to prevent the **Last modified** attribute being updated:

```
<app:control keep-last-modified="true">
  <app:draft>yes</app:draft>
  <vaext:state name="draft-published"
```

```
                href="http://host-ip-address/webservice/escenic/content/state/draft-
published/editor"/>
</app:control>
```

5.7 Create a Content Item

To create a new content item, your client application must create an Atom entry resource containing a Viz Data Format (VDF) payload document:

```
<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:app="http://www.w3.org/2007/app"
       xmlns:metadata="http://xmlns.escenic.com/2010/atom-metadata"
 xmlns:dcterms="http://purl.org/dc/terms/">
  <title type="text">My first item</title>
  <app:control>
    <app:draft>yes</app:draft>
  </app:control>
  <content type="application/vnd.vizrt.payload+xml">
    <vdf:payload xmlns:vdf="http://www.vizrt.com/types"
         model="http://host-ip-address/webservice/publication/pub-name/escenic/model/
content-type/content-type-name">
      <vdf:field name="title">
        <vdf:value>My first item</vdf:value>
      </vdf:field>
      <vdf:field name="summary">
        <vdf:value>This is a summary</vdf:value>
      </vdf:field>
      <vdf:field name="body">
        <vdf:value>
          <div xmlns="http://www.w3.org/1999/xhtml">
            <p>This is the body!</p>
          </div>
        </vdf:value>
      </vdf:field>
    </vdf:payload>
  </content>
</entry>
```

VDF is a proprietary format . Basically, content items are encoded as a sequence of **`vdf:field`** elements - one for each field in the content item. The fields in a content item are determined by its type (as defined in the publication **`content-type`** resource). In order to create a valid content item, therefore you need to know its type: the name of the type, what fields it can contain and what values are allowed in those fields.

VDF encompasses formats for both a **payload** document that contains actual data (as shown in the example above) and a **model** document that describes the structure of a payload document. A VDF model document, in other words, contains all the information you need to create a valid payload document of a particular type.

Given that you know the name of the publication you are creating a content item for and the name of its content type, you can retrieve the VDF model document that describes it as follows:

```
curl -u user:password -X GET http://host-ip-address/webservice/publication/pub-name/
escenic/model/content-type/content-type-name
```

where:

pub-name
is the name of the target publication

content-type-name
is the name of the target content type

The model document contains the information you need to create a content item of the specified type. For information about how different Escenic field types are described in a VDF model document, see section 5.5. You must include the URL of the model document in the **`model`** attribute of the root **`payload`** element of the document you create (as shown in the example above).

Save the document you have created in a file (**`my-new-item.xml`**, for example). In order to create the new content item you can then **`POST`** this file to the URI of section you want to add it to. For example:

```
curl --include -u user:password -X POST -H "Content-Type: application/atom+xml" \
> http://host-ip-address/webservice/escenic/section/section-id/content-items --upload-
file my-new-article.xml
HTTP/1.1 100 Continue
Server: Resin/3.1.11
Content-Length: 0
Date: Wed, 11 Jan 2012 14:16:47 GMT

HTTP/1.1 201 Created
Server: Resin/3.1.11
Location: http://host-ip-address/webservice/escenic/content/220771
Content-Type: application/octet-stream
Content-Length: 0
Date: Wed, 11 Jan 2012 14:16:47 GMT
```

where *section-id* is the ID of the section to which the content item is to be added.

In order for the **`POST`** operation to work, you must specify a HTTP header as shown above.

If you're using **`curl`**, it's a good idea to specify **`--include`** with **`PUT`** operations: **`curl`** will then output the response header returned from the web service as shown above, and you can verify whether or not the operation was successful:

- A response code in the 2*xx* range indicates success.
- A response code in the 4*xx* range means that you made an invalid addition and the server won't accept your new content item.
- A response code in the 5*xx* range means there is a server error.

The **`Location`** response specifies the location of the newly-created content item: you can retrieve it by submitting a **`GET`** request to this URL.

> A quick way of finding out how to create a correctly structured content item in VDF format is to **`GET`** a content item of the same type and copy the structure.

5.7.1 Creating Content Items in Different States

You can create a document in a state other than **draft** by replacing the **`app:draft`** element in the Atom entry resource that you upload with a **`vaext:state`** element that specifies the required state. To create a document that is immediately published, for example, you would need to replace:

```
<app:control>
  <app:draft>yes</app:draft>
</app:control>
```

with:

```
<app:control>
  <vaext:state xmlns:vaext="http://www.vizrt.com/atom-ext">published</vaext:state>
</app:control>
```

The **vaext** namespace prefix must be declared somewhere in the Atom file (for example on the element itself as shown above).

For a list of the various states that you can specify in this way, see section 5.11.1.

> The right to set content item states is governed by the Content Engine's permissions system. You can only set a state if the user creating the document has sufficient access rights.

5.7.2 Creating Binary Content items

To create a binary content item such as an image, video, sound clip or a Word or PDF attachment you first need to upload the binary object the content item is to contain. You do this by **POST**ing it to a fixed web service URL for binary attachments:

```
curl --include -u user:password -X POST -H "Content-Type: image/png" --upload-file my-
image.png \
> http://host-ip-address/webservice/escenic/binary
HTTP/1.1 100 Continue

HTTP/1.1 201 Created
Server: Apache-Coyote/1.1
X-ECE-Active-Connections: 1
Location: http://host-ip-address/webservice/escenic/binary/-1/2014/8/26/7/dc2dd7d4-
ab5d-46d7-8248-ee5b0c0d2268.bin
Content-Length: 0
Date: Tue, 26 Aug 2014 05:23:12 GMT
```

Make sure that the **Content-Type** header is set correctly for the type of file you are uploading. The **Location** field of the response contains the URL of the uploaded file. You can then use this URL to construct an Atom entry resource for the content item:

```
<?xml version="1.0" encoding="UTF-8"?>
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:app="http://www.w3.org/2007/app"
       xmlns:metadata="http://xmlns.escenic.com/2010/atom-metadata"
 xmlns:dcterms="http://purl.org/dc/terms/">
  <title type="text">basic-grid.png</title>
  <app:control>
    <app:draft>yes</app:draft>
  </app:control>
  <content type="application/vnd.vizrt.payload+xml">
    <vdf:payload xmlns:vdf="http://www.vizrt.com/types"
                 model="http://host-ip-address/webservice/publication/pub-name/
escenic/model/content-type/content-type-name">
      <vdf:field name="name">
        <vdf:value>basic-grid.png</vdf:value>
      </vdf:field>
      <vdf:field name="description"/>
```

```
      <vdf:field name="location"/>
      <vdf:field name="photographer"/>
      <vdf:field name="alttext"/>
      <vdf:field name="copyright"/>
      <vdf:field name="binary">
        <vdf:value>
        <link rel="edit-media"
        href="http://host-ip-address/webservice/escenic/
binary/-1/2014/8/25/15/7123b4f2-658b-4ec5-8ced-4e4e914679c7.bin"
              type="image/png"
              title="basic-grid.png"/>
        </vdf:value>
      </vdf:field>
      <vdf:field name="com.escenic.tags">
        <vdf:list/>
      </vdf:field>
    </vdf:payload>
  </content>
</entry>
```

Make sure that:

- The **`vdf:payload`** element's **`model`** attribute references an appropriate binary content type: in this case it needs to be a content type designed to hold images
- The **`link`** element's **`rel`** attribute is set to **`edit-media`**
- The **`link`** element's **`type`** attribute is set to the same value as you specified when uploading the binary object
- The **`link`** element's **`href`** attribute is set to URL returned when you uploaded the binary object

You can now upload the Atom entry resource in the same way as for any other content item (see section 5.7).

5.8 Preview a Content Item

Before your client application submits a modified content item you may want to allow the user to preview the modified version. The example in this section shows how to preview a simple change to the title of a content item.

The first objective is to retrieve the content item:

```
curl -u user:password -X GET http://host-ip-address/webservice/escenic/content/43
```

The content item is returned as an Atom entry resource:

```
<entry xmlns="http://www.w3.org/2005/Atom"
       xmlns:app="http://www.w3.org/2007/app"
       xmlns:metadata="http://xmlns.escenic.com/2010/atom-metadata"
       xmlns:dcterms="http://purl.org/dc/terms/">
  <id>http://host-ip-address/webservice/escenic/content/43</id>
  <title type="text">Test</title>
  <app:edited>2010-06-23T09:09:50.654Z</app:edited>
  <dcterms:created>2010-06-22T10:22:20.000Z</dcterms:created>
  <author>
    <name>demo Administrator</name>
```

```
    <uri>http://host-ip-address/webservice/escenic/person/2</uri>
  </author>
  <dcterms:identifier>4</dcterms:identifier>
  <metadata:reference source="ece-auto-gen" sourceid="6d7203c9-27d5-4fce-b14a-
a466ead83875"/>
  <link rel="http://www.vizrt.com/types/relation/home-section"
        href="http://host-ip-address/webservice/escenic/section/4"
        title="New Articles"
        type="application/atom+xml; type=entry"/>
  <link href="http://wrk-ermo:12345/publication-id/incoming/article4.ece"
        rel="alternate"/>
  <link href="http://host-ip-address/webservice/escenic/content/preview" rel="http://
www.vizrt.com/types/relation/preview"/>
  <link href="http://host-ip-address/webservice/escenic/lock/article/43"
        rel="http://www.vizrt.com/types/relation/lock"/>
  <link rel="http://www.vizrt.com/types/relation/publication"
        href="http://host-ip-address/webservice/escenic/publication/demo"
        title="demo"
        type="application/atom+xml; type=entry"/>
  <metadata:creator>
    <name>demo Administrator</name>
  </metadata:creator>  o
  <metadata:publication href="http://host-ip-address/webservice/escenic/publication/
demo">
    <link rel="http://www.vizrt.com/types/relation/home-section"
          href="http://host-ip-address/webservice/escenic/section/4"
          title="New Articles"
          type="application/atom+xml; type=entry"/>
    <link rel="http://www.vizrt.com/types/relation/section"
          href="http://host-ip-address/webservice/escenic/section/4"
          title="New Articles"
          type="application/atom+xml; type=entry"/>
  </metadata:publication>
  <link href="http://host-ip-address/webservice/escenic/content/43" rel="edit"/>
  <link href="http://host-ip-address/webservice/escenic/content/43" rel="self"/>
  <content type="application/vnd.vizrt.payload+xml">
    <vdf:payload xmlns:vdf="http://www.vizrt.com/types"
         model="http://host-ip-address/webservice/escenic/model/another">
      <vdf:field name="TITLE"><vdf:value>Test</vdf:value></vdf:field>
      <vdf:field name="BODY">
        <vdf:value>
            <div xmlns="http://www.w3.org/1999/xhtml">
              <p>This is a test</p>
            </div>
        </vdf:value>
      </vdf:field>
    </vdf:payload>
  </content>
</entry>
```

Your client application can now modify the title of the content item (highlighted in the above listing). If you are following this example using **`curl`** from the command line, you can simple copy the returned entry into a text editor and manually change the content of the **`<vdf:field name="TITLE">`** element:

```
    <vdf:field name="TITLE"><vdf:value>Edited title</vdf:value></vdf:field>
```

and save the results to a file (called **`my-edited-item.xml`**, for example).

In order to submit the updated content item, you must **POST** the file you have saved to the preview URI identified by the **http://www.vizrt.com/types/relation/preview** relation:

```
curl --include -u user:password -X POST -H "Content-Type: application/atom+xml" \
> http://host-ip-address/webservice/escenic/content/preview --upload-file my-edited-
article.xml
HTTP/1.1 100 Continue

HTTP/1.1 201 Created
Date: Mon, 13 Dec 2010 09:23:24 GMT
Location: http://host-ip-address/incoming/article1234.ece?token=-1723563771
Server: Jetty(6.1.19)
```

In order for the **POST** operation to work, you must specify one HTTP header as shown above:

Content-Type: application/atom+xml
: You must specify the content type of the data you are uploading.

- A **201** response code indicates success. The **location** header contains the URL of the content item preview.
- A response code in the *4xx* range means that you made an invalid edit and the server won't accept your modification.
- A response code in the *5xx* range means there is a server error.

5.9 Tag a Content Item

tags are added to a content item using a field called **com.escenic.tags**. This field is a ordered list constructed of payloads. The order of the list is kept. The payload itself is constructed of two fields.

tag
: The **href** attribute of the **origin** element is used to identify the tag. The value of the href attribute can be derefenced to get the tag entry. The **value** element contains the read only value of the tag title.

```
<field name="tag">
  <origin href="http://example.com/tags/cn"/>
  <value>Celebrity News</vdf:value>
</vdf:field>
```

relevance
: The relevance of the tag in regards to this content item. Relevance is a number between 0.0 and 1.0 inclusive. If the **relevance** field is omitted the server will return with a **400 Bad request** response.

```
<field name="relevance">
  <value>0.7</vdf:value>
</vdf:field>
```

When you **GET** the entry after a successful **PUT** or **POST**, the resulting atom entry will contain atom **category** elements in addition to the list of tags in the **com.escenic.tag** field. This is a informational feature offered for generic Atom processors.

```
<category scheme="tag:geographic@escenic.com,2011-04-28" term="Oslo" label="Oslo"
 title="Tag: Europe / Norway / Oslo"/>
```

To add tags to an existing content item:

1. Retrieve the content item as an Atom entry resource (as described in section 5.6).
2. Add the **`com.escenic.tags`** field to the entry or replace it if the content item was was tagged already.

```
<field name="com.escenic.tags">
  <list> <!-- order is kept -->
    <payload>
      <field name="tag">
        <origin href="http://example.com/mj"> <!-- URL of the tag entry -->
        <value>Michael Jackson</value> <!-- read only value of the tag "title" --
>
      </field>
      <field name="relevance"> <!-- mandatory -->
        <value>1.0</value> <!-- must be floating point between (inclusive) 0.0 and
 1.0 -->
      </field>
    </payload>
    <payload>
      <field name="tag">
        <origin href="http://example.com/tags/hc">
        <value>Holy Cow</value>
      </field>
      <field name="relevance>
        <value>0.8</value>
      </field>
    </payload>
    <payload>
      <field name="tag">
        <origin href="http://example.com/tags/cn">
        <value>Celebrity News</value>
      </field>
      <field name="relevance>
        <value>0.7</value>
      </field>
    </payload>
  </list>
</field>
```

3. Save the modified entry in a file.
4. **`PUT`** the file back to the same URI from which it was retrieved (as described in section 5.6).

5.10 Add a Content Item to a Section

The process of adding a content item to a section (or removing it from one) is no different from any other change you might make to a content item. You simply:

1. Retrieve the content item as an Atom entry resource (as described in section 5.6).
2. Insert one or more Atom **`link`** elements to the required sections into the retrieved entry, or remove unwanted **`link`** elements. **`link`** elements must be inserted as children of **`metadata:publication`** elements.

3. Save the modified entry in a file.
4. PUT the file back to the same URI from which it was retrieved (as described in section 5.6).

The listing below shows a retrieved content item entry in which the **metadata:publication** element is highlighted:

```
<entry xmlns="http://www.w3.org/2005/Atom"
       xmlns:app="http://www.w3.org/2007/app"
       xmlns:metadata="http://xmlns.escenic.com/2010/atom-metadata"
       xmlns:dcterms="http://purl.org/dc/terms/">
  <id>http://host-ip-address/webservice/escenic/content/43</id>
  <title type="text">Test</title>
  <app:edited>2010-06-23T09:09:50.654Z</app:edited>
  <dcterms:created>2010-06-22T10:22:20.000Z</dcterms:created>
  <author>
    <name>demo Administrator</name>
    <uri>http://host-ip-address/webservice/escenic/person/2</uri>
  </author>
  <dcterms:identifier>4</dcterms:identifier>
  <metadata:reference source="ece-auto-gen" sourceid="6d7203c9-27d5-4fce-b14a-
a466ead83875"/>
  <link rel="http://www.vizrt.com/types/relation/home-section"
        href="http://host-ip-address/webservice/escenic/section/4"
        title="New Articles"
        type="application/atom+xml; type=entry"/>
  <link href="http://wrk-ermo:12345/publication-id/incoming/article4.ece"
        rel="alternate"/>
  <link href="http://host-ip-address/webservice/escenic/content/preview" rel="http://
www.vizrt.com/types/relation/preview"/>
  <link href="http://host-ip-address/webservice/escenic/lock/article/43"
        rel="http://www.vizrt.com/types/relation/lock"/>
  <link rel="http://www.vizrt.com/types/relation/publication"
        href="http://host-ip-address/webservice/escenic/publication/demo"
        title="demo"
        type="application/atom+xml; type=entry"/>
  <metadata:creator>
    <name>demo Administrator</name>
  </metadata:creator>
  <metadata:publication href="http://host-ip-address/webservice/escenic/publication/
demo">
    <link rel="http://www.vizrt.com/types/relation/home-section"
          href="http://host-ip-address/webservice/escenic/section/4"
          title="New Articles"
          type="application/atom+xml; type=entry"/>
    <link rel="http://www.vizrt.com/types/relation/section"
          href="http://host-ip-address/webservice/escenic/section/4"
          title="New Articles"
          type="application/atom+xml; type=entry"/>
  </metadata:publication>
  <link href="http://host-ip-address/webservice/escenic/content/43" rel="edit"/>
  <link href="http://host-ip-address/webservice/escenic/content/43" rel="self"/>
  <content type="application/vnd.vizrt.payload+xml">
    <vdf:payload xmlns:vdf="http://www.vizrt.com/types"
         model="http://host-ip-address/webservice/escenic/model/another">
      <vdf:field name="TITLE"><vdf:value>Test</vdf:value></vdf:field>
      <vdf:field name="BODY">
        <vdf:value>
            <div xmlns="http://www.w3.org/1999/xhtml">
```

```
              <p>This is a test</p>
            </div>
          </vdf:value>
        </vdf:field>
      </vdf:payload>
    </content>
  </entry>
```

The **metadata:publication** element contains two **link** elements. Their **href** attributes both reference the same section, but they have different **rel** attribute settings:

http://www.vizrt.com/types/relation/section
: This relationship indicates that the content item belongs to a section. The **metadata:publication** element may contain several **link**s of this type, one for each section to which the content item belongs.

http://www.vizrt.com/types/relation/home-section
: This relationship indicates that the referenced section is the content item's home section. The **metadata:publication** element may contain only one **link** of this type.

To add the content item to another section, you simply insert another **http://www.vizrt.com/types/relation/section link** element:

```
<metadata:publication href="http://host-ip-address/webservice/escenic/publication/
demo">
  <link rel="http://www.vizrt.com/types/relation/home-section"
        href="http://host-ip-address/webservice/escenic/section/4"
        title="New Articles"
        type="application/atom+xml; type=entry"/>
  <link rel="http://www.vizrt.com/types/relation/section"
        href="http://host-ip-address/webservice/escenic/section/4"
        title="New Articles"
        type="application/atom+xml; type=entry"/>
  <link rel="http://www.vizrt.com/types/relation/section"
        href="http://host-ip-address/webservice/escenic/section/5"
        title="New Articles"
        type="application/atom+xml; type=entry"/>
</metadata:publication>
```

Note that the **link** element you add must reference an **existing** section.

5.10.1 Cross-Published Content Items

A cross-published content item contains several **metadata:publication** elements, one for each publication in which the content item is published.

When you are adding a section **link**, to such a content item, however, you do not need to worry about which **metadata:publication** element you add it to. The Content Engine already knows which publication the specified section belong to. It simply adds the section link to the content item, and ignores the parent **metadata:publication** element.

5.11 Change Content Item State

The Atom entry resource representing a retrieved content item always contains an **app:control** element containing information about the current state of the content item. For example:

```
<app:control>
  <app:draft>yes</app:draft>
  <vaext:state name="draft-published"
               href="http://host-ip-address/webservice/escenic/content/state/draft-
published/editor"/>
</app:control>
```

The important element here is the **vaext:state** element. It always has the following two attributes:

name
: The name of the current state.

href
: The URI of a resource containing information about the state transitions that may be performed by the current user. If you look at the example above, you will see that the last two parts of the URI are **draft-published** (the current state) and **editor** (the role of the user that submitted the request). If the user who retrieved this content item had journalist access rights, then the last part of this URI would be **journalist**.

In order to change the state of a content item therefore, a client application needs to:

1. Retrieve the content item as described in section 5.4.
2. Retrieve the list of possible state transitions for the content item using the URI in the **vaext:state** element. See section 5.11.1 for an example of what this resource looks like.
3. Select the required new state from the list of possible transitions.
4. Insert the new state into the **vaext:state** element of the retrieved content item. For example:

   ```
   <vaext:state name="draft-published"
                href="http://host-ip-address/webservice/escenic/content/state/draft-
   published/editor">
     draft-submitted
   </vaext:state
   ```

5. **PUT** the modified content item back as described in section 5.6.

If content item staging is disabled, then:

- The **vaext:state** element's **href** attribute will contain a slightly different URI containing the string **legacy**.
- The returned resource will contain a different (smaller) set of possible state transitions.

5.11.1 Example State Transitions Document

Following the **vaext:name href** URI shown in section 5.11 will return a resource with the following content:

```
<state xmlns="http://xmlns.escenic.com/2013/content-state" xmlns:ui="http://
xmlns.escenic.com/2008/interface-hints">
  <name>draft-published</name>
  <ui:label>Draft (with published)</ui:label>
```

```
  <ui:icon>http://host-ip-address/webservice/escenic/content/state/icon/draft-
published.png</ui:icon>
  <actions>
    <action>published</action>
    <action>revert</action>
    <action>submitted-published</action>
    <action>approved-published</action>
    <action>un-publish-keep</action>
    <action>un-publish-revert</action>
    <action>deleted</action>
  </actions>
  <choices>
    ...
  </choices>
</state>
```

The actual allowed state values are listed in the **`/state/actions/action`** elements at the top of the file. Most of the rest of the file consists of a **`choices`** element containing a more detailed representation of the same information. (The **`choices`** element actually contains an abstract representation of the flow control user interface that would be displayed by Content Studio for this content item.)

5.12 Follow Content Item Relations

This section shows how a content item's relations are included in Atom entry resources, and how you can navigate from a content item to it's related content items.

If a retrieved content item has related content items, then they are always included in the returned Atom entry resource. They appear as **`link`** elements, with the **`rel`** attribute set to **`related`**. The following example shows the Atom entry resource for a content item with two related content items (highlighted in **bold**):

```
<entry xmlns="http://www.w3.org/2005/Atom"xmlns:vdf="http://www.vizrt.com/types"
       xmlns:ns2="http://www.w3.org/1999/xhtml" xmlns:ns="http://www.w3.org/2005/Atom"
       xmlns:metadata="http://xmlns.escenic.com/2010/atom-metadata"
 xmlns:dcterms="http://purl.org/dc/terms/"
       xmlns:app="http://www.w3.org/2007/app">
  <id>http://host-ip-address/webservice/escenic/content/5403</id>
  <title type="text">UK and US terror alerts</title>
  <updated>2010-10-03T22:07:41.000Z</updated>
  <app:edited>2011-01-14T14:24:16.000Z</app:edited>
  <published>2010-10-03T22:07:41.000Z</published>
  <dcterms:created>2010-10-03T22:07:41.000Z</dcterms:created>
  <author>
    <name>Mark mark</name>
    <uri>http://host-ip-address/webservice/escenic/person/103</uri>
  </author>
  <dcterms:identifier>5403</dcterms:identifier>
  <metadata:reference source="ece-auto-gen"
                      sourceid="38f4f7a2-c2bb-4026-a79f-3bde3c3592df" />
  <link href="http://host-ip-address/webservice/escenic/section/44"
        rel="http://www.vizrt.com/types/relation/home-section" title="News"
        type="application/atom+xml; type=entry" />
  <link href="http://ecedemo5/wf-escenic-times/news/article5403.ece"
        rel="alternate" />
  <link href="http://host-ip-address/webservice/escenic/publication/wf-escenic-times"
```

```
        rel="http://www.vizrt.com/types/relation/publication"
        title="wf-escenic-times" type="application/atom+xml; type=entry" />
  <metadata:creator>
    <name>Mark mark</name>
  </metadata:creator>
  <metadata:publication href="http://host-ip-address/webservice/escenic/publication/
wf-escenic-times">
    <link href="http://host-ip-address/webservice/escenic/section/44"
          rel="http://www.vizrt.com/types/relation/home-section" title="News"
          type="application/atom+xml; type=entry" />

    <link href="http://host-ip-address/webservice/escenic/section/44"
          rel="http://www.vizrt.com/types/relation/section" title="News"
          type="application/atom+xml; type=entry" />
  </metadata:publication>
  <link href="http://host-ip-address/webservice/escenic/content/5403" rel="edit" />
  <link href="http://host-ip-address/webservice/escenic/content/5403" rel="self" />
  <link href="http://host-ip-address/webservice/escenic/lock/article/5403"
        rel="http://www.vizrt.com/types/relation/lock" />
  <content type="application/vnd.vizrt.payload+xml">
    <vdf:payload model="http://host-ip-address/webservice/escenic/model/news">
      ...content...
    </vdf:payload>
  </content>
  <link href="http://host-ip-address/webservice/escenic/content/1513"
        rel="related" type="application/atom+xml; type=entry"
        metadata:group="STORYREL">
    <vdf:payload model="http://host-ip-address/webservice/content-summaries/news">
      <vdf:field name="TITLE">
        <vdf:value>Obama rejects 'Biden option' on Afghanistan</vdf:value>
      </vdf:field>
      <vdf:field name="LEADTEXT">
        <vdf:value>President Obama ....</vdf:value>
      </vdf:field>
    </vdf:payload>
  </link>
  <link href="http://host-ip-address/webservice/escenic/content/5391"
        rel="related" type="application/atom+xml; type=entry"
        metadata:group="STORYREL">
    <vdf:payload model="http://host-ip-address/webservice/content-summaries/news">
      <vdf:field name="TITLE">
        <vdf:value>Travel alert issued for U.S. citizens in Europe</vdf:value>
      </vdf:field>

      <vdf:field name="LEADTEXT">
        <vdf:value>The United States ...</vdf:value>
      </vdf:field>
    </vdf:payload>
  </link>
</entry>
```

The **`rel="related"`** attribute setting indicates that the link represents one of the content item's relations. The **`metadata:group`** attribute specifies the relation type group that the relation belongs to. Relation type groups are defined in the **`content-type`** publication resource (see relation-type-group) and appear as separate relation drop zones in Content Studio.

Note that each relation **`link`** element also contains the relation's summary fields packaged in a VDF **`payload`** element. This makes it possible for interactive clients to display the relation summaries along with a content item, in the same way as Content Studio.

A client can retrieve related content items by following the links, for example:

```
curl -u user:password -X GET http://host-ip-address/webservice/escenic/content/1513
```

5.13 Retrieve Content Item History

Client applications can retrieve an Atom feed containing the edit history of a content item. The history feed contains an entry for each modification made to the content item. An entry in the feed contains the time a change was made, the name of the user who made the change and information about any state change applied to the content item.

> If content item staging is enabled, then once a content item is published, any changes made to the draft copy of the content item are not recorded in the history. An entry is only add to the history feed if such changes are published.

A link to an edit history feed is included in every content item's Atom entry resource:

```
<entry>
  ...
  <link href="http://host-ip-address/webservice/escenic/content/1337/log"
        rel="http://www.vizrt.com/types/relation/log"/>
  ...
</entry>
```

A client can retrieve the history feed by following the link, for example:

```
curl -u user:password -X GET http://host-ip-address/webservice/escenic/content/1337/
log
```

This will return something like:

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <author>
    <name>Escenic Content Engine</name>
  </author>
  <id>http://host-ip-address:8080/webservice/escenic/content/1337/log</id>
  <link rel="self" href="http://host-ip-address:8080/webservice/escenic/content/1337/
log" type="application/atom+xml"/>
  <updated>2012-12-05T08:34:53.041Z</updated>
  <title type="text">Article log</title>
  <link rel="next" href="http://host-ip-address:8080/webservice/escenic/content/1337/
log/after/2846?count=10"/>
  <link rel="previous" href="http://host-ip-address:8080/webservice/escenic/
content/1337/log/before/2870?count=10"/>
  <entry>
    <updated>2012-11-29T12:33:16.000Z</updated>
    <author>John Doe</author>
  </entry>
  <entry>
    <updated>2012-11-29T12:32:48.000Z</updated>
```

```
    <app:control xmlns:app="http://www.w3.org/2007/app" xmlns:vaext="http://
www.vizrt.com/atom-ext">
      <app:draft>yes</app:draft>
      <vaext:state>submitted</vaext:state>
    </app:control>
    <author>John Doe</author>
  </entry>
  <entry>
    <updated>2012-11-26T09:35:27.000Z</updated>
    <author>John Doe</author>
  </entry>
  <entry>
    <updated>2012-11-26T09:35:17.000Z</updated>
    <author></author>
  </entry>
</feed>
```

The edit history is returned as an Atom paged collection. The first entry in the first Atom feed returned represents the most recent change to the content item. The next entry represents the second most recent change, and so on. If there is more than one page of entries, then you can retrieve the next page by following this feed's **`next`** link.

> An entry with an empty **`author`** field represents a change made by an automated process such as the syndication module or a background service.

5.14 Lock a Content Item

A lock is a piece of advice to client applications that some other client is currently modifying all or part of a content item. The correct use of locks ensures that no two clients will simultaneously update the same information. A content item can be locked with two different types of lock:

Resource lock
: Locks the entire resource (content item). No other locks can be set on this resource.

Fragment lock
: Locks only part of the resource (one field of a content item). Several fragment locks can be set on the same content item as long as they lock different fields. A resource lock, however, cannot be set on a content item locked in this way.

An interactive client such as Content Studio is expected to lock a content item with the appropriate type of lock as soon as a user starts making a modification, and unlock it again when the user either saves the change or explicitly abandons it (for example by clicking on a **Cancel** button). Similarly, an automated client should lock a content item as soon as it "intends" to carry out an update, both to inform other clients of its intentions and to avoid conflicts when it actually performs the update.

The following terms are used when talking about locks:

Acquire
: Locking all or part of a content item is described as **acquiring a lock**.

Release
: Unlocking all or part of a content item is described as **releasing a lock**. The term **forced release** is used to describe release of a lock owned by another client.

Lock URI
Locks are exposed by the Content Engine's web service as URIs called **lock URIs**. There are three types of lock URI:

lock collection URI
This URI is returned in a content item resource entry, and is the start point for locking a content item.

Public lock URI
These URIs appear in entries returned from a lock collection URI, and simply indicate that a content item is locked.

Private lock URI
This URI is returned to the client that acquires a lock, and is effectively the key to the lock.

Updating a content item involves the following steps:

1. **`GET`** the resource you want to lock.
2. Optionally, **`GET`** the resource's public locks (if any) from the resource's lock collection URI.
3. If the resource's existing public locks do not conflict with the lock you require, **`POST`** a lock request to the lock collection URI. If successful, the web service returns a private lock URI.
4. **`PUT`** the modified version of the resource. In order for the **`PUT`** request to succeed you must include the private lock URI in its HTTP header.
5. **`DELETE`** the private lock URI in order to release your lock.

These steps are described in greater detail in the following sections.

5.14.1 Get the Resource to Lock

Your client application must first **`GET`** the resource to be locked in order to discover its lock collection URI:

```
curl -u user:password http://host-ip-address/webservice/escenic/content/4
```

The returned resource entry contains a link with the relation type section 3.2.9 This link contains the lock collection URI:

```
<entry>
  ...
  <link href="http://host-ip-address/webservice/escenic/lock/article/4"
        rel="http://www.vizrt.com/types/relation/lock"/>
  ...
</entry>
```

5.14.2 Get Public Locks

Before the client attempts to lock the resource it can optionally check to see whether or not a conflicting lock is already set. This is done by sending a **`GET`** request to the lock collection URI. For example:

```
curl -u user:password http://host-ip-address/webservice/escenic/lock/article/4
```

The following example shows a response indicating that the resource is already locked:

```
<feed xmlns="http://www.w3.org/2005/Atom">
```

```
  <author>
    <name>Escenic Content Engine</name>
  </author>
  <id>http://host-ip-address/webservice/escenic/lock/article/4</id>
  <link rel="self" href="http://host-ip-address/webservice/escenic/lock/article/4"
 type="application/atom+xml"/>
  <updated>2010-12-04T08:05:08.474Z</updated>
  <entry xmlns:age="http://purl.org/atompub/age/1.0" xmlns:metadata="http://
xmlns.escenic.com/2010/atom-metadata">
    <id>http://host-ip-address/webservice/escenic/lock/article/4/public/72</id>
    <title type="text">Lock of the fragment SUMMARY</title>
    <link href="http://host-ip-address/webservice/escenic/lock/article/4/public/72"
 rel="self"/>
    <metadata:fragment>SUMMARY</metadata:fragment>
    <author>
      <uri>http://host-ip-address/webservice/escenic/person/1</uri>
      <name>testpub Administrator</name>
    </author>
    <updated>2010-12-04T04:49:18.000Z</updated>
    <age:expires>2010-12-05T04:49:18.000Z</age:expires>
    <summary type="text">Lock created by: lpt-sai/10.211.10.176</summary>
    <content type="text">The fragment 'SUMMARY' of the resource is locked by testpub
 Administrator</content>
  </entry>
</feed>
```

The returned feed contains an entry with the public lock URI **`http://host-ip-address/webservice/escenic/lock/article/4/public/72`**. The **`metadata:fragment`** element, however, indicates that this is a fragment lock that only locks the content item's **`SUMMARY`** field. If the client does not intend to lock the same fragment or the whole resource, then it can still acquire a lock.

> This step is optional: a client can POST a lock request without first checking the existing locks, and in many cases it may be most efficient to do so. An interactive clients will, however, often need to **`GET`** this list of public locks in order to present the information in its user interface.

5.14.3 Post Lock Request

To acquire a lock the client must send a **`POST`** request to the resource's lock collection URI. The posted file must contain a valid Atom entry element like this:

```
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:metadata="http://
xmlns.escenic.com/2010/atom-metadata">
  <summary type="text">Lock created by: User1 from WS</summary>
  <metadata:fragment>TITLE</metadata:fragment>
</entry>
```

The **`metadata:fragment`** element is only required when requesting a fragment lock. Omit it to request a resource lock.

The request must contain a **`Content-Type`** header the value **`application/atom+xml;type=entry`**. A valid request might look like this:

```
curl --include -u user:password -X POST -H "Content-Type: application/atom
+xml;type=entry" \
     http://host-ip-address/webservice/escenic/lock/article/4 \
     --upload-file atom-entry-file
```

where *atom-entry-file* is the path of a file containing an Atom entry like the one shown above.

If the lock request succeeds, then the web service returns a **`201 Created`** response in which **`Location`** contains the resource's private lock URI:

```
HTTP/1.1 201 Created
Server: Apache-Coyote/1.1
Location: http://host-ip-address/webservice/escenic/lock/article/4/private/9
Content-Length: 0
Date: Sat, 04 Dec 2010 02:35:38 GMT
```

If the lock request fails because the requested resource is already locked someone else has already locked the resource then the web service returns a **`409 CONFLICT`** response.

Once the client has acquired the private lock URI, it can proceed with the update as described in section 5.14.4. If, however, the client needs information about the lock for some reason, it can retrieve the lock resource by sending a **`GET`** request to the private lock URI:

```
curl --include -u user:password http://host-ip-address/webservice/escenic/lock/
article/4/private/9
```

The web service then returns an entry containing information about the lock:

```
<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:age="http://purl.org/atompub/age/1.0"
 xmlns:metadata="http://xmlns.escenic.com/2010/atom-metadata">
  <id>http://host-ip-address/webservice/escenic/lock/article/4/public/39</id>
  <title type="text">Lock of the fragment TITLE</title>
  <link href="http://host-ip-address/webservice/escenic/lock/article/4/public/39"
 rel="self"/>
  <metadata:fragment>TITLE</metadata:fragment>
  <author>
    <uri>http://host-ip-address/webservice/escenic/person/1</uri>
    <name>pub1 Administrator</name>
  </author>
  <updated>2010-12-04T02:35:39.000Z</updated>
  <age:expires>2010-12-05T02:35:39.000Z</age:expires>
  <summary type="text"/>
  <content type="text">The fragment 'TITLE' of the resource is locked by pub1
 Administrator</content>
</entry>
```

Note that the content of this resource is identical to the content of the public lock resource (the **`self`** link, for example, contains the URI of the public lock rather than the private lock).

5.14.4 Put Updated Resource

Once the client has a private lock URI, it can **`PUT`** an update request. The request must contain the following headers:

- **`Content-Type`**, set to **`application/atom+xml;type=entry`**
- **`X-Escenic-Locks`**, containing the private lock URI. If you want to request several fragment locks at once, you can do so by repeating this header.
- **`If-Match`**, set to the appropriate **`E-Tag`** value. See section 6.2 for more information about **`E-Tag`** and the use of the **`If-Match`** header.

Here is an example update request:

```
curl --include -u user:password -X PUT \
-H "Content-Type: application/atom+xml;type=entry" \
-H "X-Escenic-Locks: http://host-ip-address/webservice/escenic/lock/article/4/
private/9" \
-H 'If-Match: "129694559e911ee4c6d04212982"' \
http://host-ip-address/webservice/escenic/content/4 --upload-file atom-entry-file
```

where *atom-entry-file* is the path of a file containing the updated resource:

```
<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:app="http://www.w3.org/2007/app"
 xmlns:metadata="http://xmlns.escenic.com/2010/atom-metadata" xmlns:dcterms="http://
purl.org/dc/terms/">
  <id>http://host-ip-address/webservice/escenic/content/4</id>
  <title type="text">My Updated Title</title>
  <updated>2010-12-01T10:13:00.000Z</updated>
  <app:edited>2010-12-03T10:13:06.000Z</app:edited>
  <published>2010-12-01T10:13:00.000Z</published>
  <dcterms:created>2010-12-01T10:13:00.000Z</dcterms:created>
  <author>
    <name>Marcus Tullius Cicero</name>
    <uri>http://host-ip-address/webservice/escenic/person/2</uri>
  </author>
  <dcterms:identifier>4</dcterms:identifier>
  <metadata:reference source="ece-auto-gen" sourceid="0b8a67bc-7b2d-493b-b6b5-
d659f1beff81"/>
  <link rel="http://www.vizrt.com/types/relation/home-section" href="http://host-ip-
address/webservice/escenic/section/3" title="Examples" type="application/atom+xml;
 type=entry"/>
  <link href="http://host-ip-address/pub1/Examples/article4.ece" rel="alternate"/>
  <link rel="http://www.vizrt.com/types/relation/publication" href="http://host-ip-
address/webservice/escenic/publication/pub1" title="pub1" type="application/atom+xml;
 type=entry"/>
  <metadata:creator/>
  <metadata:publication href="http://host-ip-address/webservice/escenic/publication/
pub1">
    <link rel="http://www.vizrt.com/types/relation/home-section" href="http://host-
ip-address/webservice/escenic/section/3" title="Examples" type="application/atom+xml;
 type=entry"/>
    <link rel="http://www.vizrt.com/types/relation/section" href="http://host-ip-
address/webservice/escenic/section/3" title="Examples" type="application/atom+xml;
 type=entry"/>
  </metadata:publication>
  <link href="http://host-ip-address/webservice/escenic/content/4" rel="edit"/>
  <link href="http://host-ip-address/webservice/escenic/content/4" rel="self"/>
  <link href="http://host-ip-address/webservice/escenic/lock/article/4" rel="http://
www.vizrt.com/types/relation/lock"/>
  <content type="application/vnd.vizrt.payload+xml">
    <vdf:payload xmlns:vdf="http://www.vizrt.com/types" model="http://host-ip-address/
webservice/escenic/model/news">
      <vdf:field name="TITLE"><vdf:value>My Updated Title</vdf:value></vdf:field>
      <vdf:field name="SUMMARY">
        <vdf:value>More text for Latin lovers</vdf:value>
      </vdf:field>
      <vdf:field name="BODY">
        <vdf:value>
          <div xmlns="http://www.w3.org/1999/xhtml">
            <p>Simple Body</p>
```

```
          </div>
        </vdf:value>
      </vdf:field>
    </vdf:payload>
  </content>
</entry>
```

If the **PUT** request succeeds, then the web service returns a **200 OK** response:

```
HTTP/1.1 100 Continue

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/atom+xml;type=entry
Content-Length: 0
Date: Sat, 04 Dec 2010 02:53:02 GMT
```

In most cases the fragment identifier used in lock requests corresponds directly to the name of a content item field. The fragment identifier **TITLE** used in the above example, however, is a special case. It corresponds to whatever content item field has been designated as the title in the publication's **content-type** resource. This field is never returned in an entry resource's VDF payload: its content is instead returned as the entry's **title** element. In order to update this field, therefore, the client must actually update the entry resource's **title** element as shown in the above example.

For information about how a field is designated as content item's title, see title-field.

5.14.4.1 Updating a Resource Without a Lock

A client can in fact attempt to **PUT** an updated resource without first acquiring a lock by simply omitting the **X-Escenic-Locks** header from the request. The web service then attempts to:

1. Acquire a resource lock on the client's behalf.
2. Perform the update.
3. Release the lock.

If it cannot do so because the resource (or a fragment of the resource) is already locked, then it returns a **409 CONFLICT** response.

Note that in this case the web service always treats the request as a resource lock request, even if a fragment lock would be sufficient to carry out the submitted update.

5.14.5 Delete Private Lock

Once the resource has been updated (or if the update is to be abandoned for some reason) the client can release the lock by submitting a **DELETE** request to the private lock URI:

```
curl --include -u username:password -X DELETE http://host-ip-address/webservice/
escenic/lock/article/4/private/9
```

If the **DELETE** request succeeds, then the web service returns a **204 No Content** response:

```
HTTP/1.1 204 No Content
Server: Apache-Coyote/1.1
Date: Sat, 04 Dec 2010 02:57:09 GMT
```

It is possible to force the release of a lock held by another client by submitting a **DELETE** request to the public lock URI:

```
curl --include -u user:password -X DELETE http://host-ip-address/webservice/escenic/
lock/article/4/public/39
```

If the **DELETE** request succeeds, then the web service returns a **204 No Content** response:

```
HTTP/1.1 204 No Content
Server: Apache-Coyote/1.1
Date: Sat, 04 Dec 2010 03:09:38 GMT
```

If a client fails to delete a private lock URI, the lock will be released by the web service after 24 hours. To keep a resource locked for longer than this, a client must release its lock and immediately acquire a new one.

Any attempt to update a resource using a deleted private lock URI will fail.

5.15 Delete a Content Item

To delete a content item, your client application must send an HTTP **DELETE** request to the same URI that would be used to retrieve it (see section 5.4). For example:

```
curl --include -u user:password -X DELETE \
> http://host-ip-address/webservice/escenic/content/43
HTTP/1.1 204 No Content
Server: Apache-Coyote/1.1
Date: Tue, 19 Feb 2013 09:12:00 GMT
```

If you're using **curl**, it's a good idea to specify **--include** with **DELETE** operations: **curl** will then output the response header returned from the web service as shown above, and you can verify whether or not the operation was successful:

- A response code in the 2*xx* range indicates success.
- A response code in the 4*xx* range means that you made an invalid edit and the server won't accept your modification.
- A response code in the 5*xx* range means there is a server error.

When you delete a content item, the object is not actually deleted from the database: its state is set to **deleted**, which means that it can be undeleted in the future if necessary.

5.16 Retrieve a List or Inbox

To retrieve links to a section's lists and/or inboxes, first retrieve the section as described in section 5.25, and then follow one of these link types:

- **http://www.vizrt.com/types/relation/lists** for a list feed, for example:

```
<link rel="http://www.vizrt.com/types/relation/lists"
      href="http://host-ip-address/webservice/escenic/section/22/lists"
      type="application/atom+xml"/>
```

- **`http://www.vizrt.com/types/relation/inboxes`** for an inbox feed, for example:

```
<link rel="http://www.vizrt.com/types/relation/inboxes"
      href="http://host-ip-address/webservice/escenic/section/22/inboxes"
      type="application/atom+xml"/>
```

To retrieve a list feed using the first of the above links, for example, you would submit the following request:

```
curl -u user:password -X GET http://host-ip-address/webservice/escenic/section/22/
lists
```

This will return an Atom feed containing entries for all the section's lists that looks something like this:

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <author>
    <name>Escenic Content Engine</name>
  </author>
  <id>http://host-ip-address/webservice/escenic/section/22/lists</id>
  <link href="http://host-ip-address/webservice/escenic/section/22/lists"
        rel="self" type="application/atom+xml" />
  <updated>2015-05-04T13:23:49.958Z</updated>
  <entry>
    <id>http://host-ip-address/webservice/content/com.escenic.list/423</id>
    <title type="text">Editor's Choice</title>
    <link href="http://host-ip-address/webservice/escenic/list-pool/423"
          rel="http://www.vizrt.com/types/relation/list-pool"
          type="application/atom+xml" />
    <link href="http://host-ip-address/webservice/escenic/changelog/list/423"
          rel="http://www.vizrt.com/types/relation/changelog"
          type="application/atom+xml" />
    <link href="http://host-ip-address/webservice/content/com.escenic.list/423"
          rel="edit-media" title="Editor's Choice"
          type="application/vnd.escenic.content+xml; type=com.escenic.list" />
    <link href="http://upload.wikimedia.org/wikipedia/commons/6/64/
Cebus_albifrons_edit.jpg"
          rel="monkey" title="Mr. Monkey" type="image/jpeg" />
  </entry>
  <entry>
    ...
  </entry>
  ...
</feed>
```

Following the section's **`http://www.vizrt.com/types/relation/inboxes`** link produces a very similar feed, containing entries for all the section's inboxes.

To retrieve a specific list, follow the appropriate entry's **`http://www.vizrt.com/types/relation/list-pool`** link. To retrieve the "Editor's Choice" list shown above, for example, you would submit the following request:

```
curl -u user:password -X GET http://host-ip-address/webservice/escenic/list-pool/423
```

This will return a list pool Atom feed, containing entries for all the items in the selected list:

```
<feed xmlns="http://www.w3.org/2005/Atom">
  <author>
    <name>Escenic Content Engine</name>
  </author>
```

```
  <id>http://host-ip-address/webservice/escenic/list-pool/423</id>
  <link rel="self"
        href="http://host-ip-address/webservice/escenic/list-pool/423"
        type="application/atom+xml"/>
  <updated>2015-04-22T12:52:11.311Z</updated>
  <link
    rel="http://www.vizrt.com/types/relation/changelog"
    href="http://host-ip-address/webservice/escenic/changelog/list/423"
    type="application/atom+xml"/>
  <entry>
    <id>http://host-ip-address/webservice/escenic/list-pool/handle/583347</id>
    <title type="text">(type=1 ,id=204697 ,pubId=-1)</title>
    <link
      rel="edit"
      href="http://host-ip-address/webservice/escenic/list-pool/handle/583347"
      type="application/atom+xml"/>
    <link
      rel="self"
      href="http://host-ip-address/webservice/escenic/list-pool/handle/583347"
      type="application/atom+xml"/>
    <link
      rel="about"
      href="http://host-ip-address/webservice/escenic/content/204697"
      type="application/atom+xml"/>
    <content type="application/vnd.vizrt.payload+xml">
      <vdf:payload
        xmlns:vdf="http://www.vizrt.com/types"
        model="http://host-ip-address/webservice/publication/dev.nightly/escenic/
model/com.escenic.poolentry">
      <vdf:field name="com.escenic.priority">
        <vdf:value>-1</vdf:value>
      </vdf:field>
      <vdf:field name="com.escenic.pinned">
        <vdf:value>false</vdf:value>
      </vdf:field>
    </vdf:payload>
    </content>
  </entry>
  <entry>
    ...
  </entry>
  ...
</feed>
```

Each entry in the feed references a list pool **handle**, which you can retrieve by following the entry's **`self`** or **`edit`** link. For example:

```
curl -u user:password -X GET http://host-ip-address/webservice/escenic/list-pool/
handle/583347
```

This returns a single Atom entry containing the handle for one list item:

```
<entry xmlns="http://www.w3.org/2005/Atom">
  <id>http://host-ip-address/webservice/escenic/list-pool/handle/583347</id>
  <title type="text">(type=1 ,id=204697 ,pubId=-1)</title>
  <link
    rel="edit"
    href="http://host-ip-address/webservice/escenic/list-pool/handle/583347"
    type="application/atom+xml"/>
```

```
  <link
    rel="self"
    href="http://host-ip-address/webservice/escenic/list-pool/handle/583347"
    type="application/atom+xml"/>
  <link
    rel="about"
    href="http://host-ip-address/webservice/escenic/content/204697"
    type="application/atom+xml"/>
  <content type="application/vnd.vizrt.payload+xml">
    <vdf:payload
      xmlns:vdf="http://www.vizrt.com/types"
      model="http://host-ip-address/webservice/publication/dev.nightly/escenic/model/
com.escenic.poolentry">
      <vdf:field name="com.escenic.priority">
        <vdf:value>-1</vdf:value>
      </vdf:field>
      <vdf:field name="com.escenic.pinned">
        <vdf:value>false</vdf:value>
      </vdf:field>
    </vdf:payload>
  </content>
  <link
    rel="collection"
    href="http://host-ip-address/webservice/escenic/list-pool/423"
    type="text/uri-list"/>
</entry>
```

From such a handle entry you can retrieve the content item the handle represents by following the **about** link, or return to the parent list/inbox by following the **collection** link.

The process of retrieving in-boxes and in-box entries is exactly the same as retrieving lists and list entries. Both structures are represented in the web service by list pool Atom feeds and handle entries.

The payload of a handle Atom entry always contains two fields, **com.escenic.priority** (default value **-1**) and **com.escenic.pinned** (default value **false**). These fields are present in both list and in-box entry handles, but they are only actually used in lists, for pinning items to specific positions (see Using A List Editor for a description of this functionality in Content Studio). An item is pinned to a specific position in a list by setting **com.escenic.priority** to the required position and setting **com.escenic.pinned** to **true**.

5.17 Edit a List or Inbox

Editing a list or inbox can involve the following operations:

- Add one or more items
- Insert/move one or more items to a new position in the list or inbox
- Pin/unpin an item to a position in the list (not supported for inboxes)
- Remove one or more items

5.17.1 Add Items

To add items to a list or inbox, you need to submit a **POST** request to the list or inbox with a **text/uri-list** document in the body of your request. The document must contain the web service

URIs of the content items you want to add to the list or inbox. To add two content items to a list, for example, you would need to first create a file containing your list of URIs, called **`uri-list.txt`** in this example:

```
http://host-ip-address/webservice/escenic/content/1093403
http://host-ip-address/webservice/escenic/content/1093556
```

And then submit it with the following post request:

```
curl --include -u user:password -X POST -H "Content-Type: text/uri-list" \
> http://host-ip-address/webservice/escenic/list-pool/listID --upload-file uri-
list.txt
HTTP/1.1 100 Continue

HTTP/1.1 201 Created
Server: Apache-Coyote/1.1
X-ECE-Active-Connections: 1
Location: http://host-ip-address/webservice/escenic/list-pool/15585
ETag: "2218338705"
Vary: Accept
Content-Type: application/atom+xml
Content-Length: 2566
Date: Mon, 04 Apr 2016 13:20:18 GMT

<?xml version="1.0"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <author>
    <name>Escenic Content Engine</name>
  </author>
  <id>http://host-ip-address/webservice/escenic/list-pool/15585</id>
  <link rel="self" href="http://host-ip-address/webservice/escenic/list-pool/15585"
 type="application/atom+xml"/>
  <updated>2016-04-04T12:56:07.124Z</updated>
  <title type="text">My List</title>
  <link rel="http://www.vizrt.com/types/relation/changelog" href="http://host-ip-address/
webservice/escenic/changelog/list/15585" type="application/atom+xml"/>
  <link rel="http://www.vizrt.com/types/relation/home-section" href="http://host-ip-
address/webservice/escenic/section/1" type="application/atom+xml" title="Home"/>
  <entry>
    <id>http://host-ip-address/webservice/escenic/list-pool/handle/16795</id>
    <updated>2016-04-04T12:56:07.125Z</updated>
    <title type="text">(type=1 ,id=1093403 ,pubId=-1)</title>
    <link rel="edit" href="http://host-ip-address/webservice/escenic/list-pool/
handle/16795" type="application/atom+xml"/>
    <link rel="self" href="http://host-ip-address/webservice/escenic/list-pool/
handle/16795" type="application/atom+xml"/>
    <link rel="about" href="http://host-ip-address/webservice/escenic/content/1093403"
 type="application/atom+xml"/>
    <content type="application/vnd.vizrt.payload+xml">
      <vdf:payload xmlns:vdf="http://www.vizrt.com/types" model="http://host-ip-address/
webservice/escenic/publication/demo/model/content-type/com.escenic.poolentry">
        <vdf:field name="com.escenic.priority">
          <vdf:value>-1</vdf:value>
        </vdf:field>
        <vdf:field name="com.escenic.pinned">
          <vdf:value>false</vdf:value>
        </vdf:field>
      </vdf:payload>
    </content>
```

```
  </entry>
  <entry>
    <id>http://host-ip-address/webservice/escenic/list-pool/handle/16796</id>
    <updated>2016-04-04T12:56:07.125Z</updated>
    <title type="text">(type=1 ,id=1093556 ,pubId=-1)</title>
    <link rel="edit" href="http://host-ip-address/webservice/escenic/list-pool/
handle/16796" type="application/atom+xml"/>
    <link rel="self" href="http://host-ip-address/webservice/escenic/list-pool/
handle/16796" type="application/atom+xml"/>
    <link rel="about" href="http://host-ip-address/webservice/escenic/content/1093556"
 type="application/atom+xml"/>
    <content type="application/vnd.vizrt.payload+xml">
      <vdf:payload xmlns:vdf="http://www.vizrt.com/types" model="http://host-ip-address/
webservice/escenic/publication/demo/model/content-type/com.escenic.poolentry">
        <vdf:field name="com.escenic.priority">
          <vdf:value>-1</vdf:value>
        </vdf:field>
        <vdf:field name="com.escenic.pinned">
          <vdf:value>false</vdf:value>
        </vdf:field>
      </vdf:payload>
    </content>
  </entry>
</feed>
```

The response is an Atom feed. The document contains links back to the two content items you have just added, and to the list pool handles that have been created for them.

If you now retrieve the list or in-box's list pool feed, you will see that the two new entries have been inserted at the top of the list. New entries are always added at the top by default. See section 5.17.2 for instructions on how to insert entries in any other position.

> Note that you can get the response in the form of an XHTML document rather than an Atom feed by including an accept header in your request:
>
> ```
> curl --include -u user:password -X POST -H "Content-Type: text/uri-list" -H
>  "Accept: application/xhtml+xml"\
> > http://host-ip-address/webservice/escenic/list-pool/listID --upload-file uri-
> list.txt
> ```

5.17.2 Insert/Move Items

Lists and in-boxes cannot contain duplicate items, so inserting items at a particular position and moving items to a new position are identical operations. You do it in exactly the same way as adding items to the top of a list/inbox, but instead of sending your **POST** request to the list pool URI, you must send it to the URI of one of the handles in the list pool:

```
curl --include -u user:password -X POST -H "Content-Type: text/uri-list" \
> http://host-ip-address/webservice/escenic/list-pool/handle/583347 --upload-file uri-
list.txt
HTTP/1.1 100 Continue

HTTP/1.1 201 Created
Server: Apache-Coyote/1.1
X-ECE-Active-Connections: 1
Location: http://host-ip-address/webservice/escenic/list-pool/handle/16774
ETag: "2003719864"
```

```
Vary: Accept
Content-Type: application/atom+xml
Content-Length: 2580
Date: Mon, 04 Apr 2016 13:22:40 GMT

<?xml version="1.0"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <author>
    <name>Escenic Content Engine</name>
  </author>
  <id>http://host-ip-address/webservice/escenic/list-pool/handle/16774</id>
  <link rel="self" href="http://host-ip-address/webservice/escenic/list-pool/handle/16774"
 type="application/atom+xml"/>
  <updated>2016-04-04T13:22:41.591Z</updated>
  <title type="text">My List</title>
  <link rel="http://www.vizrt.com/types/relation/changelog" href="http://host-ip-address/
webservice/escenic/changelog/list/15585" type="application/atom+xml"/>
  <link rel="http://www.vizrt.com/types/relation/home-section" href="http://host-ip-
address/webservice/escenic/section/1" type="application/atom+xml" title="Home"/>
  <entry>
    <id>http://host-ip-address/webservice/escenic/list-pool/handle/16799</id>
    <updated>2016-04-04T13:22:41.593Z</updated>
    <title type="text">(type=1 ,id=1093403 ,pubId=-1)</title>
    <link rel="edit" href="http://host-ip-address/webservice/escenic/list-pool/
handle/16799" type="application/atom+xml"/>
    <link rel="self" href="http://host-ip-address/webservice/escenic/list-pool/
handle/16799" type="application/atom+xml"/>
    <link rel="about" href="http://host-ip-address/webservice/escenic/content/1093403"
 type="application/atom+xml"/>
    <content type="application/vnd.vizrt.payload+xml">
      <vdf:payload xmlns:vdf="http://www.vizrt.com/types" model="http://host-ip-address/
webservice/escenic/publication/demo/model/content-type/com.escenic.poolentry">
        <vdf:field name="com.escenic.priority">
          <vdf:value>-1</vdf:value>
        </vdf:field>
        <vdf:field name="com.escenic.pinned">
          <vdf:value>false</vdf:value>
        </vdf:field>
      </vdf:payload>
    </content>
  </entry>
  <entry>
    <id>http://host-ip-address/webservice/escenic/list-pool/handle/16800</id>
    <updated>2016-04-04T13:22:41.594Z</updated>
    <title type="text">(type=1 ,id=1093556 ,pubId=-1)</title>
    <link rel="edit" href="http://host-ip-address/webservice/escenic/list-pool/
handle/16800" type="application/atom+xml"/>
    <link rel="self" href="http://host-ip-address/webservice/escenic/list-pool/
handle/16800" type="application/atom+xml"/>
    <link rel="about" href="http://host-ip-address/webservice/escenic/content/1093556"
 type="application/atom+xml"/>
    <content type="application/vnd.vizrt.payload+xml">
      <vdf:payload xmlns:vdf="http://www.vizrt.com/types" model="http://host-ip-address/
webservice/escenic/publication/demo/model/content-type/com.escenic.poolentry">
        <vdf:field name="com.escenic.priority">
          <vdf:value>-1</vdf:value>
        </vdf:field>
        <vdf:field name="com.escenic.pinned">
          <vdf:value>false</vdf:value>
```

```
        </vdf:field>
      </vdf:payload>
    </content>
  </entry>
</feed>
```

The response is exactly the same as if you had added the entries to the top of the list pool. In this case, however, they will have been inserted immediately after the handle to which you sent the **POST** request.

> Note that you can get the response in the form of an XHTML document rather than an Atom feed by including an accept header in your request:
>
> ```
> curl --include -u user:password -X POST -H "Content-Type: text/uri-list" -H
>  "Accept: application/xhtml+xml"\
> > http://host-ip-address/webservice/escenic/list-pool/listID --upload-file uri-
> list.txt
> ```

5.17.3 Pin/unpin an Item

To pin an item in a fixed position in a list, you must retrieve its handle as described in section 5.16, set the **com.escenic.priority** and **com.escenic.pinned** fields in its payload and send it to its **self** URI in a **POST** request. You must set **com.escenic.priority** to the required position, and **com.escenic.pinned** to **true**.

For example:

```
<entry xmlns="http://www.w3.org/2005/Atom">
  <id>http://host-ip-address/webservice/escenic/list-pool/handle/583347</id>
  <title type="text">(type=1 ,id=204697 ,pubId=-1)</title>
  <link
    rel="edit"
    href="http://host-ip-address/webservice/escenic/list-pool/handle/583347"
    type="application/atom+xml"/>
  <link
    rel="self"
    href="http://host-ip-address/webservice/escenic/list-pool/handle/583347"
    type="application/atom+xml"/>
  <link
    rel="about"
    href="http://host-ip-address/webservice/escenic/content/204697"
    type="application/atom+xml"/>
  <content type="application/vnd.vizrt.payload+xml">
    <vdf:payload
      xmlns:vdf="http://www.vizrt.com/types"
      model="http://host-ip-address/webservice/publication/dev.nightly/escenic/model/
com.escenic.poolentry">
      <vdf:field name="com.escenic.priority">
        <vdf:value>3</vdf:value>
      </vdf:field>
      <vdf:field name="com.escenic.pinned">
        <vdf:value>true</vdf:value>
      </vdf:field>
    </vdf:payload>
  </content>
  <link
    rel="collection"
    href="http://host-ip-address/webservice/escenic/list-pool/423"
```

```
    type="text/uri-list"/>
</entry>
```

Sending the above entry (called **`list-pool-item.xml`**) as follows:

```
curl --include -u user:password -X PUT -H "Content-Type: application/atom+xml;
 type=entry" \
> http://host-ip-address/webservice/escenic/list-pool/handle/583347 --upload-file
 list-pool-item.xml
HTTP/1.1 100 Continue

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
X-ECE-Active-Connections: 1
Content-Length: 0
Date: Fri, 25 Oct 2013 11:08:01 GMT
```

will pin the entry in position 3. To unpin it, simply repeat the process, this time ensuring that **`com.escenic.pinned`** is set to **`false`**. In this case, it does not matter what value the **`com.escenic.priority`** field contains; its contents will be ignored since the entry is being unpinned.

5.17.4 Remove an Item

To remove an item from a list or in-box, send an HTTP **`DELETE`** request to the entry's list pool handle. For example:

```
curl --include -u user:password -X DELETE \
> http://host-ip-address/webservice/escenic/list-pool/handle/584407
HTTP/1.1 200 OK
Date: Wed, 06 May 2015 06:53:36 GMT
Server: Apache-Coyote/1.1
X-ECE-Active-Connections: 1
Content-Length: 0
```

5.18 Navigate The Tag Hierarchy

The Content Engine's tag hierarchy, unlike all other content is not accessible from the main web service start URL. It has its own start URL, and in some senses can therefore be regarded as a separate web service. To access the tag hierarchy, a client program must submit a request to the following start URL:

```
curl -u user:password -X GET http://host-ip-address/webservice/escenic/classification
```

which returns an Atom feed like this:

```
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:thr="http://purl.org/syndication/
thread/1.0">
  <id>http://host-ip-address/webservice/escenic/classification</id>
  <title type="text">The entry point for the Escenic Classification Web Service</
title>
  <link href="http://host-ip-address/webservice/escenic/classification" rel="self"/>
  <link href="http://host-ip-address/webservice/open-search/tag-search-
description.xml" rel="search"/>
  <author>
```

```
    <name>Escenic Classification Web Service</name>
  </author>
  <updated>2011-03-11T05:57:18.540Z</updated>
  ...
  <entry>
    <id>tag:folksonomy.escenic.com,2002</id>
    <title type="text">Tags</title>
    <updated>2011-03-11T05:57:18.540Z</updated>
    <content type="application/vnd.vizrt.payload+xml">
      <vdf:payload
           model="http://host-ip-address/webservice/escenic/classification/model/tag"
           xmlns:vdf="http://www.vizrt.com/types">
        <vdf:field name="description">
          <vdf:value>A folksonomy for all types of tags</vdf:value>
        </vdf:field>
        <vdf:field name="aliases">
          <vdf:list>
            <vdf:value>AP</vdf:value>
          </vdf:list>
        </vdf:field>
      </vdf:payload>
    </content>
    <link href="http://host-ip-address/webservice/escenic/classification/
tag:folksonomy.escenic.com,2002" rel="self" title="Tags"/>
    <link href="http://host-ip-address/webservice/escenic/classification/
tag:folksonomy.escenic.com,2002" rel="down" title="Tags" thr:count="35"/>
  </entry>
  ...
</feed>
```

This feed contains an entry for each tag structure defined at the site. Each entry contains a single **down** link that returns a new feed containing entries for each top-level tag in the structure. So submitting the request:

```
curl -u user:password -X GET http://host-ip-address/webservice/escenic/classification/
tag:folksonomy.escenic.com,2002
```

returns a new Atom feed containing the top-level content of this tag structure.

```
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:thr="http://purl.org/syndication/
thread/1.0">
  <id>http://host-ip-address/webservice/escenic/classification/
tag:folksonomy.escenic.com,2002</id>
  <title type="text">Tags</title>
  <link href="http://host-ip-address/webservice/escenic/classification/
tag:folksonomy.escenic.com,2002" rel="self"/>
  <author>
    <name>Escenic Classification Web Service</name>
  </author>
  <updated>2011-03-14T11:16:00.687Z</updated>
  <link href="http://host-ip-address/webservice/escenic/changelog/classification/
tag:folksonomy@escenic.com,2002"
           rel="http://www.vizrt.com/types/relation/changelog"/>
  <opensearch:Url
     template="http://host-ip-address/webservice/escenic/classification/tag/search?
searchTerms={searchTerms}&parent=tag%3Afolksonomy%40escenic.com%2C2002"
     type="application/atom+xml" rel="parent"/>
  <entry>
    <id>tag:folksonomy.escenic.com,2002:db:1346</id>
```

```
    <title type="text">politics</title>
    <updated>2011-03-14T11:16:00.687Z</updated>
    ...
    <link
      href="http://host-ip-address/webservice/escenic/classification/tag/children/
tag:folksonomy.escenic.com,2002:db:1346"
      rel="down" title="politics" thr:count="10"/>
    ...
  </entry>
</feed>
```

A client can therefore use these links to traverse the entire tag hierarchy of a publication. Following the **down** link in the above example will return a feed containing another 10 entries representing the **politics** tag's children.

```
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:thr="http://purl.org/syndication/
thread/1.0">
  ...
  <entry>
    <id>tag:folksonomy.escenic.com,2002:ap</id>
    <title type="text">Arbeiderpartiet</title>
    <updated>2011-03-14T11:16:00.687Z</updated>
    <link
      href="http://host-ip-address/webservice/escenic/classification/tag/
tag:folksonomy.escenic.com,2002,2011:ap"
      rel="self"
      title="Arbeiderpartiet"/>
    <link
      href="http://host-ip-address/webservice/escenic/classification/tag/
tag:folksonomy.escenic.com,2002:db:1346"
      rel="http://www.vizrt.com/types/relation/parent"
      title="politics"/>
    <link
      href="http://host-ip-address/webservice/escenic/classification/
tag:folksonomy.escenic.com,2002"
      rel="http://www.vizrt.com/types/relation/top"
      title="Tags"/>
    <link
      href="http://host-ip-address/webservice/escenic/classification/tag/children/
tag:folksonomy.escenic.com,2002:ap"
      rel="down"
      title="Arbeiderpartiet" thr:count="25"/>
    <opensearch:Url
      template="http:/host-ip-address/webservice/search/{sectionPath}/{searchTerms}/?
pw={startPage?}&amp;c={count?}&amp;tag=tag%3Afolksonomy.escenic.com%2C2002%3Aap"
      type="application/atom+xml"/>
  </entry>
  ...
</feed>
```

Each tag entry contains an OpenSearch **Url** element. This element contains a search template that can be used to find content items tagged with this tag. See section 1.5 for instructions on how to use search templates.

Tag entries may also contain the following navigation links:

`http://www.vizrt.com/types/relation/parent`
Returns a feed containing an entry representing this tag's parent tag. Top level tag entries do not contain a link of this type.

`http://www.vizrt.com/types/relation/top`
Returns a feed containing an entry representing the tag structure to which this tag belongs.

5.18.1 Create a Tag

A client program can create a tag as follows:

1. Create an Atom entry that describes the new tag.

```
<entry xmlns="http://www.w3.org/2005/Atom">
  <id>tag:folksonomy.escenic.com,2002:tag-ap</id>
  <title type="text">Arbeiderpartiet</title>
  <content type="application/vnd.vizrt.payload+xml">
    <vdf:payload
         model="http://host-ip-address/webservice/escenic/classification/model/
tag"
         xmlns:vdf="http://www.vizrt.com/types">
      <vdf:field name="description">
        <vdf:value>Norwegian Labour Party</vdf:value>
      </vdf:field>
      <vdf:field name="aliases">
        <vdf:list>
          <vdf:value>AP</vdf:value>
        </vdf:list>
      </vdf:field>
    </vdf:payload>
  </content>
</entry>
```

The entry must/may contain the following elements:

`id` (optional)
The tag's identifier, which must have the form:

scheme-uri`:`*term*

where:

- *scheme-uri* is the scheme URI of the tag structure to which the tag is to be added.
- *term* is a local ID for the tag. It must be unique within the tag structure, and may contain any characters that are valid in a URI except for escaped slashes (**`%2F`**), which are not allowed.

If an **`id`** element is not supplied, then an auto-generated term (local ID) is assigned to the tag by the Content Engine.

`title`
The external, user-visible title of the tag.

`content` (optional)
A VDF payload document containing additional information about the tag. The VDF document may contain the following fields:

`description` (optional)
A description of the entity represented by the tag.

`aliases` (optional)
A list of zero or more aliases for the tag.

2. Save the Atom entry as a file.
3. **`POST`** the file either:
   - to a tag structure URL (if it is to be a top-level tag):

```
curl -u user:password \
> -X POST http://host-ip-address/webservice/escenic/classification/
tag:folksonomy.escenic.com,2002 \
> --header "Content-Type:application/atom+xml" \
> --upload-file newTag.xml
```

   - or to the URL of the required parent tag:

```
curl -u user:password \
> -X POST http://host-ip-address/webservice/escenic/classification/tag/
children/tag:folksonomy.escenic.com,2002:term-name \
> --header "Content-Type:application/atom+xml" \
> --upload-file newTag.xml
```

5.18.2 Move a Tag

A client program can move a tag to a different parent within its tag structure as follows:

1. Create an Atom entry containing the identifier of the tag.

```
<entry xmlns="http://www.w3.org/2005/Atom">
  <id>tag:folksonomy.escenic.com,2002:tag-ap</id>
</entry>
```

2. Save the Atom entry as a file.
3. **`POST`** the file either:
   - to a tag structure URL (if it is to be a top-level tag):

```
curl -u user:password \
> -X POST http://host-ip-address/webservice/escenic/classification/
tag:folksonomy.escenic.com,2002 \
> --header "Content-Type:application/atom+xml" \
> --upload-file newTag.xml
```

   - or to the URL of the required parent tag:

```
curl -u user:password \
> -X POST http://host-ip-address/webservice/escenic/classification/tag/
children/tag:folksonomy.escenic.com,2002:db:1346 \
> --header "Content-Type:application/atom+xml" \
> --upload-file newTag.xml
```

> The posted entry must contain an **`id`** element containing the correct ID of the tag to be moved.

5.18.3 Update a Tag

A client program can update a tag as follows:

1. **`GET`** an Atom entry describing the tag. For example:

```
curl -u user:password -X GET http://host-ip-address/webservice/escenic/
classification/tag/tag:folksonomy.escenic.com,2002:db:1346
```

returns an Atom entry containing the required description:

```
<entry xmlns="http://www.w3.org/2005/Atom">
  <id>tag:folksonomy.escenic.com,2002:db:1346</id>
  <title type="text">politics</title>
  <updated>2011-03-14T11:16:00.687Z</updated>
  <summary type="html">politics</summary>
  ...
</entry>
```

2. Save the Atom entry as a file.
3. Modify the Atom entry. Currently only the description of the tag may be changed.
4. **`PUT`** the file back to the same URI:

```
curl -u user:password \
> -X PUT http://host-ip-address/webservice/escenic/classification/tag/
tag:folksonomy.escenic.com,2002:db:1346 \
> --header "Content-Type:application/atom+xml" \
> --header "If-Match:*" \
> --upload-file updatedTag.xml
```

5.18.4 Add an Alias to a Tag

To add aliases to tags, follow the basic procedure described in section 5.18.3. Aliases are defined by adding values to the **`aliases`** field of a tag entry's **`content`** element. The following tag entry, for example:

```
<entry xmlns="http://www.w3.org/2005/Atom">
  <id>tag:folksonomy.escenic.com,2002:tag-ap</id>
  <title type="text">Arbeiderpartiet</title>
  <content type="application/vnd.vizrt.payload+xml">
    <vdf:payload
         model="http://host-ip-address/webservice/escenic/classification/model/tag"
         xmlns:vdf="http://www.vizrt.com/types">
      <vdf:field name="description">
        <vdf:value>Norwegian Labour Party</vdf:value>
      </vdf:field>
      <vdf:field name="aliases">
        <vdf:list>
          <vdf:value>AP</vdf:value>
          <vdf:value>Norwegian Labour Party</vdf:value>
        </vdf:list>
      </vdf:field>
    </vdf:payload>
  </content>
</entry>
```

defines a tag called **Arbeiderpartiet** with two aliases: **AP** and **Norwegian Labour Party**.

5.18.5 Delete a Tag

A client program can delete a tag by sending an **`HTTP DELETE`** request to the tag's URI. For example

```
curl -u user:password \
> -X DELETE http://host-ip-address/webservice/escenic/classification/tag/
tag:folksonomy.escenic.com,2002:tag-ap
```

Note that deleting a tag with children also deletes the children.

5.18.6 Search For a Tag

The feed returned by the root tag URL (**`http://`***host-ip-address*`/webservice/escenic/classification`**) includes a **`search`** link:

The atom feed return by the web service is given below:

```
<feed xmlns="http://www.w3.org/2005/Atom">
  ...
  <link href="http://host-ip-address/webservice/open-search/tag-search-
description.xml" rel="search"/>
  ...
</feed>
```

If the client application follows this link:

```
curl -u user:password -X GET http://host-ip-address/webservice/open-search/tag-search-
description.xml
```

The web service returns an OpenSearch document describing the URL format required to search for tags:

```
<?xml version="1.0" encoding="UTF-8"?>
<OpenSearchDescription xmlns="http://a9.com/-/spec/opensearch/1.1/">
  <ShortName>Tag Search</ShortName>
  <Description>Search for a tags</Description>
  <Url xmlns:tag="http://xmlns.escenic.com/2011/classification-tags"
       type="application/atom+xml"
       template="http://host-ip-address/webservice/escenic/classification/tag/search?
searchTerms={searchTerms}&amp;
                 tagStructures={tag:tag-
schemes?}&amp;startPage={startPage?}&amp;pageSize={count?}"/>
  <LongName/>
  <Developer/>
  <Attribution/>
  <SyndicationRight/>
  <AdultContent>false</AdultContent>
  <OutputEncoding>UTF-8</OutputEncoding>
  <InputEncoding>UTF-8</InputEncoding>
</OpenSearchDescription>
```

From this information the client can construct a URL that submits a query. All of the search parameters except **`searchTerms`** are optional. If the **`tagStructures`** parameter is omitted then all tag structures are searched for the specified search terms. It is tag **titles** that are searched, not tag **terms**, and "starts with", case-insensitive matching is used.

For example:

```
curl -u user:password http://host-ip-address/webservice/escenic/classification/tag/
search?searchTerms=arbeid
```

will find the tag "Arbeiderpartiet":

```
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:thr="http://purl.org/syndication/
thread/1.0">
  <id>http://host-ip-address/webservice/escenic/classification/tag/search</id>
```

```
  <title type="html">Search for &lt;em&gt;&lt;strong&gt;arbeid&lt;/strong&gt;&lt;/
em&gt;</title>
  <link href="http://host-ip-address/webservice/escenic/classification/tag/search"
 rel="self"/>
  <author>
    <name>Escenic Classification Web Service - Search Tags</name>
  </author>
  <updated>2011-03-15T13:41:48.515Z</updated>
  <entry>
    <id>tag:folksonomy.escenic.com,2002:ap</id>
    <title type="text">Arbeiderpartiet</title>
    <updated>2011-03-15T10:17:59.919Z</updated>
    <content type="html">politics / &lt;em&gt;&lt;strong&gt;Arbeid&lt;/strong&gt;&lt;/
em&gt;erpartiet</content>
    <content type="application/vnd.vizrt.payload+xml">
      <vdf:payload
           model="http://host-ip-address/webservice/escenic/classification/model/tag"
           xmlns:vdf="http://www.vizrt.com/types">
        <vdf:field name="description">
          <vdf:value>Norwegian Labour Party</vdf:value>
        </vdf:field>
        <vdf:field name="aliases">
          <vdf:list/>
        </vdf:field>
      </vdf:payload>
    </content>
    <link
      href="http://host-ip-address/webservice/escenic/classification/tag/
tag:folksonomy.escenic.com,2002:ap"
      rel="self"
      title="Arbeiderpartiet"/>
    <link
      href="http://host-ip-address/webservice/escenic/classification/tag/
tag:folksonomy.escenic.com,2002:db:1346"
      rel="http://www.vizrt.com/types/relation/parent"
      title="politics"/>
    <link
      href="http://host-ip-address/webservice/escenic/classification/
tag:folksonomy.escenic.com,2002"
      rel="http://www.vizrt.com/types/relation/top"
      title="Tags"/>
    <link
      href="http://host-ip-address/webservice/escenic/classification/tag/children/
tag:folksonomy.escenic.com,2002:ap"
      rel="down"
      title="Arbeiderpartiet" thr:count="0"/>
  </entry>
</feed>
```

It is possible to limit the search to a specific tag structure by specifying the **`tagStructures`** parameter.

5.18.7 Merge tags

A client program can merge two tags as follows:

1. Create a **`Payload`** containing the identifier of both tags

   ```
   <?xml version="1.0" encoding="ISO-8859-1"?>
   ```

```
<payload xmlns="http://www.vizrt.com/types">
  <field name="merge">
    <list>
      <payload>
        <field name="tag">
          <origin href="tag:folksonomy@escenic.com,2012:tag-to-merge" />
        </field>
      </payload>
    </list>
  </field>
 <field name="with">
    <origin href="tag:folksonomy@escenic.com,2012:destination-tag" />
 </field>
</payload>
```

The XML above will merge **`tag:folksonomy@escenic.com,2012:tag-to-merge`** into **`tag:folksonomy@escenic.com,2012:destination-tag`**

> You can merge more than one tag at once by adding more payloads to the **`<list>`** element

2. Save the Payload as a file.
3. POST the file to the **`merge`** resource

```
curl -u user:password
              > -X POST http://host-ip-address/webservice/escenic/classification/
merge \
              > --header "Content-Type:application/vnd.vizrt.payload+xml" \
              > --header "If-Match:*" \
              > --upload-file mergeTags.xml
```

5.19 Search for Persons

The web service's "start" feed resource (**`http://`***host-ip-address***`/webservice/escenic/section/ROOT/subsections`**, see section 5.1) contains a link of type **`http://www.vizrt.com/types/relation/person`**. This is the only start-point for person lookup. If the client application follow this link:

```
curl -u user:password http://host-ip-address/webservice/escenic/person
```

then an empty Atom feed (that is, a feed containing no entries) is returned. This empty feed, however, does contain a link of type **`http://www.vizrt.com/types/relation/person-lookup`**:

```
<feed xmlns="http://www.w3.org/2005/Atom">
  ...
  <link rel="http://www.vizrt.com/types/relation/person-lookup"
        href="http://host-ip-address/webservice/open-search/person-lookup-
description.xml"
        type="application/opensearchdescription+xml"/>
  ...
</feed>
```

Following this link in turn:

```
curl -u user:password http://host-ip-address/webservice/open-search/person-lookup-
description.xml
```

returns an OpenSearch document describing the describing the URL format required to search for persons:

```
<OpenSearchDescription xmlns="http://a9.com/-/spec/opensearch/1.1/">
  <ShortName>Person Lookup</ShortName>
  <Description>Lookup for persons</Description>
  <Url type="application/atom+xml" template="http://host-ip-address/webservice/
escenic/person/lookup?email={emailAddress}"/>
  <LongName/>
  <Developer/>
  <Attribution/>
  <SyndicationRight/>
  <AdultContent>false</AdultContent>
  <OutputEncoding>UTF-8</OutputEncoding>
  <InputEncoding>UTF-8</InputEncoding>
</OpenSearchDescription>
```

From this information the client can construct a URL that submits a person look-up request. Note that:

- Unlike content search, only one URL template is provided here, and it returns results in the form of an Atom feed.
- The only search criterion that can be used for person look-up is email address.

For example:

```
curl -u user:password http://host-ip-address/webservice/escenic/person/lookup?
email=mail@example.com
```

This returns an Atom feed containing entries for all persons with the email address **mail@example.com**.

5.20 Retrieve a Person

The result returned from a person search (see section 5.19) is an Atom feed in which each entry contains a complete person record:

```
<?xml version="1.0"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <title type="text">Person lookup for &lt;em&gt;&lt;strong&gt;mail@example.com&lt;/
strong&gt;&lt;/em&gt;</title>
  <author>
    <name>Escenic Content Engine</name>
  </author>
  <updated>2012-03-22T06:31:53.054Z</updated>
  <id>3d8d6fa0-8e35-4630-b02e-7865eace3441</id>
  <link rel="self" href="http://host-ip-address/webservice/escenic/person/lookup?
email=mail@example.com" type="application/atom+xml"/>
  <entry xmlns:app="http://www.w3.org/2007/app" xmlns:metadata="http://
xmlns.escenic.com/2010/atom-metadata" xmlns:dcterms="http://purl.org/dc/terms/">
    <id>http://host-ip-address/webservice/escenic/person/20</id>
    <link rel="self" href="http://host-ip-address/webservice/escenic/person/20"
 type="application/atom+xml; type=entry"/>
    <link rel="edit" href="http://host-ip-address/webservice/escenic/person/20"
 type="application/atom+xml; type=entry"/>
```

```
    <link rel="http://www.vizrt.com/types/relation/lock" href="http://host-ip-address/
webservice/escenic/lock/person/20" type="application/atom+xml; type=entry"/>
    <dcterms:identifier>20</dcterms:identifier>
    <link href="http://host-ip-address/webservice/escenic/publication/publication-id/"
 rel="http://www.vizrt.com/types/relation/publication" type="application/atom+xml;
 type=entry" title="dev.nightly"/>
    <metadata:publication href="http://host-ip-address/webservice/escenic/
publication/publication-id/" title="publication-title"/>
    <content type="application/vnd.vizrt.payload+xml">
      <vdf:payload xmlns:vdf="http://www.vizrt.com/types" model="http://host-ip-
address/webservice/publication/test/escenic/model/com.escenic.person">
        <vdf:field name="com.escenic.username"/>
        <vdf:field name="com.escenic.firstName">
          <vdf:value>Firstname</vdf:value>
        </vdf:field>
        <vdf:field name="com.escenic.surName">
          <vdf:value>Surname</vdf:value>
        </vdf:field>
        <vdf:field name="com.escenic.occupation"/>
        <vdf:field name="com.escenic.address"/>
        <vdf:field name="com.escenic.postalcode"/>
        <vdf:field name="com.escenic.city"/>
        <vdf:field name="com.escenic.phonework"/>
        <vdf:field name="com.escenic.phonemobile"/>
        <vdf:field name="com.escenic.phoneprivate"/>
        <vdf:field name="com.escenic.emailaddress">
          <vdf:value>mail@example.com</vdf:value>
        </vdf:field>
      </vdf:payload>
    </content>
    <app:edited>2012-03-22T04:17:44.000Z</app:edited>
    <updated>2012-03-22T04:17:44.000Z</updated>
    <published>2012-03-22T04:17:44.000Z</published>
    <dcterms:created>2012-03-22T04:17:44.000Z</dcterms:created>
    <title type="text">Surname</title>
    <summary type="text">Firstname Surname</summary>
  </entry>
</feed>
```

Therefore, you probably don't need to explicitly retrieve person objects. You can do so if you wish, by following the **`self`** or **`edit`** link in each of the feed's entries, for example:

```
curl -u user:password http://host-ip-address/webservice/escenic/person/20
```

However, all you get back is the same information as a single Atom entry:

```
<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:app="http://www.w3.org/2007/app"
 xmlns:metadata="http://xmlns.escenic.com/2010/atom-metadata" xmlns:dcterms="http://
purl.org/dc/terms/">
  <id>http://host-ip-address/webservice/escenic/person/20</id>
  <link rel="self" href="http://host-ip-address/webservice/escenic/person/20"
 type="application/atom+xml; type=entry"/>
  <link rel="edit" href="http://host-ip-address/webservice/escenic/person/20"
 type="application/atom+xml; type=entry"/>
  <link rel="http://www.vizrt.com/types/relation/lock" href="http://host-ip-address/
webservice/escenic/lock/person/20" type="application/atom+xml; type=entry"/>
  <dcterms:identifier>20</dcterms:identifier>
```

```
  <link href="http://host-ip-address/webservice/escenic/publication/publication-id/"
 rel="http://www.vizrt.com/types/relation/publication" type="application/atom+xml;
 type=entry" title="dev.nightly"/>
  <metadata:publication href="http://host-ip-address/webservice/escenic/
publication/publication-id/" title="publication-title"/>
  <content type="application/vnd.vizrt.payload+xml">
    <vdf:payload xmlns:vdf="http://www.vizrt.com/types" model="http://host-ip-address/
webservice/publication/test/escenic/model/com.escenic.person">
      <vdf:field name="com.escenic.username"/>
      <vdf:field name="com.escenic.firstName">
        <vdf:value>Firstname</vdf:value>
      </vdf:field>
      <vdf:field name="com.escenic.surName">
        <vdf:value>Surname</vdf:value>
      </vdf:field>
      <vdf:field name="com.escenic.occupation"/>
      <vdf:field name="com.escenic.address"/>
      <vdf:field name="com.escenic.postalcode"/>
      <vdf:field name="com.escenic.city"/>
      <vdf:field name="com.escenic.phonework"/>
      <vdf:field name="com.escenic.phonemobile"/>
      <vdf:field name="com.escenic.phoneprivate"/>
      <vdf:field name="com.escenic.emailaddress">
        <vdf:value>mail@example.com</vdf:value>
      </vdf:field>
    </vdf:payload>
  </content>
  <app:edited>2012-03-22T04:17:44.000Z</app:edited>
  <updated>2012-03-22T04:17:44.000Z</updated>
  <published>2012-03-22T04:17:44.000Z</published>
  <dcterms:created>2012-03-22T04:17:44.000Z</dcterms:created>
  <title type="text">Surname</title>
  <summary type="text">Firstname Surname</summary>
</entry>
```

5.21 Process a Person

Like content items, retrieved person records are embedded in Atom entry resources as Viz Data Format (VDF) payload documents:

```
<vdf:payload xmlns:vdf="http://www.vizrt.com/types"
             model="http://host-ip-address/webservice/publication/test/escenic/model/
com.escenic.person">
  <vdf:field name="com.escenic.username"/>
  <vdf:field name="com.escenic.firstName">
    <vdf:value>Firstname</vdf:value>
  </vdf:field>
  <vdf:field name="com.escenic.surName">
    <vdf:value>Surname</vdf:value>
  </vdf:field>
  <vdf:field name="com.escenic.occupation"/>
  <vdf:field name="com.escenic.address"/>
  <vdf:field name="com.escenic.postalcode"/>
  <vdf:field name="com.escenic.city"/>
  <vdf:field name="com.escenic.phonework"/>
  <vdf:field name="com.escenic.phonemobile"/>
  <vdf:field name="com.escenic.phoneprivate"/>
```

```
  <vdf:field name="com.escenic.emailaddress">
    <vdf:value>mail@example.com</vdf:value>
  </vdf:field>
</vdf:payload>
```

AS for content items, the root element of the payload document has a **model** attribute that contains a reference to a VDF model document. This document contains a schema defining the structure and content of the payload document. That is, it tells you what fields the payload document may contain, how they are organized and what data types they contain. Your client can therefore download this model document and use it as a guide to processing the contents of the payload document.

5.22 Change a Person

Once your client application has retrieved a person, it can modify it and submit the changed version. If you are following this example using curl from the command line, you can simply copy the returned entry into a text editor and manually change one of the values. For example:

```
<vdf:field name="com.escenic.city">Oslo</vdf:field>
```

and save the results to a file (called **my-edited-person.xml**, for example).

To commit your change you have to submit a **PUT** request to the same URI that you would use to retrieve it (that is, the URI supplied in the search results entry **edit** or **self** link).

```
curl --include -u user:password -X PUT -H "If-Match: *" -H "Content-Type: application/
atom+xml" \
> http://host-ip-address/webservice/escenic/person/20 --upload-file my-edited-
person.xml
HTTP/1.1 100 Continue

HTTP/1.1 204 No Content
Server: Apache-Coyote/1.1
Date: Tue, 20 Mar 2012 05:50:26 GMT
```

In order for the **PUT** operation to work, you must specify two HTTP headers as shown above:

Content-Type: application/atom+xml
: You must specify the content type of the data you are uploading.

If-Match: *
: The **If-Match** header value ***** is used here for reasons of simplicity. It is acceptable to use it for test and demonstration purposes, but should **never** be used in a production system. For more information about this header and what it does, see section 6.2.

If you're using **curl**, it's a good idea to specify **--include** with **PUT** operations: **curl** will then output the response header returned from the web service as shown above, and you can verify whether or not the operation was successful:

- A response code in the 2*xx* range indicates success.
- A response code in the 4*xx* range means that you made an invalid edit and the server won't accept your modification.
- A response code in the 5*xx* range means there is a server error.

Submitting a change in this way may not work if the resource you are attempting to modify is already locked by another client application. For more information about this and about how locking works, see section 5.14.

5.23 Create a Person

The procedure for creating a new person is more or less identical to creating a new content item. Your client application must create an Atom entry resource containing a VDF payload document:

```
<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:app="http://www.w3.org/2007/app"
 xmlns:metadata="http://xmlns.escenic.com/2010/atom-metadata" xmlns:dcterms="http://
purl.org/dc/terms/">
  <content type="application/vnd.vizrt.payload+xml">
    <vdf:payload xmlns:vdf="http://www.vizrt.com/types" model="http://host-ip-address/
webservice/publication/test/escenic/model/com.escenic.person">
      <vdf:field name="com.escenic.username"/>
      <vdf:field name="com.escenic.firstName">
        <vdf:value>Fred</vdf:value>
      </vdf:field>
      <vdf:field name="com.escenic.surName">
        <vdf:value>Flintstone</vdf:value>
      </vdf:field>
      <vdf:field name="com.escenic.occupation"/>
      <vdf:field name="com.escenic.address"/>
      <vdf:field name="com.escenic.postalcode"/>
      <vdf:field name="com.escenic.city"/>
      <vdf:field name="com.escenic.phonework"/>
      <vdf:field name="com.escenic.phonemobile"/>
      <vdf:field name="com.escenic.phoneprivate"/>
      <vdf:field name="com.escenic.emailaddress"/>
    </vdf:payload>
  </content>
  <title type="text">Flintstone</title>
  <summary type="text">Fred Flintstone</summary>
  <metadata:publication href="http://host-ip-address/webservice/escenic/publication/
test/" title="test"/>
</entry>
```

Your payload document must include a **`com.escenic.surname`** field, and the field must contain a value, otherwise no person record can be created.

This document must then be saved in a file (**`my-new-person.xml`**, for example) and **`POST`**ed to the URI of the person collection. This URI is supplied as a link in the web service's "start" feed resource and has the link type **`http://www.vizrt.com/types/relation/person`** (see section 3.2.14).

```
curl --include -u user:password -X POST -H "Content-Type: application/atom+xml" \
> http://host-ip-address/webservice/escenic/person --upload-file my-new-person.xml
HTTP/1.1 100 Continue

HTTP/1.1 201 Created
Server: Apache-Coyote/1.1
Location: http://localhost:8080/webservice/escenic/person/35
Content-Type: application/atom+xml; type=entry
Content-Length: 0
Date: Tue, 20 Mar 2012 06:59:38 GMT
```

In order for the **POST** operation to work, you must specify a HTTP header as shown above.

If you're using **curl**, it's a good idea to specify **--include** with **POST** operations: **curl** will then output the response header returned from the web service as shown above, and you can verify whether or not the operation was successful:

- A response code in the 2*xx* range indicates success.
- A response code in the 4*xx* range means that you made an invalid addition and the server won't accept your new content item.
- A response code in the 5*xx* range means there is a server error.

The **Location** response specifies the location of the newly-created person record: you can retrieve the person record by submitting a **GET** request to this URL.

> A quick way of finding out how to create a correctly structured person in VDF format is to **GET** a person and copy the structure.

5.24 Delete a Person

To delete a person, your client application must send an HTTP **DELETE** request to the same URI that would be used to retrieve it (see section 5.20). For example:

```
curl --include -u user:password -X DELETE \
> http://host-ip-address/webservice/escenic/person/20
HTTP/1.1 204 No Content
Server: Apache-Coyote/1.1
Date: Tue, 20 Mar 2012 09:12:00 GMT
```

If you're using **curl**, it's a good idea to specify **--include** with **DELETE** operations: **curl** will then output the response header returned from the web service as shown above, and you can verify whether or not the operation was successful:

- A response code in the 2*xx* range indicates success.
- A response code in the 4*xx* range means that you made an invalid edit and the server won't accept your modification.
- A response code in the 5*xx* range means there is a server error.

> When you delete a person, the object is actually deleted from the database (unlike a content item, which is just transitioned to the state **deleted**.

5.25 Retrieve a Section

The Atom feeds returned when navigating a publication's section tree, as described in (section 5.2), contain entries for all the subsections of a section. In addition to navigation links, each of these entries has a **content** element that contains the actual section definition. For example:

```
<?xml version="1.0"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  ...[elements omitted]...
  <title type="text">Subsections for Section with id=1</title>
```

```
  <entry xmlns:app="http://www.w3.org/2007/app"
         xmlns:metadata="http://xmlns.escenic.com/2010/atom-metadata"
 xmlns:dcterms="http://purl.org/dc/terms/">
    <id>http://host-ip-address/webservice/escenic/section/2</id>
    <link rel="self" href="http://host-ip-address/webservice/escenic/section/2"
 type="application/atom+xml; type=entry"/>
    <link rel="edit" href="http://host-ip-address/webservice/escenic/section/2"
 type="application/atom+xml; type=entry"/>
    <link rel="http://www.vizrt.com/types/relation/lock" href="http://host-ip-address/
webservice/escenic/lock/section/2"
          type="application/atom+xml; type=entry"/>
    <dcterms:identifier>2</dcterms:identifier>
    <link href="http://host-ip-address/webservice/escenic/publication/publication-
id/"
          rel="http://www.vizrt.com/types/relation/publication" type="application/atom
+xml; type=entry" title="demo"/>
    <metadata:publication href="http://host-ip-address/webservice/escenic/
publication/publication-id/" title="publication-title"/>
    <content type="application/vnd.vizrt.payload+xml">
      <vdf:payload xmlns:vdf="http://www.vizrt.com/types"
                   model="http://host-ip-address/webservice/publication/publication-
id/escenic/model/com.escenic.section">
        <vdf:field name="com.escenic.sectionName">
          <vdf:value>New Articles</vdf:value>
        </vdf:field>
        <vdf:field name="com.escenic.uniqueName">
          <vdf:value>ece_incoming</vdf:value>
        </vdf:field>
        <vdf:field name="com.escenic.directoryName">
          <vdf:value>incoming</vdf:value>
        </vdf:field>
        <vdf:field name="com.escenic.sectionURI"/>
        <vdf:field name="com.escenic.inheritAccess">
          <vdf:value>true</vdf:value>
        </vdf:field>
        <vdf:field name="com.escenic.virtualSource">
          <vdf:value>false</vdf:value>
        </vdf:field>
        <vdf:field name="com.escenic.hidden">
          <vdf:value>false</vdf:value>
        </vdf:field>
        <vdf:field name="com.escenic.agreementRequired">
          <vdf:value>false</vdf:value>
        </vdf:field>
        <vdf:field name="com.escenic.agreementInfo"/>
        <vdf:field name="com.escenic.agreementType"/>
        <vdf:field name="com.escenic.sectionLayout">
          <vdf:value>defaultsection</vdf:value>
        </vdf:field>
        <vdf:field name="com.escenic.articleLayout">
          <vdf:value>defaultarticle</vdf:value>
        </vdf:field>
        <vdf:field name="com.escenic.sectionParameters">
          <vdf:value/>
        </vdf:field>
      </vdf:payload>
    </content>
    ...[elements omitted]...
```

```
  </entry>
  ...[other entry elements omitted]...
</feed>
```

Therefore, you probably don't need to explicitly retrieve section objects. You can do so if you wish, by following the **`self`** or **`edit`** link in each of the feed's entries (highlighted above). For example:

```
curl -u user:password http://host-ip-address/webservice/escenic/section/3
```

However, all you will get by doing so is exactly the same content packaged as a single Atom entry:

```
<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:app="http://www.w3.org/2007/app"
 xmlns:metadata="http://xmlns.escenic.com/2010/atom-metadata" xmlns:dcterms="http://
purl.org/dc/terms/">
  <id>http://host-ip-address/webservice/escenic/section/3</id>
  <link rel="self" href="http://host-ip-address/webservice/escenic/section/3"
 type="application/atom+xml; type=entry"/>
  <link rel="edit" href="http://host-ip-address/webservice/escenic/section/3"
 type="application/atom+xml; type=entry"/>
  <link rel="http://www.vizrt.com/types/relation/lock" href="http://host-ip-address/
webservice/escenic/lock/section/3" type="application/atom+xml; type=entry"/>
  <dcterms:identifier>3</dcterms:identifier>
  <link href="http://host-ip-address/webservice/escenic/publication/publication-id/"
 rel="http://www.vizrt.com/types/relation/publication" type="application/atom+xml;
 type=entry" title="demo"/>
  <metadata:publication href="http://host-ip-address/webservice/escenic/
publication/publication-id/" title="publication-title"/>
  <content type="application/vnd.vizrt.payload+xml">
    <vdf:payload xmlns:vdf="http://www.vizrt.com/types" model="http://host-ip-address/
webservice/publication/publication-id/escenic/model/com.escenic.section">
      <vdf:field name="com.escenic.sectionName">
        <vdf:value>Examples</vdf:value>
      </vdf:field>
      <vdf:field name="com.escenic.uniqueName">
        <vdf:value>ece_examples</vdf:value>
      </vdf:field>
      <vdf:field name="com.escenic.directoryName">
        <vdf:value>Examples</vdf:value>
      </vdf:field>
      <vdf:field name="com.escenic.sectionURI"/>
      <vdf:field name="com.escenic.inheritAccess">
        <vdf:value>true</vdf:value>
      </vdf:field>
      <vdf:field name="com.escenic.virtualSource">
        <vdf:value>false</vdf:value>
      </vdf:field>
      <vdf:field name="com.escenic.hidden">
        <vdf:value>false</vdf:value>
      </vdf:field>
      <vdf:field name="com.escenic.agreementRequired">
        <vdf:value>false</vdf:value>
      </vdf:field>
      <vdf:field name="com.escenic.agreementInfo"/>
      <vdf:field name="com.escenic.agreementType"/>
      <vdf:field name="com.escenic.sectionLayout">
        <vdf:value>defaultsection</vdf:value>
      </vdf:field>
      <vdf:field name="com.escenic.articleLayout">
```

```
        <vdf:value>defaultarticle</vdf:value>
      </vdf:field>
      <vdf:field name="com.escenic.sectionParameters">
        <vdf:value/>
      </vdf:field>
    </vdf:payload>
  </content>
  ...[elements omitted]...
</entry>
```

5.26 Process a Section

Like content items, retrieved section records are embedded in Atom entry resources as Viz Data Format (VDF) payload documents:

```
<vdf:payload xmlns:vdf="http://www.vizrt.com/types"
             model="http://host-ip-address/webservice/publication/publication-id/
escenic/model/com.escenic.section">
  <vdf:field name="com.escenic.sectionName">
    <vdf:value>Examples</vdf:value>
  </vdf:field>
  <vdf:field name="com.escenic.uniqueName">
    <vdf:value>ece_examples</vdf:value>
  </vdf:field>
  ...[elements omitted]...
</vdf:payload>
```

As for content items, the root element of the payload document has a **`model`** attribute that contains a reference to a VDF model document. This document contains a schema defining the structure and content of the payload document. That is, it tells you what fields the payload document may contain, how they are organized and what data types they contain. Your client can therefore download this model document and use it as a guide to processing the contents of the payload document.

5.27 Change a Section

Once your client application has retrieved a section, it can modify it and submit the changed version. If you are following this example using curl from the command line, you can simply copy the returned entry into a text editor and manually change one of the values. For example:

```
<vdf:field name="com.escenic.sectionName">
  <vdf:value>My Section</vdf:value>
</vdf:field>
```

and save the results to a file (called **`my-edited-section.xml`**, for example).

To commit your change you have to submit a **`PUT`** request to the same URI that you would use to retrieve it (that is, the URI supplied in the entry's **`edit`** or **`self`** link).

```
curl --include -u user:password -X PUT -H "If-Match: *" -H "Content-Type: application/
atom+xml" \
> http://host-ip-address/webservice/escenic/section/3 --upload-file my-edited-
section.xml
HTTP/1.1 100 Continue
```

```
HTTP/1.1 204 No Content
Server: Apache-Coyote/1.1
Date: Tue, 20 Mar 2012 05:50:26 GMT
```

In order for the **PUT** operation to work, you must specify two HTTP headers as shown above:

Content-Type: application/atom+xml
: You must specify the content type of the data you are uploading.

If-Match: *
: The **If-Match** header value ***** is used here for reasons of simplicity. It is acceptable to use it for test and demonstration purposes, but should **never** be used in a production system. For more information about this header and what it does, see section 6.2.

If you're using **curl**, it's a good idea to specify **--include** with **PUT** operations: **curl** will then output the response header returned from the web service as shown above, and you can verify whether or not the operation was successful:

- A response code in the 2*xx* range indicates success.
- A response code in the 4*xx* range means that you made an invalid edit and the server won't accept your modification.
- A response code in the 5*xx* range means there is a server error.

> Submitting a change in this way may not work if the resource you are attempting to modify is already locked by another client application. For more information about this and about how locking works, see section 5.14.

5.27.1 Changing/Adding Section Parameters

Changing or adding section parameters is one of the most common operations you might want to perform on a section. Section parameters are stored in a field called **com.escenic.sectionParameters**.

The parameters you specify in a **com.escenic.sectionParameters** field must be correctly formatted. Each parameter must have the form

```
name = value
```

and must appear on a separate line. For more detailed information about what is allowed, check the Javadoc for the class **java.util.Properties**. Here is an example **com.escenic.sectionParameters** field containing the parameter settings **foo=bar** and **bar=foo**:

```
<vdf:field name="com.escenic.sectionParameters">
  <vdf:value>
foo=bar
bar=foo
  </vdf:value>
</vdf:field>
```

5.28 Create a Section

The procedure for creating a new section is more or less identical to creating a new content item. Your client application must create an Atom entry resource containing a VDF payload document:

```
<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:app="http://www.w3.org/2007/app"
 xmlns:metadata="http://xmlns.escenic.com/2010/atom-metadata" xmlns:dcterms="http://
purl.org/dc/terms/">
  <link rel="http://www.vizrt.com/types/relation/parent" href="http://host-ip-address/
webservice/escenic/section/1"
        title="Home" type="application/atom+xml; type=entry"></link>
  <content type="application/vnd.vizrt.payload+xml">
    <vdf:payload xmlns:vdf="http://www.vizrt.com/types"
                 model="http://host-ip-address/webservice/publication/publication-id/
escenic/model/com.escenic.section">
      <vdf:field name="com.escenic.sectionName">
        <vdf:value>New Section</vdf:value>
      </vdf:field>
      <vdf:field name="com.escenic.uniqueName">
        <vdf:value>new_section</vdf:value>
      </vdf:field>
      <vdf:field name="com.escenic.directoryName">
        <vdf:value>new_section</vdf:value>
      </vdf:field>
    </vdf:payload>
  </content>
</entry>
```

This document must then be saved in a file (**`my-new-section.xml`**, for example) and **`POST`**ed to the URI of the section collection. This URI is supplied as a link in the web service's "start" feed resource and has the link type **`http://www.vizrt.com/types/relation/section`** (see section 3.2.19).

```
curl --include -u user:password -X POST -H "Content-Type: application/atom+xml" \
> http://host-ip-address/webservice/escenic/section --upload-file my-new-section.xml
HTTP/1.1 100 Continue

HTTP/1.1 201 Created
Server: Apache-Coyote/1.1
Location: http://localhost:8080/webservice/escenic/section/35
Content-Type: application/atom+xml; type=entry
Content-Length: 0
Date: Tue, 20 Mar 2012 06:59:38 GMT
```

In order for the **`POST`** operation to work, you must specify a HTTP header as shown above.

If you're using **`curl`**, it's a good idea to specify **`--include`** with **`POST`** operations: **`curl`** will then output the response header returned from the web service as shown above, and you can verify whether or not the operation was successful:

- A response code in the 2*xx* range indicates success.
- A response code in the 4*xx* range means that you made an invalid addition and the server won't accept your new content item.
- A response code in the 5*xx* range means there is a server error.

The **Location** response specifies the location of the newly-created section record: you can retrieve the section record by submitting a **GET** request to this URL.

> A quick way of finding out how to create a correctly structured section in VDF format is to **GET** a section and copy the structure.

5.29 Delete a Section

Deleting a section is not as straightforward as deleting a content item or person. There may be content items that depend on its existence. Any content items that have this section as their home section must either be:

- Moved to a new home section, or
- Deleted together with the section

> When a selection is deleted it is actually removed from the database - the operation is irreversible. Moreover, any content items that you delete along with it are also physically deleted. So deleting content items indirectly in this way is a much more significant operation than deleting them directly - an ordinary content item delete operation just changes the content item's state to **deleted**.

To start a delete operation, send an HTTP **DELETE** request to the same URI that would be used to retrieve it (see section 5.25). For example:

```
curl --include -u user:password -X DELETE \
> http://host-ip-address/webservice/escenic/section/20
HTTP/1.1 303 See Other
Server: Apache-Coyote/1.1
Date: Tue, 20 Mar 2012 09:12:00 GMT
Location: http://host-ip-address/webservice/escenic/section/20/delete
```

The usual response to this request is **303 See Other**, and includes a **Location** header as shown above. You must then:

1. **GET** the resource referenced in the **Location** header:

   ```
   curl -u user:password http://host-ip-address/webservice/escenic/section/20/delete
   ```

   The resource will be an Atom entry containing an **empty** VDF payload document something like this:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <entry xmlns="http://www.w3.org/2005/Atom">
     <content type="application/vnd.vizrt.payload+xml">
       <vdf:payload xmlns:vdf="http://www.vizrt.com/types"
       model="http://host-ip-address/webservice/escenic/model/
   com.escenic.section.delete.20"/>
     </content>
   </entry>
   ```

2. **GET** the VDF model document referenced by the VDF payload:

   ```
   curl -u user:password http://host-ip-address/webservice/escenic/model/
   com.escenic.section.delete.20
   ```

This provides the information you need to be able to add correctly formatted information to the payload document:

```
<?xml version="1.0" encoding="UTF-8"?>
<vdf:model xmlns:vdf="http://www.vizrt.com/types">
  <vdf:schema>
    <vdf:fielddef name="com.escenic.section.delete.title"
                  label="Delete section"
                  mediatype="text/plain"
                  xsdtype="string"
                  contenteditable="false"/>
    <vdf:fielddef name="com.escenic.section.delete.replacement"
                  label="&lt;html&gt;The section with name &quot;Section
 Name&quot; contains 892 content item(s). You can&lt;br/&gt;choose a new home
 section for your content item(s) or allow the content item(s)&lt;br/&gt;to be
 deleted&lt;/html&gt;"
                  mediatype="text/plain"
                  xsdtype="string">
      <vdf:choice scope="limit">
        <vdf:collection src="escenic/section" select="title"/>
      </vdf:choice>
    </vdf:fielddef>
    <vdf:fielddef name="com.escenic.section.delete.confirmation"
                  label="Do you really want to delete this section?"
                  mediatype="text/plain"
                  xsdtype="boolean">
      <vdf:value>false</vdf:value>
    </vdf:fielddef>
  </vdf:schema>
</vdf:model>
```

3. Add information to the empty VDF payload document that complies with the schema in the model document. The VDF model specifies that the payload can contain three fields:

   - **`com.escenic.section.delete.title`**: This field is defined as read-only (**`contenteditable="false"`**) so there is no point specifing a value for it.
   - **`com.escenic.section.delete.replacement`**: If you return the URL of another section in this field, then any content items belonging to the deleted section will be moved to it. If you leave the field empty or unspecified, then the content items will be deleted along with the section.
   - **`com.escenic.section.delete.confirmation`** : You must return **`true`** in this field to confirm deletion of the section. If you return false or specify nothing then the delete operation is canceled.

   This example will complete the section deletion and move all the orphaned content items to a new section:

```
<?xml version="1.0" encoding="UTF-8"?>
<entry xmlns="http://www.w3.org/2005/Atom">
  <content type="application/vnd.vizrt.payload+xml">
    <vdf:payload xmlns:vdf="http://www.vizrt.com/types"
    model="http://host-ip-address/webservice/escenic/model/
com.escenic.section.delete.20">
      <vdf:field name="com.escenic.section.delete.replacement">
        <vdf:origin href="http://host-ip-address/webservice/escenic/section/15"/>
        <vdf:value>News</vdf:value>
      </vdf:field>
      <vdf:field name="com.escenic.section.delete.confirmation">
```

```
        <vdf:value>true</vdf:value>
      </vdf:field>
    </vdf:payload>
  </content>
</entry>
```

4. **PUT** the resource back to the same location. If you saved the Atom entry in a file called **my-delete-section.xml**, for example, then you could use the following **curl** command:

```
curl --include -u user:password -X PUT -H "Content-Type: application/atom+xml" \
> http://host-ip-address/webservice/escenic/section/20/delete --upload-file my-
delete-section.xml
HTTP/1.1 100 Continue

HTTP/1.1 204 No Content
Server: Apache-Coyote/1.1
Date: Tue, 19 Feb 2013 05:50:26 GMT
```

6 Advanced Features

This chapter discusses use of some of the Content Engine web service's more advanced features.

6.1 Using Change Logs Effectively

The change log resources (described in section 1.4) expose all the content items in a protection domain, ordered by their last date of modification.

The Content Engine offers two mechanisms for optimizing the efficiency of change logs:

- Polling the **`previous`** link. This is the basic method of using an Escenic change log – the method you should use if you cannot or do not need to make use of server-sent events.
- Server-sent events (SSE). This is an optional technique you can use in addition to polling the **`previous`** link that provides much better scalability. It is the recommended method of using Escenic change logs.

These mechanisms are described in the following sections.

6.1.1 Polling the Previous Link

As with most Atom feeds, the usual way to use them is to poll the original feed URI (for example **`http://`***host-ip-address***`/webservice/escenic/changelog/section/`***nnn***) at regular intervals, in order to see what's changed. If the number of changes exceeds the page size, the client can follow the **`next`** links until it reaches the page containing the last entry it processed.

Although this approach is a common way to use Atom feeds, it is less than optimal if the polling frequency is high, or if many clients are involved, as may be the case with the Content Engine web service. The first page of the change log, an Atom feed of several kilobytes, must be re-rendered every time the original feed URI is polled. Since most of the Content Engine's caches are not used due to authentication, the server is required to serve the same page very frequently, which has a detrimental effect on performance, and does not scale well with multiple clients or higher polling frequencies.

For this reason, the change log feeds include links that go right to the "edge" of the list, allowing clients to poll the actual start of the list instead of just the first page. This "edge" manifests itself as a **`previous`** link on the first page. It may initially seem illogical to have a **`previous`** link, implying as it does that there is a page **before** the first page. It is, however, very useful: there is a page before the first page, but it is empty unless a new change has been added to the log since the last poll. Once a client has polled the original feed once, it can subsequently poll the **`previous`** link. If no changes have been added, then the web service will return an empty feed, consuming very few server resources.

When new content items are added to the change log (or existing content items are modified and therefore moved to the top of the change log), polling the **`previous`** link will no longer return an empty feed: it will contain entries for all the new changes. The client can deal with these changes, and then continue polling **this** feed's **`previous`** link.

The consequence of using this approach is that most clients will be polling an empty feed most of the time, and therefore consuming a relatively small amount of server resources: the server is only required to actually render content items when there are changes to communicate to the clients. It

also makes client application code simpler, since the client does not need to remember the last entry it processed, and compare it with each entry to determine whether it is new: all entries in the feed returned by the **previous** link are guaranteed to be new, and do not need to be checked.

6.1.2 Server-sent Events

The technique described in section 6.1.1 helps to ensure that polling from multiple clients places the least possible strain on the Content Engine host. Polling in general, however, has limited scalability. If the Content Engine is required to respond to frequent polling requests from a large enough numbers of clients, then performance will suffer.

Server-sent events are a standard method of avoiding the need for polling, introduced with HTML 5. Instead of polling a server for changes, a client can open a persistent link to the server and receive notifications of changes via that link whenever they occur. This has two advantages over polling:

- Improved scalability on the server, since the server only needs to do something when a change occurs, rather than every time it receives a polling request.
- Improved response times on the client, since there is no polling interval delay: the client gets notification of a change as soon as it occurs.

Using server-sent events to get notification of changes to Content Engine change logs is very straightforward. All change log feeds contain a link that can be used to set up a persistent server-sent events connection to the Content Engine. Sending a change log **GET** request like this, for example:

```
curl -u user:password -X GET http://host-ip-address/webservice/escenic/changelog/
section/481
```

Returns, in addition to any changes, a link to an **event stream**.This link always has the following attributes:

rel
: Set to **http://www.vizrt.com/types/relation/changelog**

type
: Set to **text/event-stream**

href
: Set to **http://***host-ip-address***/webservice/escenic/changelog/sse**

For example:

```
<feed>
  <id>http://host-ip-address/webservice/escenic/changelog/section/481</id>
  <updated>2016-06-07T09:08:42.800Z</updated>
  <author>
    <name>Escenic Content Engine</name>
  </author>
  <link rel="self" href="http://host-ip-address/webservice/escenic/changelog/
section/481/before/16677017?count=10"/>
  <link rel="previous" href="http://host-ip-address/webservice/escenic/changelog/
section/481/after/16677016?count=10"/>
  <link rel="next" href="http://host-ip-address/webservice/escenic/changelog/
section/481/before/14914468?count=10"/>
  <link rel="http://www.vizrt.com/types/relation/changelog"
    href="http://host-ip-address/webservice/escenic/changelog/sse"
    type="text/event-stream"/>
```

```
  <entry>
    ...
  </entry>
  ...
</feed>
```

Sending a **GET** request to this URL sets up a persistent link over which the Content Engine can send notifications. If you send a **GET** request using **curl**, for example, you will see that curl does not return control to the command line prompt, but "hangs", because the connection that has been created has not closed.

```
curl -u user:password -X GET 'http://host-ip-address/webservice/escenic/changelog/sse'
```

If you now create a new content item, or make some other change from Content Studio or from a different terminal window, you will see that the change is reported using the open connection:

```
curl -u user:password -X GET 'http://host-ip-address/webservice/escenic/changelog/sse'
data: http://host-ip-address/webservice/escenic/changelog/section/22
```

Changes will continue to be reported in this way for as long as the connection remains open.

> Server-sent events reduce the load of responding to large numbers of polling requests, but at the price of requiring the Content Engine to hold large numbers of persistent connections open. This can quickly become a problem, since Tomcat cannot handle more than 200 simultaneous client connections. Escenic therefore now ships an **SSE Proxy** with the Content Engine. An SSE Proxy can handle up to 28,000 simultaneous client connections on behalf of the Content Engine. It is possible to run many such proxies in parallel, each of which only require one connection to the Content Engine, effectively removing any limitation on the total number of SSE connections that it is possible to handle.
>
> For details of how to set up and run one or more SSE proxies for use with the Content Engine, see the SSE Proxy documentation.

6.1.2.1 Using Server-sent Events

In the browser, An **EventSource** object can be used to receive server-sent event notifications. On creation, the object is connected to the source of notifications (in our case, the Content Engine's SSE link, **http://*host-ip-address*/webservice/escenic/changelog/sse**. It has an **onmessage** event that is fired every time a notification is received. So all you need to do is write an event function that responds appropriately to the notifications received. For example:

```
var source = new EventSource("http://host-ip-address/webservice/escenic/changelog/
sse");
source.onmessage = function(event) {
    // handle the SSE notification here:
    // event.data contains the URL of the updated change log
    // event.lastEventId contains a unique ID
};
```

The notifications generated by the Content Engine set two event properties:

data
: The URL of the change log that has been updated

lastEventId
: A unique ID for the event (not currently used??)

The event data sent by the Content Engine does not directly contain any details of what change has occurred - it simply sends the URL of the change log that has been updated. Your event code can then either ignore the event if the updated change log is not of interest, or send a **`GET`** request to the change log URL in order to retrieve the change details and deal with them. The **`GET`** request sent to the change log URL is identical to an ordinary polling request, except that in this case, the response is guaranteed to contain some changes.

6.2 Optimistic Concurrency

The Content Engine relies on a technique called **optimistic concurrency** to deal with concurrent updates to content items. Clients are expected to co-ordinate updates between themselves using optimistic concurrency mechanisms. For background information about this technique, see http://www.w3.org/1999/04/Editing/.

Clients are required to **either** make active use of optimistic concurrency when updating content, **or** explicitly bypass it (not recommended), so you need to understand this section in order to make a client that will update content items.

Optimistic concurrency works as follows:

1. Each client that wants to make a modification to a content item downloads a copy of it. Each item is accompanied by an HTTP header called **`ETag`**. An **`Etag`** header is a unique identifier generated by the web service, and looks something like this:

   ```
   ETag: "129694559e911ee4c6d04212982"
   ```

2. Each client remembers the **`ETag`** associated with the content item it is modifying.
3. After modifying the content item, the client performs a **`PUT`** operation to return it to the web service, including its **`ETag`** in an **`If-Match`** header. Note that an **`ETag`** value **includes its enclosing quote marks**. A correct **`PUT`** operation executed with **`curl`** would therefore look something like this:

   ```
   curl --include -u user:password -X PUT -H 'If-Match:
    "129694559e911ee4c6d04212982"' \
   > -H "Content-Type: application/atom+xml" http://host-ip-address/webservice/
   escenic/content/4 \
   > --upload-file my-edited-article.xml
   ```

4. The web service checks the supplied **`Etag`** to see if it matches the current copy of the content item. For the first client to return changes, this will be the case, and the **`PUT`** operation will succeed. For all other clients that have been working on the content item, the **`Etag`** will not match and the web service will return a **`412 PRECONDITION FAILED`** response.

When a client receives a **`412 PRECONDITION FAILED`** response it should download the latest copy of the content item (with its new **`ETag`**), re-apply its changes to the new copy, and then **`PUT`** it back, hopefully succeeding this time. If more than two clients are working on the same copy, however, this might not be the case. In the worst case a client might need to retry many times. Clients should be able to handle a certain number of retries, back off after a few retries, and lower the rate of retry, and possibly even request human intervention.

A client can circumvent the optimistic concurrency mechanism by sending the header **`"If-Match: *"`** in its **`PUT`** operations. This header specifies that the operation should succeed no matter which version is current. The web service will then accept the **`PUT`** request without performing any

concurrency checks. The consequence of doing so is that any changes made to the content item by other clients will be overwritten. Your client should therefore **`never`** do this unless you know the consequences and are certain that it will not cause problems.

7 Making an Update Service

Content Studio has an inbuilt mechanism for accessing a web service called an **update service**. You can configure an Escenic content type so that every time a specific field is modified, Content Studio sends the entire content item to such an update service. The update service can then modify the content item in some way and return it to Content Studio. This provides a simple mechanism for adding custom extensions to Content Studio. In order to extend Content Studio in this way, all you need to do is:

- Create a web service that complies with the update service specification.
- Add an element containing the URL of your service to a field definition in your publication's `content-type` resource.

Possible uses for an update service include:

- Adding custom error checks to fields. Your service could, for example, use a national database to verify entered post codes.
- Displaying or hiding one or more fields based on the value of another field. You might, for example, define a `View` field that determines which of the other fields are currently visible.
- Adding useful information to content items. Your service could for example, perform a word count and return the result (not possible with the current version).

The advantage of using this mechanism to extend Content Studio is ease of deployment. You do not need to make any changes at all to Content Studio itself. You simply deploy a web service and configure any content types that you want to make use of the service.

> Content Studio is currently only able to make limited use of the modified content item returned by an update service. Effectively, the only use case supported is that of error checking. Extensions to support other use cases are planned for future versions, however.

7.1 Update Service Specification

An Escenic Update Service is a web application that meets the following requirements:

- The service must accept HTTP `POST` requests containing **VDF Payload** documents.
- On receipt of a `POST` request containing a VDF Payload document, the service must return a modified version of the received document. The returned document must also be a valid VDF Payload document.

The VDF format specification includes a **VDF Model** format that can be used to define the structure of a VDF Payload document. Content Studio does not, however, currently provide update services with VDF model documents, so an update service must encapsulate some knowledge about the structure of the VDF payloads it is to deal with. Since the VDF payload documents represent Escenic content items, such information can be obtained, if needed, from publication content-type resources.

The only formal requirement regarding the data returned by an update service, is that it must be a valid VDF payload document. In practice, however, the only changes that will have any effect in Content Studio are the addition of `vdf:annotation` elements to `vdf:field` elements in the

submitted document. The **annotation** attributes supported by Content Studio (and their effect) are described below.

7.2 Supported Annotation Attributes

Content Studio recognizes and acts on a **vdf:annotation** element containing any of the following attributes:

tip
The content of this attribute is used to set or change the tooltip text of the **vdf:annotation** element's owning field.

visibility
The content of this attribute is used to set or change the rules governing the visibility of the **vdf:annotation** element's owning field, as follows:

hidden
The field is hidden.

visible
The field is made visible.

Any other **visibility** values are ignored by Content Studio.

contenteditable
The content of this attribute determines whether or not the **vdf:annotation** element's owning field is editable. If it is set to **false**, then Content Studio will make the owning field read-only.

7.3 Using an Update Service

Content Studio only sends content items to an update service on demand. In order to force Content Studio to use an update service, you must add a vdf:updateservice element to one of the **field** elements in a publication's **content-type** resource. For example:

```
<field type="basic" name="postcode">
   <viz:updateservice href="update-service-uri"/>
</field>
```

When this field is modified by a user in Content Studio, a VDF Payload document representing the entire content item is sent to *update-service-uri*. *update-service-uri* should be a relative URI referencing a service in the local domain.

Note that although the **viz:updateservice** "trigger" is only added to one element, the update service is free to modify any fields in the submitted document, and Content Studio checks all fields in the returned document for **vdf:annotation** elements.

7.4 Example

The **postcode** field in a content type definition contains a **viz:updateservice** element referencing a **check-postcode** service:

```
<content-type name="review">
  ...
  <panel name="address">
    ...
    <field type="basic" name="postcode">
       <viz:updateservice href="/webservice/check-postcode"/>
    </field>
    ...
  </panel>
  ...
</content-type>
```

When the postcode field is modified, Content Studio **POST**s the content item as a payload document to **/webservice/check-postcode**:

```
<payload xmlns="http://www.vizrt.com/types">
  ...
  <field name="postcode">
    <value>XYZ</value>
  </field>
  ...
</payload>
```

The **check-postcode** service checks the content of the **postcode** field, finds that it is invalid and returns an error message in a **vdf:annotation tip** attribute:

```
<payload xmlns="http://www.vizrt.com/types">
  ...
  <field name="postcode">
    <annotation tip="Cannot find the postcode 'XYZ'."/>
    <value>XYZ</value>
  </field>
  ...
</payload>
```

Content Studio sets this message as a tooltip for the field.

8 VDF Reference

This chapter contains formal descriptions of the VDF document formats:

- Model document format
- ayload document format

8.1 model

The **vdf-model** schema defines the content model of VDF model documents. All VDF payload documents include a reference to a VDF model document that defines the content and structure of the payload document.

A VDF model serves two purposes:

- It defines allowed content and structure for clients constructing a VDF payload document that is to be **POST**ed to the Content Engine web service.
- It provides a key that can be used when parsing a VDF payload document exposed by the Content Engine web service.

VDF model documents (unlike VDF payload documents) are always supplied by the Content Engine - your client should never need to create or modify a model document. This documentation is therefore only intended to help you interpret a VDF model. When a client retrieves a VDF payload document from the Content Engine, it can use the associated VDF model to help parse and/or display the payload correctly. When creating or modifying a payload document, the associated model document can be used as a guide to ensure that the payload document is correctly structured.

Namespace URI

The namespace URI of the **model** schema is **http://www.vizrt.com/types**.

Root Element

The root of a **model** file must be a **model** element.

8.1.1 alternative

Container for a possible field value. An empty **alternative** element indicates that an empty field or null value is allowed.

Syntax

```
<alternative
    label="text"?
  />
```

Attributes

`label="`*text*`"` (optional)
The display label of the alternative, for use in interactive clients.

8.1.2 choice

Defines a set of allowed values for the field defined by the parent **`fielddef`** element.

Syntax

```
<choice
    scope="text"
    multiple="(true|false)"
  >
  (<alternative/>+|<collection/>)
</choice>
```

Attributes

`scope="`*text*`"`
Determines how the specified set of values is to be used by clients:

`limit`
Only the specified values are allowed.

`??`
The specified values are only suggestions; other values are allowed.??

`??`
Any more??

`multiple="(true|false)"`
Specifies whether or not multiple choices are allowed.

8.1.3 collection

A reference to an Atom collection containing possible field values.

Syntax

```
<collection
    src="text"
    select="text"
  />
```

Attributes

`src="`*text*`"`
The collection URL.

`select="`*text*`"`
The name of the Atom collection field from which the possible values are retrieved. Interactive clients can always retrieve value labels from the collection's **`Title`** fields

8.1.4 fielddef

Defines the content and structure of a VDF payload field. A **`fielddef`** element may also contain display "hints" that can be used by graphical clients to control application layout. These hints are

often provided in the form of **ui** elements belonging to the **http://xmlns.escenic.com/2008/interface-hints** namespace. For detailed information about these elements, see interface-hints.

Syntax

```
<fielddef
    name="text"
    label="text"
    xsdtype="text" mediatype="text"?
  >
  (<choice>...</choice>|<listdef>...</listdef>)?(ANYTHING|text)
</fielddef>
```

Attributes

name="*text*"
The name of the field.

label="*text*"
The display label of the field, for use in interactive clients.

xsdtype="*text*" (optional)
The type of the value in the field. A subset of XSD data types are used. For information about XSD data types, see http://www.w3.org/TR/xmlschema-2/#built-in-datatypes.

mediatype="*text*" (optional)
The MIME type of the value in the field. This attribute provides additional type information about the value in the field. The MIME types used fall into two categories:

- Standard MIME type identifiers defined in the IANA MIME type registry (http://www.iana.org/assignments/media-types/).
- Proprietary MIME types defined by Escenic. Currently only one such MIME type is used:

  com.escenic.relationType
  Indicates that the field contains an Escenic relation.

Although **fielddef** elements with an **xsdtype** always have a **mediatype** attribute as well, the **mediatype** attribute does not always contain useful information. Clients should always check the **mediatype** attribute for fields with an **xsdtype** of **string** or **link**. For all other **xsdtype** values, **mediatype** is always set to **text/plain** and carries no meaning.

8.1.5 listdef

A container for a VDF schema defining the content and structure of the items in a list field.

Syntax

```
<listdef>
  <schema>...</schema>+
</listdef>
```

8.1.6 model

A container for one or more VDF schemas defining the content and structure of a specific VDF payload document. If the model contains more than one **schema** then you should use the first one.

Syntax

```
<model>
  <schema>...</schema>+
</model>
```

8.1.7 schema

A container for a sequence of one or more VDF field definitions specifying the structure of:

- **Either** the structure of a VDF payload document
- **Or** the structure of the items in a list field within a payload document

All the fields defined in the schema must be present in the described payload document or sequence of list elements.

Syntax

```
<schema>
  <fielddef>...</fielddef>*
</schema>
```

8.2 payload

The **`vdf-payload`** schema defines the content model of VDF payload documents. All content exposed by the Escenic Content Engine's web service is exposed in the form of VDF payload documents, and all content **`POST`**ed to the web service is required to be supplied in the form of VDF payload documents.

`vdf-payload` documents in themselves have a very simple and permissive structure. A VDF payload document is basically a simple sequence of fields that can have any names and contain a wide range of different value types. Fields can also contain nested fields and nested payload documents.

However, a VDF payload is also required to include a reference to a VDF model document, which is a schema defining the content allowed in this specific VDF payload instance.

A VDF model serves two purposes:

- It defines allowed content and structure for clients constructing a VDF payload document that is to be **`POST`**ed to the Content Engine web service.
- It provides a key that can be used when parsing a VDF payload document exposed by the Content Engine web service.

Namespace URI

The namespace URI of the **`payload`** schema is **`http://www.vizrt.com/types`**.

Root Element

The root of a **`payload`** file must be a **`payload`** element.

8.2.1 field

A field containing a name-value pair. As well as its own value, a field may contain zero or more nested sub-fields.

Syntax

```
<field
    name="text"?
  >
  <value>...</value>
  <field>...</field>*
  <origin/>?
</field>
```

Attributes

`name="`*text*`"` (optional)
The name of a **`field`**. What field names are valid in any particular payload is determined by the payload's model document.

8.2.2 list

A container for a sequence of 0 or more embedded **`payload`** elements.

Syntax

```
<list>
  <payload>...</payload>*
</list>
```

8.2.3 origin

This element may only appear as the child of a collection **`field`** element. It represents the **origin** of the collection field's value (that is, the Atom entry from which the value was retrieved).

Syntax

```
<origin
    href="text"
  />
```

Attributes

`href="`*text*`"`
The URI of an Atom entry from which a collection field value was retrieved.

8.2.4 payload

A VDF payload holding Escenic content.

Syntax

```
<payload
    model="text"?
  >
  <field>...</field>*
</payload>
```

Attributes

`model="`*`text`*`"` (optional)
A URL referencing the VDF model document that defines the content of this document. Although this attribute is formally defined as optional, it is **required** for **`payload`** elements that form the root of a VDF payload document. For embedded payload elements that are the children of a **`list`** element it is in fact optional. If it is omitted, then the embedded **`payload`** inherits the model of its nearest **`payload`** ancestor.

8.2.5 value

The value of a **`field`**. The value may in theory be a simple value, any valid XML content or a **`list`** element. In practice, however, the content of a **`value`** element is determined by the payload's model document.

`value` elements may also appear in VDF model documents. Depending on the context in which a **`value`** element appears in a model document, it may represent either a default value or a possible value in a set of alternatives.

Syntax

```
<value>
  (<list>...</list>|(ANYTHING|text))
</value>
```