Specification of contineo's REST Interface

Sebastian Stein
seb.stein@hpfsc.de

2007-02-07, contineo version 2.5

Contents

1 Introduction

This document describes the REST web service interface of the document management system contineo[1]. The interface is intended for an integration of contineo with another software system. It is not an end-user interface used by human users. A possible integration scenario is for example to use contineo together with a workflow engine like Bonita[2]. A workflow engine like Bonita can directly invoke functionality of contineo provided through this interface like downloading a document and sending it to the next user in the workflow.

This specification is structured as follows. In the next section the general concept is described. This includes a discussion of HTTP status codes and authentication. Section 3 on page 6 describes each service offered by contineo's REST interface. Where possible, the input and output messages are described in detail using W3 XML schema files. Those schema files are also available as separate files on contineo's homepage.

The development of contineo's REST interface was funded by the Italian research project BioNet and the Italian technology park TecnoMarche[3].

2 General Concept

2.1 REST Web Service Interface

There are mainly 2 different interface technologies for web services used today. One is based on a stack of different public web service standards like WSDL [CCMW01] and SOAP [GHM+03]. This technology stack is often summarised as WS*-stack and the belonging standards are developed and maintained e. g. by the W3 Consortium[4].

The other technology is called representation state transfer (REST) and is mainly based on HTTP [FGM+99]. Where the WS*-stack technology is a concrete set of standards, REST is more like an abstract concept based on the work of Fielding [Fie00].

Both technologies are suited to fulfill the aim of integrating contineo with other applications. There are, however, some differences in the infrastructure needed to support the different technologies. Today, contineo can be deployed on every Java servlet specification[5] compliant server like Apache Tomcat[6]. Such servlet containers usually do not directly support SOAP, which is a core requirement for implementing the W3*-stack technology. In

[1] http://contineo.sourceforge.net/
[2] http://bonita.objectweb.org
[3] http://www.pstmarche.it/
[4] http://www.w3.org/
[5] http://jcp.org/aboutJava/communityprocess/final/jsr154/index.html
[6] http://tomcat.apache.org/

contrast, all Java servlet containers support HTTP, which is the only requirement for the REST technology. This is the main reason why it was decided to implement a REST interface instead of a W3*-stack interface.

A WS*-stack interface provides operations to alter some kind of object. WSDL therefore describes the different interfaces and operations from a message oriented point of view. In contrast, a REST interface focus on the notion of resources. In context of contineo resources are for example the documents stored in contineo as well as the folders and sub-folders. The so called CRUD[7] pattern can be applied to a resource. For example, a document can be uploaded (created), downloaded (read), changed (update) and removed (deleted) in contineo. This should also be possible through the REST interface. According to Fielding [Fie00] there is no need for special operations to alter resources. Instead, the standardised HTTP functions POST, GET, PUT and DELETE should be used. Based on that simple mechanism a REST interface only requires full HTTP 1.1 support on the server. This requirement is fulfilled by all Java servlet containers.

The resources are exposed to the client by URLs. The URLs can be accessed through HTTP. A URL to download (HTTP GET) a document with the document id 7 might look as follows:

```
http://.../contineo/rest/document/7
```

This URL would return the document so that the client can process it as needed. The same URL would be used with a HTTP DELETE request to delete the document.

The following URL would return the content of folder 28 including links to all sub-folders and documents stored in the requested folder:

```
http://.../contineo/rest/folder/28
```

In that case an XML document is returned containing the needed information. The structure of this document and the documents of the other interfaces are described in section 3 starting on page 6.

Besides returning a document, HTTP supports also so called HTTP status codes to give additional information about how the request was processed in contineo. A complete list of all HTTP status codes used by contineo's REST interface can be found in section 2.2 on page 4.

Another important issue is security. Of course a client should only access those resources where it has privileges to. The authentication mechanism used by contineo's REST interface is described in section 2.3 on page 5.

2.2 Usage of HTTP Status Codes

For each request to contineo's REST interface a HTTP status code will be returned. Each HTTP status code is represented by a defined number with

[7] create, read, update and delete

standardised semantics. Below is a list with all status codes supported and their specific meanings in context of contineo's REST interface:

200 Ok (succesful answer to GET)

201 Created (succesful answer to POST)

400 Bad request (syntax errors, so for example the XML document received from the client could not be validated or the requested URL is not supported)

403 Forbidden (user has no access rights to requested resource)

404 Not found (requested resource was not found)

405 Method not allowed (not all resources support the same set of HTTP methods)

500 Internal server error (an error occured in contineo while processing the request)

2.3 Authentication

The REST interface is available through ordinary URLs. The REST interface must be enabled in contineo's settings. Each human user has to authenticate by providing username and password while using the graphical user interface. A client using the REST interface must authenticate in the same way. The administrator has to create an ordinary user account and assign this user account to a user group. The client using the REST interface will be able to access all documents and folders which the user group has access to.

To find a simple solution for providing authentication support it was decided that the client must send username and password with each request. There is no session handling, so before a request is processed by contineo it will check the provided credentials.

In order to simplify the implementation of the REST interface but also the integration code needed on the client, it was decided to not use HTTP authentication, but instead two custom HTTP header fields. The header fields are:

```
X-username=theUsername
X-password=thePassword
```

It can be seen that the credentials are transfered in plain text. This is a security risk. To overcome this problem, the user should configure contineo's servlet container to only allow HTTPS access. Please consult the documentation of the used servlet container for a description how to enable HTTPS support.

3 Services Provided

3.1 Overview

In this section the different interfaces are described. This description does not include the HTTP status codes returned, because they apply to all requests and are not specific. A description of the supported HTTP status codes can be found in section 2.2 on page 4. This description does also not include how authentication is handled. This is described in detail in section 2.3 on page 5.

3.2 Create Folder

Summary:	Creates a new folder in folder XX.
URL:	http://.../contineo/rest/folder/XX
HTTP Method:	PUT
Parameter XX:	specifies the parent folder for the folder to be created
PUT Message:	The client must send a multiparted HTTP message to contineo. See the descriptions below for more information.
Return:	An XML document is returned describing some aspects of the created folder. The XML schema for the document is given below.

The client must send a multipart HTTP message [FGM+99] to contineo. The message consists of one file. The file is an XML document specifying the name of the new folder and a flag that a folder and not a document should be created. This message part must have the following part identifier:

`contineoCheckInInfo.xml`

The XML schema for this file is shown at the end of this section. Please note, while sending this document to contineo, only the required fields must be set. If the folder was successfully created, the same document with additional information is send back to the client. A valid instance of such a XML document sent to contineo can be found in section 4.1 on page 16.

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <xs:schema targetNamespace="http://contineo.sourceforge.
    net/folder" version="2.5" xml:lang="en"
3  xmlns:xs="http://www.w3.org/2001/XMLSchema">
4  <xs:annotation>
5   <xs:documentation
6    source="The HTTP PUT interface /document for creating
       a new folder or document in an already existing
       folder. This schema specifies the structure of how
        the meta data describing the new folder/document
       must be submitted."
```

```
    xml:lang="en"/>
  </xs:annotation>
  <xs:element name="createFolderData">
   <xs:complexType>
    <xs:sequence>
     <xs:element maxOccurs="1" minOccurs="1" name="
        folderInfo">
      <xs:complexType>
       <xs:sequence>
        <xs:element name="createFolder" type="xs:boolean"/
           >
        <xs:element name="name" type="xs:string"/>
        <xs:element name="id" type="xs:int" minOccurs="0"
           maxOccurs="1"/>
        <xs:element name="url" type="xs:anyURI" minOccurs=
           "0" maxOccurs="1"/>
        <xs:element name="metadataURL" type="xs:anyURI"
           minOccurs="0" maxOccurs="1"/>
        <xs:element name="language" minOccurs="0"
           maxOccurs="1">
         <xs:simpleType>
          <xs:restriction base="xs:string">
           <xs:enumeration value="de"/>
           <xs:enumeration value="en"/>
           <xs:enumeration value="es"/>
           <xs:enumeration value="fr"/>
           <xs:enumeration value="it"/>
          </xs:restriction>
         </xs:simpleType>
        </xs:element>
       </xs:sequence>
      </xs:complexType>
     </xs:element>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
</xs:schema>
```

3.3 Download Folder Content

Summary:	Returns the content of folder XX. If XX is not given, it returns the content of the root folder.
URL:	http://.../contineo/rest/folder/XX
HTTP Method:	GET
Parameter XX:	specifies the folder to be returned; optional
Return:	The content of the folder is returned as a XML document. The XML schema for the document is given below. An example XML document can be found in section 4.2 on page 16.

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <xs:schema targetNamespace="http://contineo.sourceforge.
       net/folder" version="2.5" xml:lang="en"
3    xmlns:xs="http://www.w3.org/2001/XMLSchema">
4    <xs:annotation>
5      <xs:documentation
6        source="The HTTP GET interface /folder/id returns
             the content of the folder with the given id.
             This schema specifies the structure of how the
             folder content is returned."
7        xml:lang="en"/>
8    </xs:annotation>
9    <xs:element name="folderContent">
10     <xs:complexType>
11       <xs:sequence>
12         <xs:element maxOccurs="1" minOccurs="1" name="
               header">
13           <xs:complexType>
14             <xs:sequence>
15               <xs:element name="id" type="
                     xs:positiveInteger"/>
16               <xs:element name="name" type="xs:string"/>
17               <xs:element name="writeable" type="
                     xs:boolean"/>
18               <xs:element name="parentId" type="
                     xs:positiveInteger"/>
19               <xs:element name="parentName" type="
                     xs:string"/>
20             </xs:sequence>
21           </xs:complexType>
22         </xs:element>
23         <xs:element maxOccurs="unbounded" minOccurs="0"
             name="subfolder">
24           <xs:complexType>
25             <xs:sequence>
26               <xs:element name="id" type="
                     xs:positiveInteger"/>
```

```
27                <xs:element name="name" type="xs:string"/>
28                <xs:element name="writeable" type="
                      xs:boolean"/>
29                <xs:element name="url" type="xs:anyURI"/>
30              </xs:sequence>
31            </xs:complexType>
32          </xs:element>
33          <xs:element maxOccurs="unbounded" minOccurs="0"
               name="document">
34            <xs:complexType>
35              <xs:sequence>
36                <xs:element name="name" type="xs:string"/>
37                <xs:element name="id" type="
                      xs:positiveInteger"/>
38                <xs:element name="writeable" type="
                      xs:boolean"/>
39                <xs:element name="downloadURL" type="
                      xs:anyURI"/>
40                <xs:element name="metadataURL" type="
                      xs:anyURI"/>
41                <xs:choice>
42                  <xs:element name="checkoutURL" type="
                        xs:anyURI"/>
43                  <xs:element name="checkinURL" type="
                        xs:anyURI"/>
44                </xs:choice>
45              </xs:sequence>
46            </xs:complexType>
47          </xs:element>
48        </xs:sequence>
49      </xs:complexType>
50    </xs:element>
51  </xs:schema>
```

3.4 Delete Folder

Summary:	Deletes folder XX.
URL:	http://.../contineo/rest/folder/XX
HTTP Method:	DELETE
Parameter XX:	specifies the folder to be deleted
Return:	nothing

3.5 Create Document

Summary:	Creates a new document in folder XX.
URL:	http://.../contineo/rest/folder/XX
HTTP Method:	PUT
Parameter XX:	specifies the parent folder for the document to be created
PUT Message:	The client must send a multiparted HTTP message to contineo. See the descriptions below for more information.
Return:	An XML document is returned describing some aspects of the created document. The XML schema for the document is given below.

The client must send a multipart HTTP message [FGM+99] to contineo. The message consists of two files. One file is an XML document specifying the that a document and not a folder should be created. This message part must have the following part identifier:

`contineoCheckInInfo.xml`

The XML schema for this file is shown at the end of this section. Please note, while sending this document to contineo, only the required fields must be set. If the document was successfully created, the same XML document with additional information is send back to the client. A valid instance of such a XML document sent to contineo can be found in section 4.3 on page 18.

The other file send to contineo in the multiparted message is the new document to be created. The name of the message part will be used as the filename within contineo.

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <xs:schema targetNamespace="http://contineo.sourceforge.
       net/folder" version="2.5" xml:lang="en"
3   xmlns:xs="http://www.w3.org/2001/XMLSchema">
4   <xs:annotation>
5    <xs:documentation
6     source="The HTTP PUT interface /document for creating
          a new folder or document in an already existing
          folder. This schema specifies the structure of how
           the meta data describing the new folder/document
          must be submitted."
7     xml:lang="en"/>
8   </xs:annotation>
9   <xs:element name="createFolderData">
10   <xs:complexType>
11    <xs:sequence>
12     <xs:element maxOccurs="1" minOccurs="1" name="
          folderInfo">
```

```
13        <xs:complexType>
14         <xs:sequence>
15          <xs:element name="createFolder" type="xs:boolean"/
                >
16          <xs:element name="name" type="xs:string"/>
17          <xs:element name="id" type="xs:int" minOccurs="0"
                maxOccurs="1"/>
18          <xs:element name="url" type="xs:anyURI" minOccurs=
                "0" maxOccurs="1"/>
19          <xs:element name="metadataURL" type="xs:anyURI"
                minOccurs="0" maxOccurs="1"/>
20          <xs:element name="language" minOccurs="0"
                maxOccurs="1">
21           <xs:simpleType>
22            <xs:restriction base="xs:string">
23             <xs:enumeration value="de"/>
24             <xs:enumeration value="en"/>
25             <xs:enumeration value="es"/>
26             <xs:enumeration value="fr"/>
27             <xs:enumeration value="it"/>
28            </xs:restriction>
29           </xs:simpleType>
30          </xs:element>
31         </xs:sequence>
32        </xs:complexType>
33       </xs:element>
34      </xs:sequence>
35    </xs:complexType>
36   </xs:element>
37  </xs:schema>
```

3.6 Download Document

Summary:	Returns document XX. If YY is given, this version of the document is returned, otherwise the latest version is returned.
URL:	http://.../contineo/rest/document/XX/YY
HTTP Method:	GET
Parameter XX:	specifies the document to be returned
Parameter YY:	specifies the version to be returned; optional
Return:	The document is returned. The mime-type of the response is set to the mime-type of the document. In addition the filename is set in the HTTP headers.

3.7 Download Document Meta Data

Summary:	Returns the meta data of document XX. This includes a list of all versions.
URL:	http://.../contineo/rest/document/XX/meta
HTTP Method:	GET
Parameter XX:	specifies which document's meta data should be returned
Return:	An XML document is returned containing the document's meta data as well as a list of all versions. The XML schema for the document is given below. An example XML document can be found in section 4.4 on page 18.

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <xs:schema targetNamespace="http://contineo.sourceforge.
       net/document" version="2.5" xml:lang="en"
3    xmlns:xs="http://www.w3.org/2001/XMLSchema">
4    <xs:annotation>
5      <xs:documentation
6        source="The HTTP GET interface /document/id/meta
             returns a document containing all meta data for
              the given document. This includes a list of
             all previous versions as well as entries like
             the author and source date. This schema
             specifies the structure of how the meta data is
              returned."
7        xml:lang="en"/>
8    </xs:annotation>
9    <xs:element name="documentMetaData">
10     <xs:complexType>
11       <xs:sequence>
12         <xs:element maxOccurs="1" minOccurs="1" name="
               header">
13           <xs:complexType>
14             <xs:sequence>
15               <xs:element name="id" type="
                     xs:positiveInteger"/>
16               <xs:element name="name" type="xs:string"/>
17               <xs:element name="writeable" type="
                     xs:boolean"/>
18               <xs:element name="downloadURL" type="
                     xs:anyURI"/>
19               <xs:element name="parentId" type="
                     xs:positiveInteger"/>
20               <xs:element name="parentName" type="
                     xs:string"/>
21               <xs:element name="uploadUser" type="
                     xs:string"/>
```

```
22              <xs:element name="uploadDate" type="xs:date
                    " minOccurs="0" maxOccurs="1"/>
23              <xs:element name="source" type="xs:string"/
                    >
24              <xs:element name="author" type="xs:string"/
                    >
25              <xs:element name="creationDate" type="
                    xs:date" minOccurs="0" maxOccurs="1"/>
26              <xs:element name="documentType" type="
                    xs:string"/>
27              <xs:element name="creationLocation" type="
                    xs:string"/>
28              <xs:element name="language" minOccurs="0"
                    maxOccurs="1">
29                <xs:simpleType>
30                  <xs:restriction base="xs:string">
31                    <xs:enumeration value="de"/>
32                    <xs:enumeration value="en"/>
33                    <xs:enumeration value="es"/>
34                    <xs:enumeration value="fr"/>
35                    <xs:enumeration value="it"/>
36                  </xs:restriction>
37                </xs:simpleType>
38              </xs:element>
39              <xs:choice>
40                <xs:element name="checkoutURL" type="
                      xs:anyURI"/>
41                <xs:element name="checkinURL" type="
                      xs:anyURI"/>
42              </xs:choice>
43            </xs:sequence>
44          </xs:complexType>
45        </xs:element>
46        <xs:element maxOccurs="unbounded" minOccurs="1"
              name="versionInfo">
47          <xs:complexType>
48            <xs:sequence>
49              <xs:element name="id" type="xs:string"/>
50              <xs:element name="downloadURL" type="
                    xs:anyURI"/>
51              <xs:element name="uploadUser" type="
                    xs:string"/>
52              <xs:element name="versionDate" type="
                    xs:date" minOccurs="0" maxOccurs="1"/>
53              <xs:element name="description" type="
                    xs:string"/>
54            </xs:sequence>
55          </xs:complexType>
56        </xs:element>
```

```
57        </xs:sequence>
58      </xs:complexType>
59    </xs:element>
60  </xs:schema>
```

3.8 Check out Document

Summary:	Marks document XX as checked out and returns the latest version for editing.
URL:	http://.../contineo/rest/document/XX/checkout
HTTP Method:	GET
Parameter XX:	specifies the document to be checked out
Return:	The document is returned. The mime-type of the response is set to the mime-type of the document. In addition the filename is set in the HTTP headers.

3.9 Upload new Document Version

Summary:	Upload a new version of a document together with a change description. The document is marked as checked in.
URL:	http://.../contineo/rest/document/XX
HTTP Method:	POST
Parameter XX:	specifies which document should be checked in and updated
POST Message:	The client must send a multiparted HTTP message to contineo. See the descriptions below for more information. An example can be found in section 4.5 on page 19.
Return:	nothing

The client must send a multipart HTTP message [FGM+99] to contineo. The message consists of two files. One file is an XML document describing the changes compared to the previous version. The XML schema for this file is shown at the end of this section. This part must have the following part identifier:

`contineoCheckInInfo.xml`

The second file in the multiparted message is the new version of the document. This part should have the name of the uploaded file.

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <xs:schema targetNamespace="http://contineo.sourceforge.
       net/document" version="2.5" xml:lang="en"
3    xmlns:xs="http://www.w3.org/2001/XMLSchema">
4    <xs:annotation>
```

```
5       <xs:documentation
6         source="The HTTP POST interface /document for
              uploading a new version of an already existing
              document (checkin). This is not intended for
              uploading a complete new document! This schema
              specifies the structure of how the meta data
              describing the uploaded version must be
              submitted."
7         xml:lang="en"/>
8     </xs:annotation>
9     <xs:element name="versionPostData">
10      <xs:complexType>
11        <xs:sequence>
12          <xs:element maxOccurs="1" minOccurs="1" name="
                versionInfo">
13            <xs:complexType>
14              <xs:sequence>
15                <xs:element name="description" type="
                      xs:string"/>
16                <xs:element name="versionType">
17                  <xs:simpleType>
18                    <xs:restriction base="xs:string">
19                      <xs:enumeration value="
                            newMajorVersion"/>
20                      <xs:enumeration value="newSubVersion"
                            />
21                      <xs:enumeration value="oldVersion"/>
22                    </xs:restriction>
23                  </xs:simpleType>
24                </xs:element>
25              </xs:sequence>
26            </xs:complexType>
27          </xs:element>
28        </xs:sequence>
29      </xs:complexType>
30    </xs:element>
31  </xs:schema>
```

3.10 Delete Document

Summary:	Deletes document XX.
URL:	http://.../contineo/rest/document/XX
HTTP Method:	DELETE
Parameter XX:	specifies the document to be deleted
Return:	nothing

4 Message Examples

This section provides examples XML document for each XML schema shown in the previous section. The files included here show possible and valid instances of the XML schema files.

4.1 Example Creating new Folder

This is an example for the XML schema shown in section 3.2 on page 6. Please note, in contrast to uploading a new folder you have to specify the language of the document. Currently, contineo supports the following list of languages for documents:

- de - German
- en - English
- es - Spanish
- fr - French
- it - Italian

```
<?xml version="1.0" encoding="UTF-8"?>
<createFolderData xmlns="http://contineo.sourceforge.net/
    folder"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://contineo.sourceforge.net/
    folder http://contineo.sourceforge.net/schemas/ver2
    .5/put-folder-doc.xsd ">
 <folderInfo xmlns="">
  <createFolder>true</createFolder>
  <name>folderName</name>
 </folderInfo>
</createFolderData>
```

4.2 Example Download Folder Content

This is an example for the XML schema shown in section 3.3 on page 8.

```
<?xml version="1.0" encoding="UTF-8"?>
<folderContent xmlns="http://contineo.sourceforge.net/
    folder"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://contineo.sourceforge.net/
         folder http://contineo.sourceforge.net/schemas/
         ver2.5/get-folder.xsd">
     <header xmlns="">
```

```
        <id>30</id>
        <name>folder30</name>
        <writeable>true</writeable>
        <parentId>5</parentId>
        <parentName>rootFolder</parentName>
    </header>

    <subfolder xmlns="">
        <id>31</id>
        <name>folder31</name>
        <writeable>true</writeable>
        <url>http://localhost:8080/contineo/rest/
            folder/31</url>
    </subfolder>

    <subfolder xmlns="">
        <id>32</id>
        <name>folder32</name>
        <writeable>false</writeable>
        <url>http://localhost:8080/contineo/rest/
            folder/32</url>
    </subfolder>

    <document xmlns="">
        <name>test.xls</name>
        <id>37</id>
        <writeable>true</writeable>
        <downloadURL>http://localhost:8080/contineo/rest/
            document/37</downloadURL>
        <metadataURL>http://localhost:8080/contineo/rest/
            document/37/meta</metadataURL>
        <checkoutURL>http://localhost:8080/contineo/rest/
            document/37/checkout</checkoutURL>
    </document>

    <document xmlns="">
        <name>interesting.pdf</name>
        <id>38</id>
        <writeable>true</writeable>
        <downloadURL>http://localhost:8080/contineo/rest/
            document/38</downloadURL>
        <metadataURL>http://localhost:8080/contineo/rest/
            document/38/meta</metadataURL>
        <checkinURL>http://localhost:8080/contineo/rest/
            document/38/checkin</checkinURL>
    </document>

</folderContent>
```

4.3 Example Creating new Document

This is an example for the XML schema shown in section 3.5 on page 10. Please note that the language tag is optional and not required by the XML schema. If no language tag is given, English is set as default language.

```
<?xml version="1.0" encoding="UTF-8"?>
<createFolderData xmlns="http://contineo.sourceforge.net/folder"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://contineo.sourceforge.net/folder http://contineo.sourceforge.net/schemas/ver2.5/put-folder-doc.xsd ">
 <folderInfo xmlns="">
  <createFolder>false</createFolder>
  <name>docName</name>
  <language>en</language>
 </folderInfo>
</createFolderData>
```

4.4 Example Get Document Meta Data

This is an example for the XML schema shown in section 3.7 on page 12.

```
<?xml version="1.0" encoding="UTF-8"?>
<documentMetaData xmlns="http://contineo.sourceforge.net/document"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://contineo.sourceforge.net/document http://contineo.sourceforge.net/schemas/ver2.5/get-meta.xsd">
     <header xmlns="">
          <id>37</id>
          <name>test</name>
          <writeable>true</writeable>
          <downloadURL>http://localhost:8080/contineo/rest/document/37</downloadURL>
          <parentId>30</parentId>
          <parentName>folder30</parentName>
          <uploadUser>admin</uploadUser>
          <uploadDate>2006-05-07</uploadDate>
          <source/>
          <author>Author1, Author3</author>
          <creationDate>2006-05-04</creationDate>
          <documentType>sales document</documentType>
          <creationLocation>Office in UK</creationLocation>
          <language>en</language>
          <checkoutURL>http://localhost:8080/contineo/rest/document/37/checkout</checkoutURL>
     </header>
```

```
22
23      <versionInfo xmlns="">
24          <id>1.0</id>
25          <downloadURL>http://localhost:8080/contineo/rest/
                document/37/1.0</downloadURL>
26          <uploadUser>admin</uploadUser>
27          <versionDate>2006-05-04</versionDate>
28          <description>Initial Upload of Document</
                description>
29      </versionInfo>
30
31      <versionInfo xmlns="">
32          <id>1.1</id>
33          <downloadURL>http://localhost:8080/contineo/rest/
                document/37/1.1</downloadURL>
34          <uploadUser>Latest Author</uploadUser>
35          <versionDate>2006-05-06</versionDate>
36          <description>Some minor changes</description>
37      </versionInfo>
38  </documentMetaData>
```

4.5 Example Post New Document Version (Checkin)

This is an example for the XML schema shown in section 3.9 on page 14.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <versionPostData xmlns="http://contineo.sourceforge.net/
       document"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xsi:schemaLocation="http://contineo.sourceforge.net/
            document http://contineo.sourceforge.net/schemas/
            ver2.5/post-version.xsd">
5       <versionInfo xmlns="">
6           <description>This is a new major version of the
                document.</description>
7           <versionType>newMajorVersion</versionType>
8       </versionInfo>
9  </versionPostData>
```

5 Querying the REST Interface using curl

Contineo's REST interface is meant for integrating contineo with other programs. However, for testing the interface it can be cumbersome to first implement a complete program. In this section another solution is presented based on the *curl*[8] tool. Curl is a command-line tool for sending requests

[8] http://curl.haxx.se/

to servers. In contrast to a browser, custom HTTP headers can be set - something needed for example for authentication.

Suppose there is a contineo account *user* with the password *secret* and contineo is deployed on the following endpoint *http://localhost:8080/contineo/*. To get the content of the root folder one has to issue the following command:

```
curl -H "X-username:user" -H "X-password:secret"
http://localhost:8080/contineo/rest/folder/
```

An XML file will be returned as answer representing the content of the root folder. The root folder might contain other sub-folders. The XML file returned before might contain such an entry:

```
<url>http://localhost:8080/contineo/rest/folder/31</url>
```

Again, curl can be used to get the content of this sub-folder:

```
curl -H "X-username:user" -H "X-password:secret"
http://localhost:8080/contineo/rest/folder/31
```

A folder might also contain a document. To download the document you can issue:

```
curl -H "X-username:user" -H "X-password:secret"
http://localhost:8080/contineo/rest/document/32
```

Please note, curl will output the content of the document directly on the shell. This is ok for ASCII file like a txt or html file, but it is not a good idea for binary files. In case you download a binary file, you should redirect the output to a file doing the following:

```
curl -H "X-username:user" -H "X-password:secret"
http://localhost:8080/contineo/rest/document/32 > output.zip
```

Of course contineo also returns the filename as it is stored in contineo. This information is contained in the HTTP headers also returned by contineo. The HTTP headers can be extracted using the *-D filename* option of curl:

```
curl -H "X-username:user" -H "X-password:secret" -D header.txt
http://localhost:8080/contineo/rest/document/32 > output.zip
```

The headers returned will contain the following line:

```
Content-disposition: attachment;filename=checked.zip
```

So it can be seen that the filename is actually *checked.zip* and not *output.zip*.

To create a new folder in the root folder, one has to first prepare an XML file describing the new folder. You can find an example in section 4.1 on page 16. Save this file in the directory where you are also executing the curl commands. To create this folder as a sub-folder beneath the root folder in contineo, use the following curl command:

```
curl -H "X-username:user" -H "X-password:secret" -D header.txt
-X PUT -F "contineoCheckInInfo.xml=@contineoCheckInInfo.xml"
http://localhost:8080/contineo/rest/folder/5
```

If creating the folder was successful, contineo returns an XML message containing the ID of the new folder and the URL to access it. Uploading a new document works similar as the example above. The only difference is that besides the *contineoCheckInInfo.xml* also the document must be attached to the request by using again curl's *-F* option.

This section is only meant to contain examples of using curl to query contineo's REST interface. The examples above contain almost all import combinations of curl's options so that each request can be made. In case you want to delete a resource, use the example for getting the content but add the option *-X DELETE*. If you do not want to upload a new resource but instead alter an existing one, you have to send a HTTP POST. In that case take a look at the example for creating a folder, but remove the *-X PUT* option.

The curl tool is also used in the unit tests for contineo. Those tests can be found in the Subversion source code repository of contineo[9].

6 Conclusions

A REST interface for application integration is a powerful and lightweight solution. This document specifies the REST interface for the document management system contineo. The general concept, the use of HTTP status codes, authentication and the different interfaces are described in detail. The message exchange is based on XML. Therefore for each exchanged message an XML schema file is provided for validation.

References

[CCMW01] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web service description language (wsdl) 1.1. Technical report, W3 Consortium, March 2001. http://www.w3.org/TR/wsdl.

[9] http://svn.sourceforge.net/viewvc/contineo/contineo/trunk/unittests/rest-api/

[FGM+99] Roy Thomas Fielding, J. Gettys, J. Mogul, Henrik Frystyk, L. Masinter, P. Leach, and Tim Berners-Lee. Hypertext transfer protocol - http/1.1. Technical report, W3 Consortium, June 1999. http://www.w3.org/Protocols/rfc2616/rfc2616.html.

[Fie00] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures.* PhD thesis, University of California, Irvine, USA, 2000.

[GHM+03] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, and Henrik Frystyk. Soap version 1.2 part 1: Messaging framework. Technical report, W3 Consortium, June 2003. http://www.w3.org/TR/soap12-part1/.