Dokumentacja programistycza korzystania z REST API

Wstęp

Poniższa dokumentacja opisuje sposób komunikacy z użyciem REST API. Dokumentacja ta opisuje wszystkie aspekty począwszy od najbardziej podstawowych takich jak serializacja i deserializacja danych a skończywszy na zaawansowanych tematach takich jak wyszukiwanie dokumentów w archiwum.

Dokumentacja ta nie stanowi samodzielnej dokumentacji do korzystania z Contman Director przez API. Jest to instrukcja jak korzystac z głównej dokumentacji opisującej wszystkie dostępne interfejsy w systemie - dokument zatytułowany "CONTMAN DIRECTOR API Reference".

Dostęp do obiektów.

Całe API REST dostępne jest pod adresem: `http://<adres serwera CD3>/api/`. Wszystkie obiekty opisywane przez dokumentację “CONTMAN DIRECTOR API Reference” mają swoje ścieżki dostępu względem tego adresu. Część obiektów jest dostępna poprzez “dobrze znane” ścieżki, które w dokumentacji referencyjnej są oznaczane jako “Provider URL part”. I tak, dla przykładu, dla obiektu `Categories` jego URL part jest podany jako “`global/categories`”, zatem obiekt ten jest dostępny przez API REST pod adresem `http://example.org/api/global/categories`. Część obiektów nie posiada w dokumentacji referencyjnej swojej “Provider URL part” - oznacza to, że obiekt należy najpierw odszukać poprzez różne metody, żeby poznać jego URL.

Każdy obiekt który zostanie pobrany przez API REST w XMLu będzie posiadał atrybut '`url`', reprezentujący bezpośredni dostęp do niego, oraz niektóre obiekty będą posiadały atrybut '`id`', będący unikalnym identyfikatorem tego obiektu w systemie. Należy pamiętać, że nie zawsze URL do obiektu jest niezmienialny. Dla przykładu, URL do obiektu reprezentującego Domenę '`System`', ma postać `http://example.org/api/global/domains/System`. Jednakże jeśli użytkownik zmieni nazwę domeny, to i URL zmieni swoją postać. Gwarantowane i niezmienialne adresy URL są tylko dla obiektów które w dokumentacji mają podane "Provider URL part".

Serializacja/deserializacja typów danych.

API REST wysyła i odbiera dokumenty XML, w których typy są serializowane zgodnie z następującymi zasadami:

1. Słowniki:

    ```
    <dictionary>
      <item key="klucz1">wartosc1</item>
      <klucz2>wartosc2</klucz2>
    </dictionary>
    ```

    poszczególne klucze ze słownika mogą być serializowane dwojako: albo jako węzeł o nazwie '`item`', z atrybutem '`key`' o wartości klucza (jeśli klucz zawiera znaki niepoprawne w XMLu jako nazwa węzła ('`tagName`', np. spacje w nazwie klucza), lub jako węzeł o nazwie odpowiadającej wartości klucza.
2. Listy:

```
<list>
  <element>wartosc1</element>
  <element>wartosc2</element>
</list>
```

3. Daty – wysyłane i odbierane są w formacie ISO-8601, czyli dla daty z czasem jest to format `yyyy-MM-dd'T'HH:mm:ss`, dla samej daty jako `yyyy-MM-dd`. Formaty te zgodne są ze standardem Javy.
4. Obiekty – dla ułatwienia, obiekty które posiadają swój identyfikator (atrybut '`id`' w XML), można wysyłać podając w miejsce obiektu jego numer, zatem przykładowo lista kategorii mogłaby wyglądać następująco:

```
<list>
  <element>33</element>
  <element>67</element>
</list>
```

Drugim sposobem (także tym, w jakim zwracane są obiekty z serwera), jest podanie nazwy obiektu jako nazwę węzła, oraz jako atrybuty podane przede wszystkim jego '`url`'. Serwer zwracając obiekt uzupełni go także dodatkowymi atrybutami, jednakże podczas wysyłania do serwera są one ignorowane. Przykładowo, lista kategorii mogła by także wyglądać następująco:

```
<list>
  <element><Category url="..."/></element>
  <element><Category url="..."/></element>
</list>
```

Należy pamiętać, że drugi typ wysyłanych obiektów z pewnych ograniczeń nie zawsze jest poprawnie obsługiwany na serwerze. W szczególności, ten format nie jest obsługiwany, jeśli typ parametru metody jest '`any`'. Dla przykładu metoda '`run`' z obiektu '`Script`' przymuje słownik mapowań `string => any`. W takim przypadku należałoby wysłać słownik, z elementami o wartościach numerycznych:

```
<dictionary>
  <klucz1>33</klucz1>
  <klucz2>67</klucz2>
</dictionary>
```

5. Ciągi znaków, liczby całkowite, zmiennoprzecinkowe – wysyłane są jako tekst
6. Wartości logiczne – jako tekst. Wysyłać do serwera można w jednym z wybranych formatów: `true/false, 0/1, yes/no`, bez rozróżniania wielkości znaków, serwer natomiast zawsze zwraca wartości logiczne jako `True/False` (z dużej litery).

Wywoływanie metod oraz przekazywanie parametrów

Posiadając pełen URL do obiektu, można wywołać na nim jedną z dowolnych metod opisanych w dokumentacji referencyjnej, dodając do URLa parametr żądania o nazwie '`call`'. Dla przykładu, aby pobrać listę wszystkich kategorii dostępnych w systemie, należy wywołać metodę '`getAll`' na obiecie '`Categories`'. Można to uczynić przy pomocy następujacego URLa:

```
http://example.org/api/global/categories?call=getAll
```

Jeśli metoda wymaga parametrów, można je podać na dwa sposoby. Pierwszy jest możliwy, o ile parametry są albo wartościami skalarnymi, albo obiektami. Drugi sposób należy wykorzystywać jeśli wysyłane są typy złożone – listy lub słowniki.

Pierwszy sposób polega na podaniu pozostałych parametrów jako kolejnych parametrów żądania URL. Dla przykładu, można wywołać metodę '`get`' z obiektu '`Categories`' wywołując następujący adres URL:

```
http://example.org/api/global/categories?call=get&id=33
```

Drugi sposób polega, na wysłaniu poprzez HTTP POST na serwer XMLa w treści żądania w formacie opisanym poniżej, ustawiając na dodatek nagłówek HTTP '`Content-Type`' na wartość '`text/xml; charset=UTF-8`'. Wysyłany XML, musi być słownikiem o nazwie '`parameteres`', klucze w tym słowniku są nazwami parametrów, natomiast wartościami są kolejne dane wysyłane na serwer. Dla przykładu, wywołanie metody '`search`' z obiektu `CategorySearch` wyglądałoby następująco:

URL: `http://example.org/api/global/search/categories?call=search`
POST:

```
<parameters>
 <selectAttributes>
  <list>
   <element>id</element>
   <element>name</element>
  </list>
 </selectAttributes>
 <criteria>
  <dictionary>
   <id>40</id>
  </dictionary>
 </criteria>
 <distinct>yes</distinct>
</parameters>
```

Należy zauważyć że nazwa metody zawsze przekazywana jest w URL'u, niektóre z parametrów są opcjonalne, oraz że kolejność parametrów nie ma znaczenia.

Wymagane jest obsługiwanie cookie, tzn cookie zwracane przez serwer należy dołączać w kolejnych żądaniach, w szczególności dotyczy to cookie o nazwie "coreServerName".

Autoryzacja

Aby zalogować się do serwera, nalezy wykonać żądanie POST na adres `http://example.org/api/login` wysyłając parametry: `login`, `password` oraz `domain`. Przykładowy POST:

```
<parameters>
  <login>administrator</login>
  <password>haslo</password>
  <domain>System</domain>
</parameters>
```

W odpowiedzi zostanie zwrócony obiekt '`Avatar`', z którego należy odczytać atrybut '`token`', oraz należy dodawać go do każdego następnego żądania, jako nagłówek HTTP o nazwie '`Authorization`' o wartości '`cd3session <token>`', gdzie zamiast '`<token>`' należy wstawiać odczytanego wcześniej tokena. Aby wylogować się, należy wywołać metodę '`logout`' na adres Avatara odczytanego podczas zalogowania się.

Wyszukiwanie w API

Dostępne jest wydajne wyszukiwanie w obiektach i dokumentach poprzez API. Służą do tego obiekty np. `DocumentSearch`, `CategorySearch`, `DomainSearch` itp. Posiadają one główną metodę '`search`', w której najistotniejsze parametry to '`selectAttributes`' oraz '`criteria`'.

Należy podkreślić, ze całe API dotyczące wyszukiwania wymaga podawania identyfikatorów obiektów

Posługiwanie się selectAttributes

Parametr '`selectAttributes`' określa, jakie atrybuty z odpowiadającego wyszukiwanego obiektu zostaną zwrócone. Dla przykładu, podczas wyszukiwania w użytkownikach (`UserSearch`) można podawać takie atrybuty jak np. '`domain`', '`firstName`', '`lastName`', '`login`'. Podczas

wyszukiwania w dokumentach (`DocumentSearch`) poza takim atrybutem jak '`id`', można podawać specjalne atrybuty, w postaci identyfikatorów kategori oraz identyfikatorów indeksów do jakich ten dokument należy. Dla przykładu, przekazując jako '`selectAttributes`' takie wartości jak '`id`', '`123/456`', '`123/789`', '`45/67`', zwrócone wyniki będą zawierały identyfikato dokumentu, oraz wartości indeksów 456 i 789 z kategori 123, oraz wartość indeksu 67 z kategorii 45.

Posługiwanie się criteria

Parametr '`criteria`' odpowiada za warunki nałożone na wyszukiwanie. Można go podać na dwa sposoby. Pierwszy, prosty, wymaga podania słownika, gdzie kluczami są atrybuty, na których będą nakładane warunki równości z przekazanych wartości. Przykładowy słownik: `<dictionary><name>Ala</name></dictionary>` oznacza wyszukiwanie takich obiektów, których atrybut '`name`' jest równy '`Ala`'. Ponieważ można tutaj podawać atrybuty to również można podawać identyfikatory kategori i indeksów dla wyszukiwania w dokumentach.

Drugi sposób przekazywania parametru '`criteria`' polega na przekazaniu operatora wyszukiwania. Operatory są takie jak `AND/OR/>/</==/LIKE/STARTSWITH`. Ich konstrukcja w XMLu wygląda następująco:

```
<operator type="or">
 <argument>...</argument>
 ...
</operator>
```

W zależności od typu, operatory przymują różne ilość argumentów. Kolejność argumentów ma znaczenie. Operatory `AND/OR` mogą się w sobie zagnieżdzać. Wartością argumentu w operatorze jest albo wartość skalarna, albo węzeł `<attribute name="XXX"/>` jeśli argument odwołuje się do atrybutu wyszukiwanego obiektu (również działa zasada dla podawania id kategori/id indeksu dla dokumentów).

Poniżej wymienione są wszystkie operatory:

- Operatory dwuargumentowe: `==, <, >, !=, startswith, like`
- Operatory trójargumentowe: `between`
- Operatory n-argumentowe: `or`, `and`

Przykład dla warunku `(100 < id < 500 AND '123/456' LIKE 'ala%') OR (id > 1000)`:

```
<operator type="or">
  <argument>
    <operator type="and">
      <argument>
        <operator type="between">
          <argument><attribute name="id"/></argument>
          <argument>100</argument>
          <argument>500</argument>
        </operator>
      </argument>
      <argument>
        <operator type="like">
          <argument><attribute name="123/456"/></argument>
          <argument>ala%</argument>
        </operator>
      </argument>
    </operator>
  </argument>
  <argument>
    <operator type="&amp;gt;">
      <argument><attribute name="id"/></argument>
      <argument>1000</argument>
    </operator>
  </argument>
</operator>
```

Wyszukiwanie przez API zwraca obiekt `SearchResults`, który służy dopiero do pobierania wyników, stronicowania, oraz co bardzo istotne – do

zamykania wyników wyszukiwania. `SearchResults` zwraca listę krotek z wartościami skalarnymi, nie zaś wyszukane obiekty.

Szkielet komunikacyjny

Jako referencja, dostarczony jest przykładowy szkielet komunikacyjny który prezentuje prosty model komunikacji z serwerem CD3. Wykorzystuje on do działania bibliotekę JDom, która ułatwia przetwarzanie XML, niemniej można spokojnie pozbyć się zależności od niej. Poniżej przykładowy sposób korzystania:

```
RestConnection conn = new RestConnection("http://example.org/api");
String out = conn.request("POST", "login", "<parameters>...</parameters>");
Element loginRoot = conn.getRoot(out);
String token = loginRoot.getAttributeValue("token");
conn.setAuthToken(token);

conn.call("global/categories","getAll");
...
conn.call("avatar", "logout");
```

Przykłady

Wyszukiwanie dokumentów

Chcemy wyszukać dokumenty które znajdują cię w kategorii "Test" na podstawie wartości indeksów Idx_tekst_1 oraz Idx_data_1. Idx_tekst jest typu tekst a Idx_data_1 jest typu data.

1. Logujemy się do systemu.
   Aby zalogować się do serwera, należy wykonać żądanie POST na adres http://example.org/api/login dodając nagłówek Content-Type o wartości text/xml oraz w treści wysyłając:

   ```
   <parameters>
     <login>administrator</login>
     <password>haslo</password>
     <domain>System</domain>
   </parameters>
   ```

   W odpowiedzi zostanie zwrócony obiekt '`Avatar`', z którego należy odczytać atrybut '`token`', oraz należy dodawać go do każdego następnego żądania, jako nagłówek HTTP o nazwie '`Authorization`' o wartości '`cd3session <token>`', gdzie zamiast '`<token>`' należy wstawiać odczytanego wcześniej tokena.

2. Do wyszukania dokumentów musimy znać id kategorii oraz id indeksu.
   W celu pobrania id kategorii Test należy wykonać żądanie POST: http://example.org/api/global/categories/?call=getResource&path=Test w odpowiedzi zostanie zwrócony obiekt '`Category`' w postaci:

   ```
   <?xml version='1.0' encoding='utf-8'?>
   <Category id="262" immutable="False" name="Test" url="http://example.org/api/global/categories/Test"/>
   ```

   Ze zwróconego obiektu 'Category' odczytujemy atrybut 'id'. Dodatkowo na obiekcie 'Category' można wywołać dowolne metody dostępne przez API opisane w dokumencie "CONTMAN DIRECTOR API Reference" np: http://darek-komputer/api/global/categories/Test?call=getIndexCount
   Możemy również pobrać wszystkie kategorie wysyłając żądanie POST: http://example.org/api/global/categories/?call=getAll w odpowiedzi

otrzymamy listę wszystkich kategorii.
W celu pobrania id indeksy należy wysłać żądanie POST w postaci: http://darek-komputer/api/global/indexes/definitions?call=getResource&path=Idx_tekst_1 w odpowiedzi otrzymamy obiekt 'IndexDefinition' w postaci:

```
<IndexDefinition arity="multiple" id="254" immutable="False" indexTypeId="57" name="Idx_tekst_1"
requiredNotNull="False" url="http://example.org/api/global/indexes/definitions/Idx_tekst_1"/>
```

Podobnie jak dla kategorii odczytujemy atrybut 'id'.
Powyższą czynność powtarzamy dla indeksu Idx_data_1. W naszym przykładzie id tego indeksu jest równe 257

3. Mając potrzebne informacje możemy skonstruować request do wyszukania dokumentów.
W naszym przykładzie będziemy wyszukiwać dokumenty, dla których indeks Idx_tekst_1 będzie równy '1234' oraz indeks Idx_data_1 będzie większy niż '2014-03-15'. W wynikach wyszukiwania będziemy chcieli otrzymać id dokumentu oraz wartość indeksu Idx_data_1.
Z poprzednich krokach uzyskaliśmy informację o id interesujących nas obiektów a mianowicie:
- kategoria Test - id = 262
- indeks Idx_tekst_1 - id = 254
- indeks Idx_data_1 - id = 257
W celu wyszukania dokumentów wysyłamy żądanie POST w postaci http://example.org/api/global/search/documents?call=search. Do żądania oprócz standardowego nagłówka '`Authorization`' o wartości '`cd3session <token>`' dodajemy również nagłówek 'Content-Type' o wartości 'text/xml'. W treści żądania przekazujemy poniższego xmla:

```
<?xml version='1.0' encoding='utf-8'?>
<parameters>
<selectAttributes> <!--Ta sekcja określa, jakie atrybuty z odpowiadającego wyszukiwanego obiektu zostaną
zwrócone-->
  <list>
 <element>id</element> <!--W naszym przykładzie będzie zwrócone id dokumentu-->
 <element>262/257</element>  <!-- id kategorii/id indeksu w naszym przykładzie będzie indeks Idx_data_1 w
kategorii Test -->
  </list>
</selectAttributes>
  <criteria> <!-- w tej sekcja odpowiada za warunki nałożone na wyszukiwanie w naszym przykładzie
("262/254" == 1234 or "262/257" > 2013-03-15 00:00:00) -->
   <operator type="or">
   <argument>
  <operator type= "==">
    <argument>
      <attribute name="262/254" />  <!-- id kategorii/id indeksu w naszym przykładzie będzie indeks Idx_tekst_1
w kategorii Test -->
    </argument>
    <argument>1234</argument> <!--wyszukiwana wartość indeksu-->
  </operator>
   </argument>
   <argument>
  <operator type= "&gt;">
    <argument>
      <attribute name="262/257" /> <!-- id kategorii/id indeksu w naszym przykładzie będzie indeks Idx_data_1
w kategorii Test -->
    </argument>
    <argument>2013-03-15 00:00:00</argument> <!--wyszukiwana wartość indeksu-->
  </operator>
   </argument>
 </operator>
  </criteria>
</parameters>
```

4. W odpowiedzi z żądanie z punktu 3 zwrócony zostanie obiekt SearchResults. Przykładowa odpowiedź wygląda następująco:

```
<DocumentSearchResults url="http://example.org/api/avatar/transient/44"/>
```

Na tym obiekcie musimy wywołać metodę getAll. W tym celu ze zwróconego obiektu odczytujemy atrybut url i wysyłamy żądanie POST w postaci http://example.org/api/avatar/transient/44?call=getAll. W odpowiedzi otrzymamy listę wyszukanych dokumentów w postaci:

```
<list>
  <list>
    <element>3171</element>
    <date format="ISO8601">2014-03-20</date>
  </list>
  <list>
    <element>3172</element>
    <date format="ISO8601">2014-03-18</date>
  </list>
  <list>
    <element>3173</element>
    <date format="ISO8601">2014-03-21</date>
  </list>
</list>
```

Po odczytaniu wyników wyszukiwania na obiekcie SearchResults musimy wywołać metodę close. Metoda ta usuwa obiekt SearchResults. W tym celu wysyłamy żądanie POST w postaci: http://example.org/api/avatar/transient/44?call=close.

5. W celu wylogowania wysyłamy żądanie POST w postaci: http://example.org/api/avatar?call=logout

Zasilanie archiwum

1. Logujemy się do systemu.
   Aby zalogować się do serwera, należy wykonać żądanie POST na adres http://example.org/api/login dodając nagłówek Content-Type o wartości text/xml oraz w treści wysyłając:

   ```
   <parameters>
     <login>administrator</login>
     <password>haslo</password>
     <domain>System</domain>
   </parameters>
   ```

   W odpowiedzi zostanie zwrócony obiekt '`Avatar`', z którego należy odczytać atrybut '`token`', oraz należy dodawać go do każdego następnego żądania, jako nagłówek HTTP o nazwie '`Authorization`' o wartości '`cd3session <token>`', gdzie zamiast '`<token>`' należy wstawiać odczytanego wcześniej tokena.
2. Pobieramy z systemu url klasy do której chcemy zasilić dokument. Klasa dokumentu określa gdzie fizycznie plik zostanie zapisany. W tym celu wysyłamy żądanie POST http://example.org/api/global/classes?call=getAll.W odpowiedzi otrzymamy listę klas zdefiniowanych w systemie. W naszym przykładzie będziemy posługiwali się klasą System.

```
<list>
 <DocumentClass id="53" immutable="True" name="System"
url="http://example.org/api/global/classes/System"/>
 <DocumentClass id="54" immutable="True" name="Temporary"
url="http://example.org/api/global/classes/Temporary"/>
</list>
```

3. Aby utworzyć dokument na obiekcie DocumentClass wywołujemy metodę create. W tym celu wykonujemy żądanie POST:http://example.org/api/global/classes/System?call=create. W odpowiedzi otrzymamy obiekt Dokument w postaci:

```
<?xml version='1.0' encoding='utf-8'?>
<Document id="99" url="http://<host>/api/documents/99"/>
```

4. Aby dodać zestaw wartości indeksów do kategorii na dokumencie wykonujemy metodę addCategoryValues. W dokumencie "CONTMAN DIRECTOR API Reference" opisana jest sygnatura metody addCategoryValues. Wykonujemy żądanie POST w postaci http://example.org/api/documents/99?call=addCategoryValues. Do żądania oprócz standardowego nagłówka '`Authorization`' o wartości '`cd3session <token>`' dodajemy również nagłówek 'Content-Type' o wartości 'text/xml'. W treści żądania przekazujemy poniższego xmla:

```
<?xml version='1.0' encoding='utf-8'?>
<parameters>
 <category>262</category>  <!-- tutaj podajemy Id kategorii do której dodajemy zestaw indeksów-->
 <values>
  <dictionary>
   <item key="254">DKREST</item> <!--jako klucz podajemy id indeksu >
   <item key="257"> <!--id indeksu-->
    <date format="ISO8601">2013-11-05</date> <!--wartość indeksu-->
   </item>
  </dictionary>
 </values>
</parameters>
```

5. Wysyłamy do archiwum plik. W tym celu wysyłamy żądanie
http://<host>/api/documents/99?call=uploadFile&sizehint=100&filename=test&mime=application/pdf
gdzie: sizehint - oznacza wielkość pliku
filename - nazwa pliku (opcjonalne)
mime - typ pliku (opcjonalne)
w odpowiedzi uzyskamy:

```
<?xml version='1.0' encoding='utf-8'?>
<TransferRequest failed="False" finished="False" transferUrl="http://darek-komputer/files/upload/r1"
url="http://darek-komputer/api/avatar/transient/32"/>
```

następnie na adres transferURL (http://darek-komputer/files/upload/r1) wysyłamy zawartość pliku żądaniem Http PUT