## meSPEECH API

This document details the requirements of the meSPEECH API. Note the following key concepts:

 a VOCABULARY consists of one or more selection sets. It encompassess all the concepts or representations the user is able to select from.

 a SELECTION SET is a collection of one or more items. It determines WHAT is displayed.

 a LAYOUT defines how the page is presented (normally in the form of a grid). It determines

HOW the selection set is displayed.

 a THEME defines how the page appears. It determines how the selection set LOOKS.

A selection set and layout will typically be XML. A theme will typically be CSS.

Calls are made to the api for data that may be displayed on a number of devices including a web browser or smart phone app. A smart phone app will principally be used to display the selection set with a Vocabulary Designer being used on a web browser on a larger device to determine the contents of the vocabulary and its selection sets. A subset of the apis may be used on the smart phone app to manage users for example, but access will not be provided to the Vocabulary Designer. This must be used on a web browser.

User and Vocabulary Management

Users may or not use the speech interface. They may be a speech and language therapist or other clinician who manages more than one user (who would probably use the speech interface).

Contents

[meSPEECH API 1](#_Toc321991799)

[Working with the API 5](#_Toc321991800)

[Testing tools 6](#_Toc321991801)

[cURL 6](#_Toc321991802)

[HttpRequester add-on for FireFox 6](#_Toc321991803)

[Authentication 7](#_Toc321991804)

[Authenticating with the API and retrieving a token 7](#_Toc321991805)

[Example authentication request to /api/auth using cURL 8](#_Toc321991806)

[Image 10](#_Toc321991807)

[Uploading an image to the web-server 10](#_Toc321991808)

[Example image upload to /api/image using cURL 10](#_Toc321991809)

[Managed Users 12](#_Toc321991810)

[Getting a list of managed users 12](#_Toc321991811)

[User 13](#_Toc321991812)

[Getting the details of a user 13](#_Toc321991813)

[An example of the same request made with cURL 13](#_Toc321991814)

[Adding a new user 13](#_Toc321991815)

[Updating a user 14](#_Toc321991816)

[Deleting a user 15](#_Toc321991817)

[User Vocabularies 16](#_Toc321991818)

[Getting the link entry by user-id and vacabulary-id 16](#_Toc321991819)

[Getting a list of vocabularies for a user 17](#_Toc321991820)

[Assign a Vocabulary to a User 17](#_Toc321991821)

[Removing Vocabulary from a User 18](#_Toc321991822)

[Vocabularies 19](#_Toc321991823)

[Getting a list of vocabularies created by authenticated user 19](#_Toc321991824)

[Getting the details of a vocabulary 20](#_Toc321991825)

[Adding a new vocabulary 20](#_Toc321991826)

[Updating a vocabulary 21](#_Toc321991827)

[Deleting a vocabulary 21](#_Toc321991828)

[User Themes 22](#_Toc321991829)

[Getting a list of themes for a user 22](#_Toc321991830)

[Public Themes 23](#_Toc321991831)

[Getting a list of public themes 23](#_Toc321991832)

[System Themes 24](#_Toc321991833)

[Getting a list of system themes 24](#_Toc321991834)

[Themes 25](#_Toc321991835)

[Getting the details of a theme 25](#_Toc321991836)

[Adding a new theme 25](#_Toc321991837)

[Updating a theme 26](#_Toc321991838)

[Deleting a theme 26](#_Toc321991839)

[User Layouts 28](#_Toc321991840)

[Getting a list of layouts for a user 28](#_Toc321991841)

[Public Layouts 29](#_Toc321991842)

[Getting a list of public layouts 29](#_Toc321991843)

[System Layouts 30](#_Toc321991844)

[Getting a list of system layouts 30](#_Toc321991845)

[Layouts 31](#_Toc321991846)

[Getting the details of a layout 31](#_Toc321991847)

[Adding a new layout 31](#_Toc321991848)

[Updating a layout 32](#_Toc321991849)

[Deleting a layout 32](#_Toc321991850)

[Copy Vocabulary 33](#_Toc321991851)

[Copying a public or system vocabulary 33](#_Toc321991852)

[Representation Search 34](#_Toc321991853)

[Searching the dictionary for representations of concepts 34](#_Toc321991854)

[Selection Sets 35](#_Toc321991855)

[Getting the details of a selection sets by vocabulary-id 35](#_Toc321991856)

[Getting the details of a selection set 36](#_Toc321991857)

[Adding a new selection set 36](#_Toc321991858)

[Updating a selection set 37](#_Toc321991859)

[Deleting a selection set 37](#_Toc321991860)

[Selection Set Items 38](#_Toc321991861)

[Getting the list of a items by selectionset-id 38](#_Toc321991862)

[Adding a new item 39](#_Toc321991863)

[Deleting an item 39](#_Toc321991864)

[Managing Selection Sets 40](#_Toc321991865)

[Getting the link entry by selectionset-id and vacabulary-id 40](#_Toc321991866)

[Assign a Selection Set to a vocabulary 40](#_Toc321991867)

[Removing Selection Set from a vocabulary 41](#_Toc321991868)

## Working with the API

All requests must contain your API key in the HTTP headers.

The **/api/auth** section of the API requires only the “apikey” header to be present in the request. All other parts of the API also require both “apikey” and “token” be supplied in the header.

The following example shows how to use jQuery to attach your API key into the HTTP headers, POST user credentials to the API to authenticate and extract the returned token from the response which will be required for accessing other parts of the API. Errors are captured inside the ***error*** section and logged to the console.

|  |
| --- |
| <script >  $.ajax({  headers: {  'apikey': 'my-api-key-123'  },  url: http://www.domain.com/api/auth',  type: 'POST',  data: {  data: '<user><username>PaulH</username><password>Passw0rd</password></user>'  },  success: function(xml) {  var token = $(xml).find('token').text(); // Store token for subsequent requests  console.log("Hello username: " + $(xml).find('username').text());  console.log("Token: " + token);  },  error: function(xhr) {  xml = xhr.responseXML; // complete XmlHttpRequest object returned  $(xml).find('error').each(function() {  console.log ($(this).find('message').text()); // log errors in console  });  }  });  </script> |

## Testing tools

### cURL

<http://curl.haxx.se/>

Use cURL to simulate client requests to the API. Note: To change the cURL request method, use the -X flag (e.g. curl -X PUT …)

### HttpRequester add-on for FireFox

<https://addons.mozilla.org/en-US/firefox/addon/httprequester/>

This tool is useful when doing web or REST development, or when you need to make HTTP requests such as (PUT/POST/DELETE) via the browser. The add-on keeps a history of the requests and allows you to repeat them via the GUI.

## Authentication

### Authenticating with the API and retrieving a token

**POST /api/auth**

Authenticating with the API requires username/password to be passed in XML format. A successful request will return a user entity which also includes a security token that needs to be appended to the HTTP headers when accessing other parts of the API.

**Request**

|  |
| --- |
| <user>  <username>PaulH</username>  <password>Passw0rd</password>  </user> |

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |
| 400 | Bad request, check response XML for specific error details |
| 404 | Bad credentials, check username/password is valid |

**Response**

|  |
| --- |
| <user>  <id>2</id>  <name>Paul</name>  <surname>Hewett</surname>  <username>PaulH</username>  <mobile>07779 555 5555</mobile>  <email>user@example.com</email>  <gender>M</gender>  <image>/path/to/image.jpg</image>  <year\_of\_birth>1975</year\_of\_birth>  <default\_page>user</default\_page>  <default\_symbolset>mby</default\_symbolset>  <created\_by>1</created\_by>  <token>4e7e332faff7e560f698e0dc25f485821b6a1a57</token>  <created\_date>2011-12-01 14:24:32</created\_date>  <updated\_date>2011-12-01 14:24:32</updated\_date>  </user> |

### Example authentication request to /api/auth using cURL

**Request made from command-line**

|  |
| --- |
| curl --verbose --header apikey:my-api-key-123 --data "data=<user><username>PaulH</username><password>Passw0rd</password></user>" http://www.domain.com/api/auth |

**Response**

|  |
| --- |
| \* About to connect() to www.domain.com port 80 (#0)  \* Trying 10.0.0.41... connected  \* Connected to www.domain.com (10.0.0.41) port 80 (#0)  > POST /api/auth HTTP/1.1  > User-Agent: curl/7.21.4 (i386-pc-win32) libcurl/7.21.4 zlib/1.2.5  > Host: www.domain.com:80  > Accept: \*/\*  > apikey:asdqwe123  > Content-Length: 75  > Content-Type: application/x-www-form-urlencoded  >  < HTTP/1.1 202 Accepted  < Date: Tue, 17 Jan 2012 16:51:36 GMT  < Server: Apache/2.2.17 (Win32) mod\_ssl/2.2.17 OpenSSL/0.9.8o PHP/5.2.16  < X-Powered-By: PHP/5.2.16  < Set-Cookie: PHPSESSID=sv7t6q9nlp9lnc7drsjqoehlb4; path=/  < Expires: Thu, 19 Nov 1981 08:52:00 GMT  < Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0  < Pragma: no-cache  < Content-Length: 555  < Content-Type: text/xml  <  <?xml version="1.0" encoding="UTF-8"?>  <user>  <id>2</id>  <name>Paul</name>  <surname>Hewett</surname>  <username>PaulH</username>  <mobile>07779 555 5555</mobile>  <email>user@example.com</email>  <gender>M</gender>  <image>/path/to/image.jpg</image>  <year\_of\_birth>1975</year\_of\_birth>  <default\_page>user</default\_page>  <default\_symbolset>mby</default\_symbolset>  <created\_by>1</created\_by>  <token>4e7e332faff7e560f698e0dc25f485821b6a1a57</token>  <created\_date>2011-12-01 14:24:32</created\_date>  <updated\_date>2011-12-01 14:24:32</updated\_date>  </user>  \* Connection #0 to host [www.domain.com](http://www.domain.com) left intact  \* Closing connection #0 |

## Image

### Uploading an image to the web-server

**POST /api/image/**

Upload an image to the web-server and retrieve the URL of the remote file. The URL can then be used to populate the contents of the <image> element within <user> and <theme> entities.

**Images that are not precisely 100x100 pixels in dimension will be cropped and resized by the API.**

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |
| 400 | Bad request, check response XML for specific error |

**Response**

|  |
| --- |
| <image>  <url>http://www.domain.com/path/to/image.jpg</url>  </image> |

### Example image upload to /api/image using cURL

**Request from command-line**

|  |
| --- |
| curl --verbose --header apikey:my-api-key-123 --header token:4e7e332faff7e560f698e0dc25f485821b6a1a57 -F file=@my.jpg http://www.domain.com/api/image |

**Response**

|  |
| --- |
| \* About to connect() to www.domain.com port 80 (#0)  \* Trying 10.0.0.41... connected  \* Connected to www.domain.com (10.0.0.41) port 80 (#0)  > POST /api/image HTTP/1.1  > User-Agent: curl/7.21.4 (i386-pc-win32) libcurl/7.21.4 zlib/1.2.5  > Host: www.domain.com:80  > Accept: \*/\*  > apikey:asdqwe123  > token:4e7e332faff7e560f698e0dc25f485821b6a1a57  > Content-Length: 110839  > Expect: 100-continue  > Content-Type: multipart/form-data; boundary=----------------------------c3b9d5f340fd  >  < HTTP/1.1 100 Continue  < HTTP/1.1 200 OK  < Date: Tue, 17 Jan 2012 17:03:35 GMT  < Server: Apache/2.2.17 (Win32) mod\_ssl/2.2.17 OpenSSL/0.9.8o PHP/5.2.16  < X-Powered-By: PHP/5.2.16  < Content-Length: 115  < Content-Type: text/xml  <  <?xml version="1.0" encoding="UTF-8"?>  <image>  <url>http://www.domain.com/uploads/images/my.jpg</url>  </image>  \* Connection #0 to host www.domain.com left intact  \* Closing connection #0 |

## Managed Users

### Getting a list of managed users

**GET /api/managedusers**

Retrieve a list of managed users that belong to the authenticated user. If no managed users are found, the response will just contain an empty <managedusers> element.

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |

**Response**

|  |
| --- |
| <managedusers>  <user>  <id>2</id>  <name>Paul</name>  <surname>Hewett</surname>  <username>PaulH</username>  <mobile>07779 555 5555</mobile>  <email>user@example.com</email>  <gender>M</gender>  <image>/path/to/image.jpg</image>  <year\_of\_birth>1975</year\_of\_birth>  <default\_page>user</default\_page>  <default\_symbolset>mby</default\_symbolset>  <created\_by>1</created\_by>  <created\_date>2011-12-01 14:24:32</created\_date>  <updated\_date>2011-12-01 14:24:32</updated\_date>  </user>  </managedusers> |

## User

### Getting the details of a user

**GET /api/user/{user-id}**

Retrieve details of a user by user id. A complete entity is displayed below.

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |
| 404 | Not found, check user-id |

**Response**

|  |
| --- |
| <user>  <id>2</id>  <name>Paul</name>  <surname>Hewett</surname>  <username>PaulH</username>  <mobile>07779 555 5555</mobile>  <email>paul@paulhewett.com</email>  <gender>M</gender>  <image>/path/to/image.jpg</image>  <year\_of\_birth>1975</year\_of\_birth>  <default\_page>user</default\_page>  <default\_symbolset>mby</default\_symbolset>  <created\_by>1</created\_by>  <created\_date>2011-12-01 14:24:32</created\_date>  <updated\_date>2011-12-01 14:24:32</updated\_date>  </user> |

### An example of the same request made with cURL

**Request from command-line**

|  |
| --- |
| curl --verbose --header apikey:my-api-key-123 --header token:my-token http://www.domain.com /api/user/user-id-here |

### Adding a new user

**POST /api/user/**

Add a new user. A complete entity with additional metadata is returned after a successful request. To populate <image> element, use **/api/image** to upload a file first.

**Request**

|  |
| --- |
| <user>  <name>New</name>  <surname>User</surname>  <username>NewUser</username>  <password>Passw0rd</password>  <password2>Passw0rd</password2>  <mobile>07779 555 5555</mobile>  <email>user@example.com</email>  <gender>M</gender>  <image>/path/to/image.jpg</image>  <year\_of\_birth>1975</year\_of\_birth>  <default\_page>speech</default\_page>  <default\_symbolset>mby</default\_symbolset>  </user> |

**HTTP Codes**

|  |  |
| --- | --- |
| 201 | Request OK; content created |
| 400 | Bad request, check response XML for specific error details |

### Updating a user

**PUT /api/user/{user-id}**

Update an existing user. A complete entity with additional metadata is returned after a successful request.

**Request**

|  |
| --- |
| <user>  <name>New</name>  <surname>User</surname>  <username>NewUser</username>  <password>Passw0rd</password>  <password2>Passw0rd</password2>  <mobile>07779 555 5555</mobile>  <email>user@example.com</email>  <gender>M</gender>  <year\_of\_birth>1975</year\_of\_birth>  <default\_page>speech</default\_page>  <default\_symbolset>mby</default\_symbolset>  </user> |

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |
| 400 | Bad request, check response XML for specific error details |
| 401 | Access denied; created\_by doesn’t match entity being updated |

### Deleting a user

**DELETE /api/user/{user-id}**

Delete a user. HTTP code 204 (No Content) will be returned on success.

**HTTP Codes**

|  |  |
| --- | --- |
| 204 | Request OK |
| 403 | Forbidden; User has managed users which need to be deleted first |
| 401 | Access denied; created\_by doesn’t match entity being updated |
| 404 | Not found, check user-id |

## User Vocabularies

### Getting the link entry by user-id and vocabulary-id

**GET /api/uservocabularies/?vocabulary\_id={vocabulary-id}&user\_id={** **user-id}**

Retrieve the link entry connecting vocabulary with user.

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |
| 404 | Not found, check user-id andvocabulary-id |

**Response**

|  |
| --- |
| <vocabulary>  <id>1</id>  <user\_id>1</user\_id>  <vocabulary\_id>1</vocabulary\_id>  <start\_date>2011-01-24 23:31:39</start\_date>  <created\_date>2011-01-01 12:00:39</created\_date>  <modified\_date>2011-01-01 12:00:39</modified\_date>  </vocabulary> |

### Getting a list of vocabularies for a user

**GET /api/uservocabularies/{user-id}**

View detail of all vocabularies for a specified user. If no vocabularies are found, the response will just contain an empty <vocabularies> element.

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |

**Response**

|  |
| --- |
| <vocabularies>  <vocabulary>  <id>1</id>  <name>Test Vocabulary 001</name>  <theme\_id>1</theme\_id>  <description>This is a test vocabulary</description>  <type>User</type>  <created\_by>1</created\_by>  <created\_date>2012-01-10 11:18:39</created\_date>  <modified\_date>2012-01-10 11:18:39</modified\_date>  </vocabulary>  </vocabularies> |

### Assign a Vocabulary to a User

**POST /api/uservocabularies /**

Assigns a vocabulary to a user – adds a record in the user\_vocabulary table.

**Request**

|  |
| --- |
| <vocabulary>  <user\_id>1</user\_id>  <vocabulary\_id>1</vocabulary\_id>  <start\_date>2011-01-24 23:31:39</start\_date>  </vocabulary> |

**HTTP Codes**

|  |  |
| --- | --- |
| 201 | Request OK; content created |
| 400 | Bad request, check response XML for specific error details |

### Removing Vocabulary from a User

**DELETE /api/uservocabularies /{linkentry- id}**

Removes the link entry connecting vocabulary with user. HTTP code 204 (No Content) will be returned on success.

**HTTP Codes**

|  |  |
| --- | --- |
| 204 | Request OK |
| 404 | Not found, check linkentry-id |

## Vocabularies

### Getting a list of vocabularies created by authenticated user

**GET /api/vocabularies /**

View detail of all vocabularies created by authenticated user. If no vocabularies are found, the response will just contain an empty <vocabularies> element.

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |

**Response**

|  |
| --- |
| <vocabularies>  <vocabulary>  <id>1</id>  <name>Test Vocabulary 001</name>  <theme\_id>1</theme\_id>  <description>This is a test vocabulary</description>  <type>User</type>  <created\_by>1</created\_by>  <created\_date>2012-01-10 11:18:39</created\_date>  <modified\_date>2012-01-10 11:18:39</modified\_date>  </vocabulary>  </vocabularies> |

### Getting the details of a vocabulary

**GET /api/vocabularies/{vocabulary-id}**

Retrieve details of a vocabulary by vocabulary id. A complete entity is displayed below.

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |
| 404 | Not found, check vocabulary-id |

**Response**

|  |
| --- |
| <vocabulary>  <id>1</id>  <name>Test Vocabulary</name>  <theme\_id>1</theme\_id>  <description><![CDATA[This is a test vocabulary]]></description>  <type>User</type>  <created\_by>1</created\_by>  <created\_date>2012-01-10 11:18:39</created\_date>  <modified\_date>2012-01-10 11:18:39</modified\_date>  </vocabulary> |

### Adding a new vocabulary

**POST /api/vocabularies/**

Add a new vocabulary. A complete entity with additional metadata is returned after a successful request.

**Note: One you have created a vocabulary, you can assign it to a user with ‘Assign a Vocabulary to a User’ on Page 17.**

**Request**

|  |
| --- |
| <vocabulary>  <name>New Vocabulary</name>  <theme\_id>1</theme\_id>  <description><![CDATA[This is a new vocabulary]]></description>  <type>user</type>  </vocabulary> |

**HTTP Codes**

|  |  |
| --- | --- |
| 201 | Request OK |
| 400 | Bad request, check XML response for specific error |

### Updating a vocabulary

**PUT /api/vocabularies /{vocabulary-id}**

Update an existing vocabulary. A complete entity with additional metadata is returned after a successful request.

**Request**

|  |
| --- |
| <vocabulary>  <name>Test Vocabulary</name>  <theme\_id>1</theme\_id>  <description><![CDATA[This is a test vocabulary]]></description>  <type>user</type>  </vocabulary> |

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |
| 400 | Bad request, check response XML for specific error |
| 401 | Access denied; created\_by doesn’t match entity being updated |
| 404 | Not found; check vocabulary-id |

### Deleting a vocabulary

**DELETE /api/vocabularies/{vocabulary-id}**

Delete a vocabulary. HTTP code 204 (No Content) will be returned on success.

**HTTP Codes**

|  |  |
| --- | --- |
| 204 | Request OK |
| 401 | Access denied; created\_by doesn’t match entity being deleted |
| 403 | Request rejected; vocabulary is currently in use |
| 404 | Not found, check vocabulary-id |

## User Themes

### Getting a list of themes for a user

**GET /api/themesuser/**

View the users themes available to the user, excluding system themes, but including their OWN public themes. If no themes are found, the response will just contain an empty <themes> element.

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |

**Response**

|  |
| --- |
| <themes>  <theme>  <id>2</id>  <name>Black and White</name>  <description><![CDATA[This is a high contrast user theme]]></description>  <type>user</type>  <css><![CDATA[body {background-color: #000; color: #fff; } ]]></css>  <image>/path/to/image.jpg</image>  <created\_by>1</created\_by>  <created\_date>2011-02-16 14:00:24</created\_date>  <modified\_date>2011-02-16 14:00:24</modified\_date>  </theme>  </themes> |

## Public Themes

### Getting a list of public themes

**GET /api/themespublic/**

Returns themes that are public and do not belong to the current user. If no themes are found, the response will just contain an empty <themes> element.

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |

**Response**

|  |
| --- |
| <themes>  <theme>  <id>2</id>  <name>Black and White</name>  <description><![CDATA[This is a minimalistic public theme]]></description>  <type>public</type>  <css><![CDATA[body {background-color: #000; color: #fff; } ]]></css>  <image>/path/to/image.jpg</image>  <created\_by>1</created\_by>  <created\_date>2011-02-16 14:00:24</created\_date>  <modified\_date>2011-02-16 14:00:24</modified\_date>  </theme>  </themes |

## System Themes

### Getting a list of system themes

**GET /api/themessystem/**

Return all system themes. If no themes are found, the response will just contain an empty <themes> element.

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |

**Response**

|  |
| --- |
| <themes>  <theme>  <id>3</id>  <name>High Contrast</name>  <description><![CDATA[This is a high contrast system theme]]></description>  <type>system</type>  <css><![CDATA[body {background-color: #000; color: #fff; } ]]></css>  <image>/path/to/image.jpg</image>  <created\_by>1</created\_by>  <created\_date>2011-02-16 14:00:24</created\_date>  <modified\_date>2011-02-16 14:00:24</modified\_date>  </theme>  </themes> |

## Themes

### Getting the details of a theme

**GET /api/themes/{theme-id}**

Retrieve details of a theme by theme id. A complete entity is displayed below.

**Response**

|  |
| --- |
| <theme>  <id>2</id>  <name>Black and White</name>  <description><![CDATA[This is a minimalistic public theme]]></description>  <type>public</type>  <css><![CDATA[body {background-color: #000; color: #fff; } ]]></css>  <image>/path/to/image.jpg</image>  <created\_by>1</created\_by>  <created\_date>2011-02-16 14:00:24</created\_date>  <modified\_date>2011-02-16 14:00:24</modified\_date>  </theme> |

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |
| 404 | Not found, check theme-id |

### Adding a new theme

**POST /api/themes/**

Add a new theme. A complete entity with additional metadata is returned after a successful request. To populate <image> element, use **/api/image** to upload a file first.

**Request**

|  |
| --- |
| <theme>  <name>New Theme</name>  <description><![CDATA[This is a description]]></description>  <type>user</type>  <css><![CDATA[ body { color: red; }]]></css>  <image>/path/to/image.jpg</image>  <created\_by>1</created\_by>  </theme> |

**HTTP Codes**

|  |  |
| --- | --- |
| 201 | Request OK |
| 400 | Bad request, check XML response for specific error |

### Updating a theme

**PUT /api/themes/{theme-id}**

Update an existing theme. A complete entity with additional metadata is returned after a successful request.

**Request**

|  |
| --- |
| <theme>  <name>New Theme</name>  <description><![CDATA[This is a description]]></description>  <type>user</type>  <css><![CDATA[ body { color: blue; }]]></css>  <image>/path/to/image.jpg</image>  <created\_by>1</created\_by>  </theme> |

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |
| 400 | Bad request; check response XML for specific error |
| 401 | Access denied; created\_by doesn’t match entity being updated |
| 404 | Not found; check theme-id |

### Deleting a theme

**DELETE /api/themes/{theme-id}**

Delete a theme. HTTP code 204 (No Content) will be returned on success.

**HTTP Codes**

|  |  |
| --- | --- |
| 204 | Request OK |
| 401 | Access denied; created\_by doesn’t match entity being updated |
| 403 | Forbidden; theme is currently in use |
| 404 | Not found, check theme-id |

## User Layouts

### Getting a list of layouts for a user

**GET /api/layoutsuser/**

View the user’s layouts available to the user, excluding system layouts, but including their OWN public layouts. If no layouts are found, the response will just contain an empty <layouts> element.

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |
| 404 | Not found; check user-id |

**Response**

|  |
| --- |
| <layouts>  <layout>  <id>16</id>  <name>Test Layout</name>  <description><![CDATA[This a test layout]]></description>  <type>user</type>  <xml><![CDATA[Nested XML for rendering the grid]]></xml>  <preview><![CDATA[HTML code representing grid]]></preview>  <created\_by>1</created\_by>  <created\_date>2012-01-16 13:39:35</created\_date>  <modified\_date>2012-01-16 13:39:35</modified\_date>  </layout>  </layouts> |

## Public Layouts

### Getting a list of public layouts

**GET /api/layoutspublic/**

Returns layouts that are public and do not belong to the current user. If no layouts are found, the response will just contain an empty <layouts> element.

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |

**Response**

|  |
| --- |
| <layouts>  <layout>  <id>16</id>  <name>Test Layout</name>  <description><![CDATA[This a test layout]]></description>  <type>public</type>  <xml><![CDATA[Some nested XML for rendering the grid]]></xml>  <preview><![CDATA[Some HTML code representing the layout]]></preview>  <created\_by>1</created\_by>  <created\_date>2012-01-16 13:39:35</created\_date>  <modified\_date>2012-01-16 13:39:35</modified\_date>  </layout>  </layouts> |

## System Layouts

### Getting a list of system layouts

**GET /api/layoutssystem/**

View detail of all system layouts. If no layouts are found, the response will just contain an empty <layouts> element.

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |

**Response**

|  |
| --- |
| <layouts>  <layout>  <id>16</id>  <name>Test Layout</name>  <description><![CDATA[This a test layout]]></description>  <type>system</type>  <xml><![CDATA[Some nested XML for rendering the grid]]></xml>  <preview><![CDATA[Some HTML code representing the layout]]></preview>  <created\_by>1</created\_by>  <created\_date>2012-01-16 13:39:35</created\_date>  <modified\_date>2012-01-16 13:39:35</modified\_date>  </layout>  </layouts> |

## Layouts

### Getting the details of a layout

**GET /api/layouts/{layout-id}**

Retrieve details of a layout by layout id. A complete entity is displayed below.

**Response**

|  |
| --- |
| <layout>  <id>1</id>  <name>3x3 Grid</name>  <description><![CDATA[This a 9 square grid]]></description>  <type>user</type>  <xml><![CDATA[Embedded XML here]]></xml>  <preview><![CDATA[Some HTML code representing the layout]]></preview>  <created\_by>1</created\_by>  <created\_date>2011-02-16 00:00:00</created\_date>  <modified\_date>2011-02-16 00:00:00</modified\_date>  </layout> |

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |
| 404 | Not found, check layout-id |

### Adding a new layout

**POST /api/layouts/**

Add a new layout. A complete entity with additional metadata is returned after a successful request.

**Request**

|  |
| --- |
| <layout>  <name>Test Layout</name>  <description>This a test layout</description>  <type>user</type>  <xml><![CDATA[Embedded XML here]]></xml>  <preview><![CDATA[Some HTML code representing the layout]]></preview>  </layout> |

**HTTP Codes**

|  |  |
| --- | --- |
| 201 | Request OK |
| 400 | Bad request, check response XML for specific error |

### Updating a layout

**PUT /api/layouts/{layout-id}**

Update an existing layout. A complete entity with additional metadata is returned after a successful request.

**Request**

|  |
| --- |
| <layout>  <name>Test Layout</name>  <description><![CDATA[This a test layout that has been changed]]></description>  <type>system</type>  <xml><![CDATA[Some XML here]]></xml>  <preview><![CDATA[Some HTML code representing the layout]]></preview>  </layout> |

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |
| 400 | Bad request, check response XML for specific error |
| 401 | Access denied; created\_by doesn’t match entity being updated |
| 404 | Not found, check layout-id |

### Deleting a layout

**DELETE /api/layouts/{layout-id}**

Delete a layout. HTTP code 204 (No Content) will be returned on success.

**HTTP Codes**

|  |  |
| --- | --- |
| 204 | Request OK |
| 401 | Access denied; created\_by doesn’t match entity being updated |
| 403 | Forbidden; layout is currently in use |
| 404 | Not found, check layout-id |

## Copy Vocabulary

### Copying a public or system vocabulary

**GET /api/copyvocabulary/{vocabulary-id}**

Copy an existing public or system vocabulary before it can be customized. A complete entity with additional metadata is returned after a successful request.

**Request**

|  |
| --- |
| <vocabulary>  <name>Test Layout</name>  **<…>**  </ vocabulary > |

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |
| 401 | Access denied; cannot copy another user’s private vocabulary |
| 404 | Not found, check vocabulary-id |

## Representation Search

### Searching the dictionary for representations of concepts

**GET /api/representation/?tags={tags}&language={pcs}&representation\_type={image|audio|..}**

Return a list of representations and the associated image or other media. If no representations are found, the response will just contain an empty <items> element.

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |
| 400 | Bad request, check response XML for specific error |

**Response**

|  |
| --- |
| <items>  <item>  <id>10543</id>  <code>7BGGG</code>  <sense\_indicator>9999</sense\_indicator>  <part\_of\_speech>unknown</part\_of\_speech>  <lemma><![CDATA[dried apples]]></lemma>  <representation><![CDATA[/path/to/image.jpg]]></representation>  <language>mby</language>  <tags><![CDATA[dried apples;black and white;]]></tags>  </item>  </items> |

## Selection Sets

### Getting selection sets by vocabulary-id

**GET /api/selectionsets/?vocabulary\_id={vocabulary-id}**

Retrieve selection sets that belong to the supplied vocabulary-id. If no selection sets are found, the response will just contain an empty <selectionsets> element.

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |

**Response**

|  |
| --- |
| <selectionsets>  <selectionset>  <id>17</id>  <layout\_id>1</layout\_id>  <name>Test Selection Set 001</name>  <description>This is a test selection set</description>  <created\_date>2011-03-09 07:52:28</created\_date>  <modified\_date>2011-03-09 07:52:28</modified\_date>  </selectionset>  </selectionsets> |

### Getting the details of a single selection set

**GET /api/selectionsets/{selectionset-id}**

Retrieves details of a selection set.

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |
| 404 | Not found, check selectionset-id |

**Response**

|  |
| --- |
| <selectionset>  <id>17</id>  <layout\_id>1</layout\_id>  <name>Test Selection Set 001</name>  <description>This is a test selection set</description>  <created\_date>2011-03-09 07:52:28</created\_date>  <modified\_date>2011-03-09 07:52:28</modified\_date>  </selectionset> |

### Adding a new selection set

**POST /api/selectionsets/**

Adds a new selection set. A complete entity with additional metadata is returned after a successful request.

**Request**

|  |
| --- |
| <selectionset>  <layout\_id>1</layout\_id>  <name>Test Selection Set 001</name>  <description>This is a test selection set</description>  </selectionset> |

**HTTP Codes**

|  |  |
| --- | --- |
| 201 | Request OK; content created |
| 400 | Bad request, check response XML for specific error details |

### Updating a selection set

**PUT /api/selectionsets/{selectionset-id}**

Update an existing selection set. A complete entity with additional metadata is returned after a successful request.

**Request**

|  |
| --- |
| <selectionset>  <layout\_id>1</layout\_id>  <name>Test Selection Set 001</name>  <description>This is a test selection set</description>  </selectionset> |

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |
| 400 | Bad request, check response XML for specific error details |
| 404 | Not found, check selectionset-id |

### Deleting a selection set

**DELETE /api/selectionsets/{selectionset-id}**

Delete an existing selection set. Link entries connecting it to vocabulary and items assigned to it are also removed. HTTP code 204 (No Content) will be returned on success.

**HTTP Codes**

|  |  |
| --- | --- |
| 204 | Request OK |
| 404 | Not found, check selectionset-id |

## Selection Set Items

### Getting a list of a items in a selection set

**GET /api/items/?selectionset\_id={selectionset-id}**

Retrieve items that belong to a selection set by supplied selectionset-id. If no items are found, the response will just contain an empty <items> element.

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |

**Response**

|  |
| --- |
| <items>  <item>  <id>35</id>  <selectionset\_id>3</selectionset\_id>  <representation>/uploads/representation/alphabet.gif</representation>  <gloss><![CDATA[Gloss 002 (Description of representation)]]></gloss>  <enable\_image>1</enable\_image>  <priority>1</priority>  <force\_location>1</force\_location>  <action>openWebPage</action>  <action\_parameter>http://www.google.com/</action\_parameter>  <concept\_code>DXDM5</concept\_code>  <style>black</style>  <created\_date></created\_date>  <modified\_date></modified\_date>  </item>  </items> |

### Adding a new item

**POST /api/items/**

Adds a new item to a selection set. A complete entity with additional metadata is returned after a successful request.

**Request**

|  |
| --- |
| <item>  <selectionset\_id>3</selectionset\_id>  <representation>/uploads/representation/alphabet.gif</representation>  <gloss><![CDATA[Gloss 002 (Description of representation)]]></gloss>  <enable\_image>1</enable\_image>  <priority>1</priority>  <force\_location>1</force\_location>  <action>openWebPage</action>  <action\_parameter>http://www.google.com/</action\_parameter>  <concept\_code>DXDM5</concept\_code>  <style>black</style>  </item> |

**HTTP Codes**

|  |  |
| --- | --- |
| 201 | Request OK; content created |
| 400 | Bad request, check response XML for specific error details |

### Deleting an item

**DELETE /api/items/{item-id}**

Delete an existing selection set item. HTTP code 204 (No Content) will be returned on success.

**HTTP Codes**

|  |  |
| --- | --- |
| 204 | Request OK |
| 404 | Not found, check item-id |

## Managing Selection Sets

### Getting the link entry by selectionset-id and vocabulary-id

**GET /api/vocabularyselectionset/?vocabulary\_id={vocabulary-id}&selectionset\_id={** **selectionset-id}**

Retrieve the link entry connecting selection set with vocabulary.

**HTTP Codes**

|  |  |
| --- | --- |
| 200 | Request OK |
| 404 | Not found, check selectionset-id andvacabulary-id |

**Response**

|  |
| --- |
| <selectionset>  <id>1</id>  <vocabulary\_id>1</vocabulary\_id>  <selectionset\_id>1</selectionset\_id>  <created\_date></created\_date>  <modified\_date></modified\_date>  </selectionset> |

### Assign a Selection Set to a vocabulary

**POST /api/vocabularyselectionset/**

Assigns a selection set to a vocabulary – adds a record in the vocabulary\_selectionset table.

**Request**

|  |
| --- |
| <selectionset>  <vocabulary\_id>1</vocabulary\_id>  <selectionset\_id>1</selectionset\_id>  </selectionset> |

**HTTP Codes**

|  |  |
| --- | --- |
| 201 | Request OK; content created |
| 400 | Bad request, check response XML for specific error details |

### Removing Selection Set from a vocabulary

**DELETE /api/vocabularyselectionset /{linkentry- id}**

Removes the link entry connecting selection set with vocabulary. HTTP code 204 (No Content) will be returned on success.

**HTTP Codes**

|  |  |
| --- | --- |
| 204 | Request OK |
| 404 | Not found, check linkentry-id |