API v2

Customer Documentation

Version: 1.8 - [21.09.2016]

clickworker GmbH
Hatzper Straße 34
D-45149 Essen

Telefon: +49 201 959 718-0
Fax: +49 201 959 718-99
Internet: www.clickworker.com

Geschäftsführung: Christian Rozsenich
AG Essen HRB 22914 – Ust-IdNr. DE246978893

Inhaltsverzeichnis

1 Introduction

clickworker offers a wide range of crowdsourcing services to customers. These services enable companies to easily outsource thousands of manual tasks to the crowd.

Currently we offer three different approaches to benefit from the scalability of our crowd services:

- Full Service delivered by our Solutions department
- Self-Service through our Self-Service-Marketplace
- Self-Service through our REST-based API

This document explains how to apply our REST based API for your crowdsourcing project.

Our Self-Service REST-based API is designed to provide a seamless integration of our customers' IT-systems with clickworker. The API is based on the on the Representational State Transfer (REST) standard. This means that all interaction with our platform is enabled by the usage of the well known http(s)-protocol.

Together with the API call reference, this document demonstrates how to use the API calls. It will also state limitations and important information about the API in the FAQ section at the end of the document.

The topics covered are:

- High-level description of the API integration steps
- Step-by-step guide of a project setup
- How-to for specific usage scenarios
- FAQ section

2 High-level description of the API integration steps

To benefit from clickworker's crowdsourcing services, customers need valid login credentials for the platform. With these credentials, they can upload their tasks through the API within an appropriate template, which includes several parameters like language, text length and a briefing for the clickworkers. Immediately our crowd starts working on these tasks. Finally, the customer must fetch the results through the API.

Two *clickworker* environments host installations of the API:

- Production: This is the production environment. It uses the latest released version of the API, works with production data, and fully interacts with external systems like email servers.

  The current production environment is available at:
  https://api.clickworker.com/api/marketplace/v2/

- Sandbox: This is the testing environment. It uses the latest released version of the API, works with dedicated test data, and has limited interaction with external systems like email servers. Customers can use this environment to test their client implementations. No clickworker crowd is available on Sandbox.

  The current sandbox environment is available at:
  https://sandbox.clickworker.com/api/marketplace/v2/

This high-level process is described more detailed in the following sections of this chapter.

1. Getting login credentials (username and password) for testing environment ('sandbox')
2. Testing access to the Sandbox environment
3. Choosing the appropriate product
4. Setting up the task-templates
5. Submitting your tasks.
6. Working on your tasks as a clickworker (sandbox only).
7. Reviewing / appriving or rejecting results
8. Fetching results data

2.1 Credentials and access via API

The clickworker Marketplace API uses HTTP Authentication ("Basic Auth") for access control.

For the testing environment ('sandbox') credentials must to be requested (helpdesk@clickworker.com) from our Solutions department.

The credentials used for API authentication on our production environment ('production') are the username and password. In order to request API access on production, please log in with your customer account and go to "Profile" / "API access".

To ensure data security, the API requires the use of Secure Socket Layers (SSL).

As the API is REST-based, a proper REST-client is needed to communicate with our API. For a first try-out a Mozilla Firefox Add-on is available[1]. For production usage we highly recommended to contact us

[1] https://addons.mozilla.org/de/firefox/addon/restclient/?src=search

(helpdesk@clickworker.com), as we also offer a client-side integration service. Alternatively you may use other command line tools such as “curl”, which is available on a wide range of operating systems.

With the given credentials you can get customer specific account information through the API.

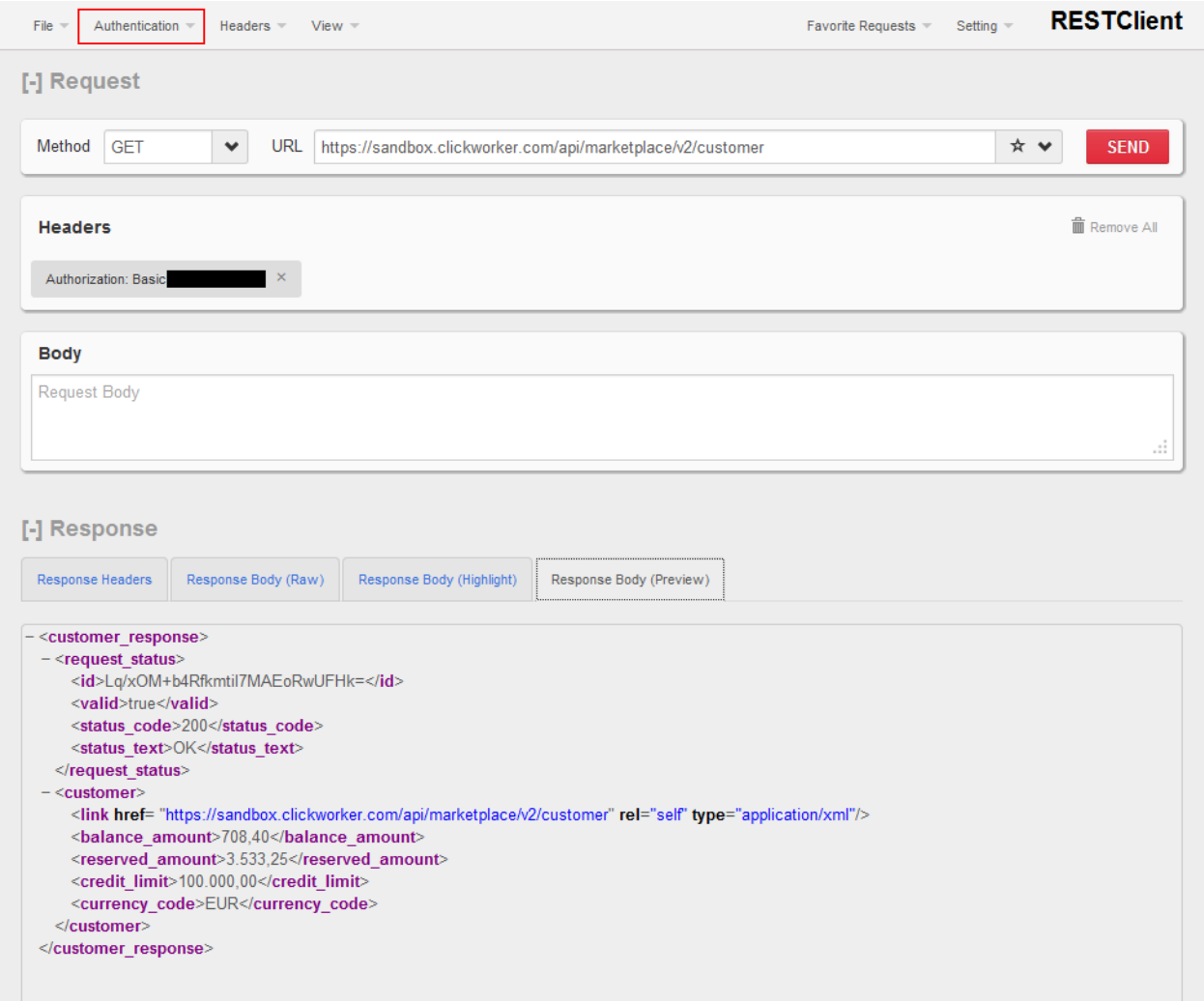

Figure 1: Firefox REST-Client

As shown in Figure 1 the URL for our testing environment is: https://sandbox.clickworker.com/api/marketplace/v2/customer.

Make sure that you provided your credentials to the REST-client (click on ‘Authentication’ – see red rectangle). Also add a http-Header as “Content-Type: application/xml” (click on ‘Headers’->’Custom Header’).

If everything is setup correctly, the response will be a XML-document under the tab ‘Response Body (Preview)’ (see Figure 1).

With this response you have verified that you have a working access to our API. In the following the necessary steps for a project setup are described.

2.2 Choose a product

The first step in the project creation is to choose which product to use for creating your project. Each product has unique quality mechanisms, pricings and options to configure the service. Out of the box, two services are offered. Our Solutions department may have configured a specific product to meet your client conditions and needs.

- **TextCreate**

  "Your content authored competently and economically. Our clickworkers create exclusive content and articles for you. From a single text to a large-scale order, quickly and at affordable rates."

  This product offers texts in **different languages** and in **different lengths**.

  The different languages offered are: German [de], Swedish [sv], Dutch [nl], Italian [it], Polish [pl], Turkish [tr], Spanish [es], Hungarian [hu], Czech [cs], Portuguese [pt], Romanian [ro], French [fr], Russian [ru], English [en], Danish [da], Finnish [fi], Norwegian [no].

  Additional to the language, customers can choose between different text lengths. These are (numbers mean words – [from#to]): 10#55, 50#110, 101#155, 151#205, 201#255, 251#305, 301#355, 351#405, 401#455, 451#505, 501#555, 551#605, 601#710, 701#810, 801#910, 901#1010 and 1001#1210.

  As a final decision customers need to choose if a second *clickworker* should proof-read the text to achieve an outstanding quality.

  Each customer project needs a proper task description for text creation, so our clickworkers know what kind of text they are supposed to write. We strongly advise also to deliver an example text, so the created texts meet the customer's expectations even more accurate.

- **TextCreateWithKeywords**

  "Content is King. Search engine optimized text. Our authors create and correct content that makes your site more attractive to search engines."

  The offered text lengths differ slightly from TextCreate and are (numbers mean words – [from#to]): 10#55, 50#110, **100#160**, 151#205, 201#255, 251#305, 301#355, 351#405, 401#455, 451#505, 501#555, 551#605, 601#710, 701#810, 801#910, 901#1010, 1001#1210 and **1450#1500**.

  For this product customers have to add at least one keyword to each text creation task, which will be incorporated into the texts by our clickworkers. This way we provide a highly-scaling SEO-Text creation option. All other aspects are equal to our 'TextCreate' product (see above).

2.3 Define your document templates (Task-Template)

Once you have chosen a product, you need to define your document structure by creating a new TaskTemplate. The TaskTemplate defines:

- The briefing to the clickworker for this type of task

Each customer needs to charge their account to create tasks through API. If you want to test our API on Sandbox, please contact us (helpdesk@clickworker.com). We will provide credits for your sandbox accounts.

- Input fields provided to the clickworker to fulfil the task (e.g. topic, keywords)
- Result fields filled out by clickworker when fulfilling the task (e.g title field, abstract, long text)

Defining a new TaskTemplate is explained in chapter “3.1 Define configuration”. New TaskTemplates are defined by posting the XML definition.

2.4 Creating tasks

Once you have chosen a product and configured your templates, you can post new tasks for your project, and resolution by our clickworkers. Together with the task description (briefing) this input data defines the task to be delivered by the clickworkers.

Submitted tasks will be distributed to the crowd and clickworkers will author the article according to the briefing. Depending on the quality assurance options configured, results will be reviewed by the crowd or our lecturer service and then sent to you for final approval. This step is called the ‘buyoff’.

2.5 Buyoff

Completing your submitted tasks will take some time. At any time you may have multiple tasks submitted and ongoing. The order of completion will be different than the order of submitting tasks. There are two options to get an overview of your active and completed tasks:

- **Polling / Pull approach**: As the customer, you may check regularly if your tasks have reached the state ‘Feedback’

- **Push notification / Notification approach:** When submitting a new task, you may add a Notification URL which will be called when the task has reached that state ‘Feedback’.

Push notifications are highly recommended as it is superior in terms of performance and load on your client system.

The buyoff carried out by the customer itself will be done via the “buyoff Job”: Each task is partitioned into different jobs, representing different steps in the overall workflow. E.g. if the task has been configured to include quality assurance there are at least 3 jobs involved to solve the task. The first job is to create the content. The second job is to review the content. And the third job is to execute the buyoff. So on the testing environment (‘sandbox’) the customer **needs to compete two jobs through the clickworker workplace** before the task reaches the buyoff state. This last job includes the results of the task and requests for approval.

In order to buy off or reject the result, you need to post an XML-document (http-POST) carrying the approval information from the customer’s API client to the API. This approval information ensures, that clickworkers will receive an appropriate rating and therefore is cruicial for sustaining good quality in your projects. More details on this will be given in section 3.3.2. Once approved, your may fetch the task results via the API.

2.6 Get task results

During task creation (see 2.4) the customer will have received a unique identifier for each task which was uploaded. With this given identifier you may check the results by calling this reference URL (http-GET). On the Sandbox-Environment this URL would be:

https://sandbox.clickworker.com/api/marketplace/v2/customer/tasks/<identifier>

Results are delivered as a XML payload within the response body as an answer to the http-request. The structure in detail can vary and depends on the structure the customer has configured within the step of the initial project creation (see 2.3).

2.7 Managing Clickworker Teams

Teams allow to group clickworkers and include or exclude them from your projects.

Teams are typically used in the following scenarios:

- Excluding clickworkers that have already participated in a previous survey.
- Only addressing specific clickworkers with a survey for follow up surveys
- Excluding authors from your text project (blacklisting)
- Assigning text projects to a list of known authors that have worked for you previously

2.7.1 Creating a new team

To create a new team you want to send POST request to /api/marketplace/v2/customer/teams, with specified XML body

```
<team>
  <id>[TEAM ID]</id>
  <name>[TEAM NAME]</name>
  <note>[OPTIONAL NOTE]</note>
  <hide>false</hide>
</team>
```

- <id> Team ID identifier (in)
- <name> Team name
- <note> Any note (optional).
- <hide> Identifies that team is hidden (optional boolean attribute).

2.7.2 Deleting a team

To permanently delete a team you want to send DELETE request to /api/marketplace/v2/customer/teams/<id> where <id> is an integer number – the Id of a team.

2.7.3 Modifying a team

To hide a team from marketplace drop down lists, use a PUT request to /api/marketplace/v2/customer/teams/<id> and set attribute hide into true:

```
<team>
  <hide>true</hide>
</team>
```

You may update the following attributes of a team:

- <name> The team name
- note Description of the team
  hide show/hide team from lists

2.7.4 Listing teams of a customer

To retrieve all customer's teams you want to send GET request to /api/marketplace/v2/customer/teams. Response will look like:

```
<teams_response>
  <request_status>
    <id>wybSKEcLKGBNbd+LKX7zXIsvVJc=</id>
    <valid>true</valid>
    <status_code>200</status_code>
    <status_text>OK</status_text>
    <total_count>5</total_count>
    <page_num>1</page_num>
    <page_size>50</page_size>
  </request_status>
  <teams>
    <id>12242</id>
    <name>Content Managers</name>
  </teams>
  <teams>
    <id>232304</id>
    <name>Interpreters</name
  </teams>
</teams_response>
```

2.7.5 Adding clickworkers to a team

To add clickworkers to the team you want to send PUT request to /api/marketplace/v2/customer/teams/<team_id>, with specified XML body

```
<team>
  <add_user_ids>[USER_ID]</add_user_ids>
  <add_user_ids>[USER_ID]</add_user_ids>
  <add_user_ids>[USER_ID]</add_user_ids>
</team>
```

<add_user_ids> is a list (Array) of clickworkers users Ids that should be added to the team, e.g. [1,2,3,4]

2.7.6 Removing clickworkers from a team

To remove clickworkers from the team send a PUT request to /api/marketplace/v2/customer/teams/<team_id> with the following XML body:

```
<team>
  <remove_user_ids>[USER_ID]</remove_user_ids>
  <remove_user_ids>[USER_ID]</remove_user_ids>
  <remove_user_ids>[USER_ID]</remove_user_ids>
</team>
```

<remove_user_ids> is a list (Array) of clickworkers users Ids that should be removed from the team, e.g. [1,2,3,4]

2.8 Summary

The previous chapters have explained the typical usage scenario for the API. The next chapter explains the process in more detail.

3 Project setup - Step-by-step

As mentioned in the previous section we offer two different products through the API. The initial configuration of the product according to the customer's needs is called the 'Project setup'.

Herewith the customer specifies the following aspects of his project:

- which Product is ordered (TextCreate, TextCreateWithKeywords including client specific products)
    - language of the text
    - length of the text
    - quality assurance on/off
- input data
- output data

3.1 Define TaskTemplate

To define the TaskTemplate, including your order specific parameters such as the briefing, target language, etc., the customer needs to create a specific structured XML-document for the API. This XML-document is called the 'Task-template'.

3.1.1 Configuration of standard attributes

The following XML document shows the standard attributes for the text products. Standard attributes are common to all TaskTemplates, regardless of the product chosen.

```
<task_template>
  <code>[unique_template_id_1234]</code>
  <name>[TEMPLATE NAME]</name>
  <titles>
  <de>[GERMAN TITLE]</de>
  <en>[ENGLISH TITLE]</en>
  </titles>
  <descriptions>
  <de>[GERMAN DESCRIPTION]</de>
  <en>[ENGLISH DESCRIPTION]</en>
  </descriptions>
</task_template>
```

Note: Values to be be filled in are marked in brackets placeholders "[]".

These standard product parameters have the following meaning:

- <code>: a unique identifier for the customer (string).
- <name>: a unique Task-template name for the customer (string)
- <titles>: The title in German (<de>) and English (<en>), will be shown to the clickworkers. It should be short, but descriptive.

- <descriptions>: The working instruction of the requested texts. H*ere the cust*omer needs to state as clear as possible what kind of text he expects to be written by the clickworkers.

The <descriptions> tag above is **essential** for the quality of the created texts. As it is shown to the clickworkers as their working instruction it should be easy to understand and formatted well. We strongly advise to use **encoded** html-tags to structure the description in a meaningful manner. For this online tools like http://www.opinionatedgeek.com/DotNet/Tools/HTMLEncode/encode.aspx can be used.

An encoded description looks like this:

```
&lt;h1&gt;I am an encoded html-working instruction&lt;/h1&gt;
&lt;h2&gt;I am an encoded html-working instruction&lt;/h2&gt;
&lt;h3&gt;I am an encoded html-working instruction&lt;/h3&gt;
&lt;ul&gt;
  &lt;li&gt;Point 1&lt;/li&gt;
  &lt;li&gt;Point 2&lt;/li&gt;
  &lt;li&gt;Point 3&lt;/li&gt;
&lt;/ul&gt;
```

which is encoded from the following original html text:

```
<h1>I am an encoded html-working instruction</h1>
<h2>I am an encoded html-working instruction</h2>
<h3>I am an encoded html-working instruction</h3>
<ul>
  <li>Point 1</li>
  <li>Point 2</li>
```

3.1.2 Product specific attributes

The child element <product> contains the product specific attributes.

```
<product>
  <link href="/api/marketplace/v2/products/TextCreate" rel="product"
type="application/xml"/>

  <attributes>
  <code>textcreate_language</code>
  <value>[WANTED LANGUAGE]</value>
  </attributes>

  <attributes>
  <code>textcreate_text_length</code>
  <value>[WANTED TEXT LENGTH]</value>
  </attributes>

  <attributes>
  <code>textcreate_proof_read</code>
  <value>[textcreate_proof_yes | textcreate_proof_no]</value>
  </attributes>
```

```
  <attributes>
  <code>textcreate_sample</code>
  <value>[EXAMPLE TEXT]</value>
  </attributes>

  <attributes>
  <code>textcreate_description</code>
  <value>[DESCRIPTION]</value>
  </attributes>
</product>
```

Choose your product by replacing

<link href="[URL TO PRODUCT]" rel="product" type="application/xml" />

by

- /api/marketplace/v2/products/TextCreate
- /api/marketplace/v2/products/TextCreateWithKeywords
- or the URL for your customer specific product provided by Solutions

See 2.2 for the differences between these two products.

In the example above the attributes for the "TextCreate" product are shown. Their meaning are:

- textcreate_language (mandatory)
  Defines the language of the text to be authored. The languages supported and their needed language code are listed here: [languages]

- textcreate_text_length (mandatory)
  Defines the valid range of text length. All results fields are counted and summed up. The supported lengths are listed here: [text lengths]. The syntax must be: [lower limit]#[upper limit] - e.g. 10#55

- textcreate_proof_read (mandatory)
  Define whether quality assurance using review by second clickworker is requested. Valid options are: "textcreate_proof_yes" or "textcreate_proof_no".
  Hint: For testing purposes in sandbox, you may want to switch quality assurance off, so you can bypass a number of jobs.

- textcreate_sample (optional)
  Contains an example text for clickworkers. We strongly recommended to provide examples, as they will significantly improve the quality of authored content. clickworker will have a clearer understanding of the style and character of the requested text.

- textcreate_description (mandatory)
  Contains an internal description for the task. This content will not be shown to clickworkers.

An example of a well-configured Task-template is shown as follows:

```
<product>
<link href="/api/marketplace/v2/products/TextCreate" rel="product"
type="application/xml"/>

  <attributes>
  <code>textcreate_language</code>
  <value>de</value>
  </attributes>

  <attributes>
  <code>textcreate_text_length</code>
  <value>10#55</value>
  </attributes>

  <attributes>
  <code>textcreate_proof_read</code>
  <value>textcreate_proof_yes</value>
  </attributes>

  <attributes>
  <code>textcreate_sample</code>
  <value>Hier sollte Ihr Beispieltext stehen.</value>
  </attributes>

  <attributes>
  <code>textcreate_description</code>
  <value>Lorem ipsum dolor sit amet, consectetur, adipisci velit, … </value>
  </attributes>
</product>
```

3.1.3 Form

Finally, the <form> element, defines the document structure for the tasks, based on input and output elements. It consists of several <elements> elements. Each of this element is either a read-only input or a writable output of each task.

3.1.3.1 Input

If a clickworker, for example, is asked to write a text about a hotel, and the information source is the URL of the hotel's website, this URL must be given to the clickworker. Exactly for this kind of input data for each task a read-only element is defined. These configurations are used to render the input values as HTML. In the hotel-URL example it could be defined as follows:

```
<elements>
  <is_output>false</is_output>
  <titles>
  <de> Hotel-URL</de>
  <en>Hotel URL</en>
  </titles>
```

```
  <is_mandatory>true</is_mandatory>
  <type>url</type>
  <item_code>link_to_hotel</item_code>
  <sort_order>10</sort_order>
</elements>
```

The children are:

- <is_output>
  The value ‘false’ in the above example defines this element as an input element for the task.

- <titles>
  This element provides the title of the element in the different languages: de (german) and en (english). The value here is shown to the *clickworker* in front of the corresponding input data (it is the label of the element).

- <is_mandatory>
  If “true”, defines that the clickworker must provide a value for this field.

- <type>
  In the above-described hotel URL example the type is given as ‘url’. With this the input data will be rendered as a clickable html link. Depending on the semantics of the input data several types of elements are available. See 5 for details.

- <item_code>
  The item code needs to be a unique identifier in the context of the Task-template. Herewith the association to the later uploaded input data is done.

- <sort_order>
  The final configuration done for the element is the sort order. This defines the position of the input element in the form. Over all elements defined within the Task-template the sort order determines the vertical order from top to bottom shown to the clickworkers.

Although the XML document is quite comprehensive, it is advised to check how the form looks like from a clickworker’s perspective. In order to do so, submit a task and login as a clickworker in the sandbox environment for preview.

3.1.3.2 Output

In addition to the input data, the desired output data structure carrying the results of the task must be defined. Similar to the input elements, output data structure is also defined by a set of <elements>. The main difference is that its child element <is_output> will be set to ‘true’.

Each output element can be configured more detailed, so the complete output can be structured to fulfil the customer’s requirements. The most frequently used detail configuration for an output element is the definition of the minimal and maximal length of a text field.

According to the hotel URL example, you may want the text the author to provide the result as a separate header field and a paragraph section, rather than one chunk of text. In this common scenario you would need to define two output elements.

The configuration for a heading and a paragraph is:

```
<elements>
  <is_output>true</is_output>
  <titles>
  <de>Überschrift</de>
  <en>Heading</en>
  </titles>
  <is_mandatory>true</is_mandatory>
  <type>text_field</type>
  <item_code>title_top</item_code>
  <sort_order>20</sort_order>
  <options>
  <code>min_length</code>
  <value>5</value>
  </options>
  <options>
  <code>max_length</code>
  <value>15</value>
  </options>
</elements>

<elements>
  <is_output>true</is_output>
  <titles>
  <de>Textblock</de>
  <en>Paragraph</en>
  </titles>
  <is_mandatory>true</is_mandatory>
  <type>text_area</type>
  <item_code>paragraph_bottom</item_code>
  <sort_order>30</sort_order>
  <options>
  <code>min_length</code>
  <value>5</value>
  </options>
  <options>
  <code>max_length</code>
  <value>40</value>
  </options>
</elements>
```

As shown the accumulated min_length of both elements is 10. The accumulated max_length of both elements is 55. This fits into the configured text length for the overall text length from 10 to 55.

Important Note: Make sure that the min/max lengths for the output fields add up to the valid min/max length of the total text. Otherwise this may result in situations where clickworkers cannot save their work due to conflicting contraints!

For a complete list of additional configuration options of output elements see chapter 5.

3.1.4 Task-template - Putting it all together

Putting all parts of the Task-template together the complete ordering/configuration of the project looks like this:

```
<task_template>
```

```
<code>unqiue_template_id_1234</code>
<name>Hotel-Beschreibungen</name>
<titles>
<de>Informative Beschreibungen von Hotels</de>
<en>Informative descriptions of hotels</en>
</titles>
<descriptions>
<de>Hier sollte Ihr Briefing für die clickworker stehen.</de>
<en>Here you should describe the instructions for the clickworkers.</en>
</descriptions>
<product>
<link href="/api/marketplace/v2/products/TextCreate" rel="product"
   type="application/xml"/>

<attributes>
  <code>textcreate_language</code>
  <value>de</value>
</attributes>

<attributes>
  <code>textcreate_text_length</code>
  <value>10#55</value>
</attributes>

<attributes>
  <code>textcreate_proof_read</code>
  <value>textcreate_proof_yes</value>
</attributes>

<attributes>
  <code>textcreate_sample</code>
  <value>Hier sollte ein Beispieltext stehen.</value>
</attributes>

<attributes>
  <code>textcreate_description</code>
  <value>Lorem ipsum dolor sit amet, consectetur, adipisci velit, … </value>
</attributes>
</product>

<form>
<elements>
  <is_output>false</is_output>
  <titles>
  <de>Hotel-URL</de>
  <en>Hotel URL</en>
  </titles>
  <is_mandatory>true</is_mandatory>
  <type>url</type>
  <item_code>link_to_hotel</item_code>
  <sort_order>10</sort_order>
</elements>

<elements>
  <is_output>true</is_output>
  <titles>
  <de>Überschrift</de>
```

```
    <en>Heading</en>
    </titles>
    <is_mandatory>true</is_mandatory>
    <type>text_field</type>
    <item_code>title_top</item_code>
    <sort_order>20</sort_order>
    <options>
    <code>min_length</code>
    <value>10</value>
    </options>
    <options>
    <code>max_length</code>
    <value>15</value>
    </options>
  </elements>

  <elements>
    <is_output>true</is_output>
    <titles>
    <de>Textblock</de>
    <en>Paragraph</en>
    </titles>
    <is_mandatory>true</is_mandatory>
    <type>text_area</type>
    <item_code>paragraph_bottom</item_code>
    <sort_order>30</sort_order>
  </elements>
  </form>
</task_template>
```

To finish the initial project configuration, send a http-POST request to https://<environment>.clickworker.com/api/marketplace/v2/customer/task_templates.. As we strongly recommend to test the setup first on the sandbox environment the <environment> placeholder should be ‘sandbox’ (for production this would be ‘api’).

The Task-template itself would be saved as a XML-document and forming the request-body.

3.2 Submitting a Task

Once the TaskTemplate has been defined, you are ready to submit new tasks. The structure of the XML document carrying the task data must fit to the corresponding Task-template. However there are some elements that need to be part of every task-creation process.

3.2.1 Task - common elements

```
<task>
  <customer_ref>[CUSTOMER ID]</customer_ref>
  <template>
  <link href="/api/marketplace/v2/customer/task_templates/[TEMPLATE CODE]"
     rel="task_template"                                    type="application/xml" />
  </template>
  <notifications>
  <event>CUSTOMER_INPUT_REQUIRED</event>
  <callback_url>http://notification.example.com</callback_url>
  <callback_method>POST</callback_method>
```

```
  <payload_format>XML</payload_format>
  </notifications>
  <notifications>
  <event>TASK_COMPLETED</event>
  <callback_url>http://notification.example.com</callback_url>
  <callback_method>POST</callback_method>
  <payload_format>XML</payload_format>
  </notifications>
</task>
```

- <customer_ref>
  This is an element to carry an identifier, which the customer may need for his own reference. This identifier will be included in each task output as it is.

- <template>
  As described in 3.1.3 the Task-template defines the structure of the input and the output data. Therefore the task itself must reference the Task-template it is designed for. As shown in 3.1.4 during the project setup process a Task-template is uploaded via API. Within this the customer has filled the <code> element with an unique identifier of the template. This identifier needs to be added to the common path:

```
<template>
  <link
href="/api/marketplace/v2/customer/task_templates/unqiue_template_id_1234"
rel="task_template" type="application/xml" />
</template>
```

- <notifications>
  The final task configuration value offers the customer the option to be notified via a http POST call about the state of the task adaption. During the editing process of the task on the *clickworker* platform the task passes several states. These state changes are indicated by two different events. The customer can define for each of these events an URL which will be automatically called when this event occurs:

  1. TASK_COMPLETED
     This event is triggered when all work related to the task, including the customer's review, has been completed.
  2. CUSTOMER_INPUT_REQUIRED
     This event is triggered when the customer needs to approve the task results.

  Part of this notification will be a payload carrying the results of the task. See 3.2.3 for details.

3.2.2 Task - Task-template specific elements

Now the task specific input data for an individual task needs to be added. As shown in the Task-template in 3.1.4 the <form> element defines an input item with the <item_code> given as 'link_to_hotel':

```
<elements>
  <is_output>false</is_output>
  <titles>
  <de>Hotel-URL</de>
  <en>Hotel URL</en>
```

```
  </titles>
  <is_mandatory>true</is_mandatory>
  <type>url</type>
  <item_code>link_to_hotel</item_code>
  <sort_order>10</sort_order>
</elements>
```

In this example this input data is needed by the *clickworker* to fulfil the task. This input element needs to be referenced by its item code within the XML document carrying the task data:

```
<input>
  <items>
  <code>link_to_hotel</code>
  <content>http://www.myhotel.com</content>
  </items>
</input>
```

3.2.3 Task - Putting it all together

The structure of the XML document carrying the task data the customer needs to upload would look like this:

```
<task>
  <customer_ref>[CUSTOMER ID]</customer_ref>
  <template>
  <link href="/api/marketplace/v2/customer/task_templates/[TEMPLATE CODE]"
     rel="task_template"                                    type="application/xml" />
  </template>
  <input>
  <items>
    <code>link_to_hotel</code>
    <content>http://www.myhotel.com</content>
  </items>
  </input>
  <notifications>
  <event>CUSTOMER_INPUT_REQUIRED</event>
  <callback_url>http://notification.example.com</callback_url>
  <callback_method>POST</callback_method>
  <payload_format>XML</payload_format>
  </notifications>
  <notifications>
  <event>TASK_COMPLETED</event>
  <callback_url>http://notification.example.com</callback_url>
  <callback_method>POST</callback_method>
  <payload_format>XML</payload_format>
  </notifications>
</task>
```

Send this XML document as a HTTP-POST request to https://<environment>.clickworker.com/api/marketplace/v2/customer/tasks. As we strongly recommend to test the setup first on the sandbox environment the <environment> placeholder should be ‘sandbox’ (for production this would be ‘api’).

The response to this API-Request will be a http-status code:

- 200, if the task was successfully created
- 400, if the referenced Task-template does not exist

and a response body:

```
<?xml version="1.0"?>
<task_response>
  <request_status>
  <id>gl1ZwYiU/BXh1+UYxn/9yoiCSLE=</id>
  <valid>true</valid>
  <status_code>200</status_code>
  <status_text>OK</status_text>
  </request_status>
  <task>
  <link
     href="https://sandbox.clickworker.com/en/api/marketplace/v2/customer/tasks/240581
     0" rel="self" type="application/xml"/>
  <id>2405810</id>
  <customer_ref>[CUSTOMER ID]</customer_ref>
  <template>
    <link
      href="https://sandbox.clickworker.com/en/api/marketplace/v2/customer/task_templ
      ates/[TEMPLATE CODE]" rel="self" type="application/xml"/>
    <product>
    <link
       href="https://sandbox.clickworker.com/en/api/marketplace/v2/products/TextCreat
       e" rel="self" type="application/xml"/>
    </product>
  </template>
  <net_amount>[€]</net_amount>
  <tax_amount>[€]</tax_amount>
  <amount>[€]</amount>
  <currency>EUR</currency>
  <state>queued</state>
  <inputs>
    <code>link_to_hotel</code>
    <content>http://www.myhotel.com</content>
  </inputs>
  <notifications>
    <link href=
      "https://sandbox.clickworker.com/api/marketplace/v2/customer/tasks/2405811/noti
      fications/19265" rel="self" type="application/xml"/>
    <event>CUSTOMER_INPUT_REQUIRED</event>
    <callback_url>http://notification.example.com</callback_url>
    <callback_method>POST</callback_method>
    <payload_format>XML</payload_format>
  </notifications>
    <notifications>
    <link href=
      "https://sandbox.clickworker.com/api/marketplace/v2/customer/tasks/2405811/noti
      fications/19266" rel="self" type="application/xml"/>
    <event>TASK_COMPLETED</event>
    <callback_url>http://notification.example.com</callback_url>
    <callback_method>POST</callback_method>
    <payload_format>XML</payload_format>
  </notifications>
```

```
  </task>
</task_response>
```

As shown above, part of the response is a <id> element. **The content of this element will be the reference required for buyoff**. It is a unique identifier for this individual task.

3.3 Buyoff of results

As outlined before, upon completion of a task, the customer is typically notified about the task completion. As soon as the task is waiting for approval, the customer is notified by a call to the URL as defined by the <callback_url> element of the task-upload procedure (see 3.2.1). The body of this request will be a XML document carrying the necessary information for the customer to execute the buyoff.

3.3.1 BuyOff Notification

The following payload is sent by the notification to inform the customer that there is some action required:

```
<?xml version="1.0"?>
<notification>
  <link
   href="https://sandbox.clickworker.com/de/api/marketplace/v2/customer/tasks/2405829/
   notifications/19265" rel="self" type="application/xml"/>
  <event>CUSTOMER_INPUT_REQUIRED</event>
  <task>
  <link
     href="https://sandbox.clickworker.com/de/api/marketplace/v2/customer/tasks/240582
     9" rel="self" type="application/xml"/>
  </task>
  <job>
  <link
     href="https://sandbox.clickworker.com/de/api/marketplace/v2/customer/jobs/5968545
     " rel="self" type="application/xml"/>
  </job>
</notification>
```

As shown above the <job> element includes a <link> with an URL to the job which expects the buyoff. The customer now needs to judge the result quality.

With the given URL an http-get request will show the results for review:

```
<?xml version="1.0"?>
<job_response>
<request_status>
  <id>RgsvEX3l3o8dS8pAjkQ02eXb2Ew=</id>
  <valid>true</valid>
  <status_code>200</status_code>
  <status_text>OK</status_text>
</request_status>
<job>
  <link
   href="https://sandbox.clickworker.com/api/marketplace/v2/customer/jobs/5968547"
   rel="self" type="application/xml"/>
  <id>5968547</id>
  <task>
```

```
  <link
    href="https://sandbox.clickworker.com/api/marketplace/v2/customer/tasks/2405830"
    rel="task" type="application/xml"/>
  </task>
  <items>
  <code>link_to_hotel</code>
  <content>http://www.myhotel.com</content>
  </items>
  <items>
  <code>title_top</code>
  <content>Title text created by clickworker</content>
  </items>
  <items>
  <code>paragraph_bottom</code>
  <content>Paragraph text created by clickworker</content>
  </items>
</job>
</job_response>
```

The various <items> elements within the <job> element correlate to the definition of the Task-template (see 3.1.4). The given item with the <item_code> 'link_to_hotel' was an input element, telling the clickworker to write a text about the hotel found under that URL.

The items with <item_code> 'title_top' and 'paragraph_bottom' are the output elements created by the *clickworker*. These texts are the ones which need to be judged to approve or to reject these results.

3.3.2 Approval / Rejection

3.3.2.1 Approval

Assumed that the quality of the output elements meet the customer expectations, the job should be approved.

This job approval is executed by the posting of a XML document using the http-PUT method. The XML-document sent as payload of that http-PUT request is simple:

```
<job>
  <input>
  <items>
    <code>accepted</code>
    <content>1</content>
  </items>
  </input>
</job>
```

If you have indicated to get notified upon task completion, the notification will be sent to you as soon as you have approved the results (TASK_COMPLETED event). The payload of this notification is as follows. Therefore you do not need to pool for results.

```
<?xml version="1.0"?>
<notification>
  <link type="application/xml" rel="self" href="
   https://sandbox.clickworker.com/de/api/marketplace/v2/customer/tasks/2405829/notifi
   cations/19266"/>
```

```
  <event>TASK_COMPLETED</event>
  <task>
  <link type="application/xml" rel="self" href="
     https://sandbox.clickworker.com/de/api/marketplace/v2/customer/tasks/2405829"/>
  </task>
  <customer_output>
    <title_top>Title text created by clickworker</title_top>
    <paragraph_bottom>Paragraph text created by clickworker</paragraph_bottom>
  </customer_output>
</notification>
```

Within the <customer_output> element the output elements are embedded.

3.3.2.2 Rejection

Assumed that the result is not satisfying and does not comply with the briefing you may reject the job results. Rejecting a job is done by posting a XML document (http-PUT method). In comparison to the approval, the XML-Document needs some more elements to let the clickworker know why the job result has been rejected and allow for mitigation. Additionally as a requirement from the chosen product (within this chapter the Product “TextCreate” was chosen - see 3.1.2) individual aspects of the created text need to be judged:

```
<job>
  <input>
    <!-- Text is rejected: accepted=0 -->
    <items>
      <code>accepted</code>
      <content>0</content>
    </items>
    <!--Judge grammar: 4=Perfect, 3=Very good, 2 = Ok, 0 = poor -->
    <items>
      <code>grade_gr</code>
      <content>2</content>
    </items>
    <!--Comment your judge on grammar. This is obliged if you judged it with a value <
3 -->
    <items>
      <code>grade_gr_comment</code>
      <content>Here is my grade_gr comment</content>
    </items>
    <!--Judge spelling and punctuation: 4=Perfect, 3=Very good, 2 = Ok, 0 = poor -->
    <items>
      <code>grade_sppu</code>
      <content>2</content>
    </items>
    <!--Comment your judge on spelling and punctuation. This is obliged if you judged
     it with a value < 3 -->
    <items>
      <code>grade_sppu_comment</code>
      <content>Here is my grade_sppu comment</content>
    </items>
    <!--Judge sentence structure: 4=Perfect, 3=Very good, 2 = Ok, 0 = poor -->
    <items>
      <code>grade_sest</code>
      <content>2</content>
    </items>
```

```
    <!--Comment your judge on sentence structure. This is obliged if you judged it
     with a value < 3 -->
    <items>
      <code>grade_sest_comment</code>
      <content>Here is my grade_sest comment</content>
    </items>
    <!--Judge the style: 4=Perfect, 3=Very good, 2 = Ok, 0 = poor -->
    <items>
      <code>grade_st</code>
      <content>2</content>
    </items>
    <!--Comment your judge on style. This is obliged if you judged it with a value < 3
-->
    <items>
      <code>grade_st_comment</code>
      <content>Here is my grade_st comment</content>
    </items>
    <!--Judge comprehension and clarity: 4=Perfect, 3=Very good, 2 = Ok, 0 = poor -->
    <items>
      <code>grade_cocl</code>
      <content>2</content>
    </items>
    <!--Comment your judge on comprehension and clarity. This is obliged if you judged
     it with a value < 3 -->
    <items>
      <code>grade_cocl_comment</code>
      <content>Here is my grade_cocl comment</content>
    </items>
    <!--Judge the adherence to the working instructions: 4=Perfect, 3=Very good, 2 =
     Ok, 0 = poor -->
    <items>
      <code>grade_foin</code>
      <content>2</content>
    </items>
    <!--Comment your judge on adherence to the working instructions. This is obliged
     if you judged it with a value < 3 -->
    <items>
      <code>grade_foin_comment</code>
      <content>Here is my grade_foin comment</content>
    </items>
    <!--Judge selection of information: 4=Perfect, 3=Very good, 2 = Ok, 0 = poor -->
    <items>
      <code>grade_sein</code>
      <content>2</content>
    </items>
    <!--Comment your judge on the selection of information. This is obliged if you
     judged it with a value < 3 -->
    <items>
      <code>grade_sein_comment</code>
      <content>Here is my grade_sein comment</content>
    </items>
    <!--Comment for the author. This is obliged.-->
    <items>
      <code>comment</code>
      <content>Text ist in Ordnung.</content>
    </items>
  </input>
```

```
</job>
```

The following table lists the aspects for the judgement of the ‘TextCreate’ and ‘TextCreateWithKeywords’ Product. The column ‘item’ lists the <item_code> of the aspect. Column ‘meaning’ explains what the aspect means. Finally the third column ‘condition’ lists the circumstances under which the corresponding aspect must be included in the XML-Document executing the buyoff:

item	meaning	condition
accepted	Is the result accepted?	Always. Either 0 (rejected) or 1 (accepted)
grade_gr	Judge the grammar.	If accepted=0. 4=Perfect, 3=Very good, 2 = Ok, 0 = poor
grade_gr_comment	Comment your Judge on grammar.	If grade_gr < 3.
grade_sppu	Judge spelling and punctuation.	If accepted=0. 4=Perfect, 3=Very good, 2 = Ok, 0 = poor
grade_sppu_comment	Comment your judge on spelling and punctuation.	If grade_sppu < 3.
grade_sest	Judge sentence structure.	If accepted=0. 4=Perfect, 3=Very good, 2 = Ok, 0 = poor
grade_sest_comment	Comment your judge on sentence structure.	If grade_sest < 3.
grade_st	Judge the style.	If accepted=0. 4=Perfect, 3=Very good, 2 = Ok, 0 = poor
grade_st_comment	Comment your judge on style.	If grade_st < 3.
grade_cocl	Judge comprehension and clarity.	If accepted=0. 4=Perfect, 3=Very good, 2 = Ok, 0 = poor
grade_cocl_comment	Comment your judge on comprehension and clarity.	If grade_cocl < 3.
grade_foin	Judge the adherence to the working instructions.	If accepted=0. 4=Perfect, 3=Very good, 2 = Ok, 0 = poor
grade_foin_comment	Comment your judge on the adherence to the working instructions.	If grade_foin < 3.
grade_sein	Judge selection of information.	If accepted=0. 4=Perfect, 3=Very good, 2 = Ok, 0 = poor
grade_sein_comment	Comment your judge on selection of information.	If grade_sein < 3.
comment	Comment your reason to reject the results.	If accepted=0.

Table 1 Buyoff items

After the customer has executed a rejection, the clickworker will once get the chance to correct the results according to the customer’s comments. After the rework the customer is getting informed as before (see 3.3.1). The whole buyoff process restarts until the customer is doing the approval (see 3.3.2.1).

If a text is declined twice, the task will be canceled and 'cloned'; the task will be handled by another clickworker. In this scenario the results will be created for another task-id. As the customer only knows the 'original' task id (see 3.2.3), the parameter '**show_cloned_task=true**' should always be part of the above-mentioned request URL. With this parameter the customer is always able to get the results referencing the 'original' id.

3.4 Getting the results

As shown in 3.3.2.1, once the approval has been done, the results will be delivered as payload of the TASK_COMPLETED notification. If no callback URL was defined [see 3.2.1], you may get a detailed view of every task he ever created. During the task creation process you will receive the **id** as a unique reference to this task. Using this id allows to check all task related aspects.
In order to get the results, use the following http-GET request:
https://[environment].clickworker.com/api/marketplace/v2/customer/tasks/<identifier>?show_cloned_task=true

For a task id given as 2405829 on the sandbox environment this would be:

https://sandbox.clickworker.com/api/marketplace/v2/customer/tasks/2405829?show_cloned_task=true

If the task has reached the state 'finished' there will be a <results> item within the XML-response for each output element as defined within the Task-template (see 3.1.3.2). The results part of the 'task-details' request would look like this:

```
<task_response>
  .
  .
  .
<results>
  <code>title_top</code>
  <content>Title text created by clickworker</content>
</results>
<results>
  <code>paragraph_bottom</code>
  <content>Paragraph text created by clickworker</content>
</results>
  .
  .
  .
</task_response>
```

3.5 TextCreateWithKeywords

As shown in 3.1.2 one part of the Task-template is the product configuration. With 'TextCreateWithKeyword' we would choose a different product and the product configuration would need to be changed as follows:

```
<product>
  <link href="/api/marketplace/v2/products/TextCreateWithKeywords" rel="product"
type="application/xml"/>
```

```
  <attributes>
  <code>textcreatewithkeywords_language</code>
  <value>[WANTED LANGUAGE]</value>
  </attributes>

  <attributes>
  <code>textcreatewithkeywords_text_length</code>
  <value>[WANTED TEXT LENGTH]</value>
  </attributes>

  <attributes>
  <code>textcreatewithkeywords_proof_read</code>
  <value>[textcreate_proof_yes | textcreate_proof_no]</value>
  </attributes>

  <attributes>
  <code>textcreatewithkeywords_sample</code>
  <value>[EXAMPLE TEXT]</value>
  </attributes>

  <attributes>
  <code>textcreatewithkeywords_description</code>
  <value>[DESCRIPTION]</value>
  </attributes>

  <attributes>
    <code> textcreatewithkeywords_team_whitelist</code>
    <value>[Team ID]</value>
  </attributes>

  <attributes>
    <code> textcreatewithkeywords_team_blacklist</code>
    <value>[Team ID]</value>
  </attributes>
</product>
```

Mainly the <code> elements differ from the 'TextCreate' product according to the product's name. Also a slightly difference within the offered text lengths must be considered (see 2.2). Last but not least the <link> element must point to the product.

The main difference is viewable within the form definition of the Task-template. There must be a definition of the keywords the *clickworker* is supposed to include in the text he they creates. So additionally to the 'normal' input elements (3.1.3.1) the keyword elements must be defined as follows:

```
<elements>
  <is_output>false</is_output>
  <titles>
    <de>Keyword A</de>
    <en>Keyword A</en>
  </titles>
  <is_mandatory>true</is_mandatory>
  <type>keyword</type>
  <item_code>keyword_a</item_code>
  <sort_order>50</sort_order>
  <options>
    <code>min_occurrence_ref</code>
```

```
    <value>keyword_a_min</value>
  </options>
  <options>
    <code>max_occurrence_ref</code>
    <value>keyword_a_max</value>
  </options>
  <options>
    <code>reference_code</code>
    <value>all</value>
  </options>
</elements>
```

This element has the value ‘keyword’ for the <type> element configured.

The type ‘keyword’ has the following options to be defined:

code	meaning
min_occurrence_ref	A reference to the <code> element of the task creation process where the minimum keyword occurrence is defined.
max_occurrence_ref	A reference to the <code> element of the task creation process where the maximum keyword occurrence is defined.
reference_code	Defines the Task’s data item where keywords are expected and will be counted. Enter the code of a data item here to check for keywords only in thiss item. Otherwise, leave blank or use the special value “all” to scan for keywords in all suitable fields.

These options and their relevance become more apparent if we look to the corresponding Task-upload.

3.5.1 Task submission using Product TextCreateWithKeywords

For the above described definition of a keyword element the related Task-upload (see also 3.2.2) needs to be extended by an additional input element:

```
<items>
  <code>keyword_a</code>
  <content>Bottrop</content>
</items>
<items>
  <code>keyword_a_min</code>
  <content>1</content>
</items>
<items>
  <code>keyword_a_max</code>
  <content>2</content>
</items>
```

The items with the defined <code> ‘keyword_a_min’ and ‘keyword_a_max’ define the minimal and maximal occurence count of the keyword ‘Bottrop’. These item codes are defined within the Task-template. Together with the defined value ‘all’ for the keyword option ‘reference_code’ this means that the term ‘Bottrop’ must be included 1 or 2 times within the result fields ‘title_top’ or ‘paragraph_bottom’ (as per our example in 3.1.3.2).

With this definition the clickworker has to submit the results including 1 or 2 times the term ‘Bottrop’.

If the keyword should only be included in a particular output field this would need to be defined within the Task-template. E.G. a definition of:

```
<options>
    <code>reference_code</code>
    <value>title_top</value>
</options>
```

would ensure that the keyword is only included to the output field ‘title_top’.

3.5.2 Summary

As mentioned before there are two main aspects which differ between the ‘TextCreate’ and the ‘TextCreateWithKeywords’ Product:

- The Task-template defines an input element of type ‘keyword’
- The Task-upload must include the keyword and defines it’s occurrence count

3.6 Survey

Additional to the both above explained text-based products we offer a Product “Survey”. This product offers the opportunity to survey our Clickworkers to any scientific or market oriented topic. The main idea here is that a customer prepares a survey on any external website reachable via a Url. The Clickworkers will click on the given link and participate in the survey. Once the survey is finished a code is shown to the participants. This code is pre-configured on the clickworker.com platform and needs to be entered into a form to proof that he finished the survey. With this the Clickworker is getting paid.

The use of the Product survey starts again with the definition of the task-template.

3.6.1 Product specific attributes

As the standard attributes for the task-template are always the same (see 3.1.1) we start with the Product specific attributes:

```
<product>
  <link href="/api/marketplace/v2/products/Survey" rel="product"
type="application/xml"/>

  <attributes>
  <code>mp_df_survey_country</code>
  <value>[Countrycode]</value>
  </attributes>

  <attributes>
  <code>mp_df_survey_payment</code>
  <value>[CW payment in Cent]</value>
```

```
  </attributes>

  <attributes>
  <code>mp_df_survey_age_from</code>
  <value>[18-99]</value>
  </attributes>

  <attributes>
  <code>mp_df_survey_age_to</code>
  <value>[18-99]</value>
  </attributes>

  <attributes>
  <code>mp_df_survey_gender</code>
  <value>[f|m|a]</value>
  </attributes>

  <attributes>
  <code>mp_df_survey_expiration</code>
  <value>[expiration-date]</value>
  </attributes>

  <attributes>
  <code>mp_df_survey_code</code>
  <value>[confirmation-code]</value>
  </attributes>

  <attributes>
    <code>mp_df_survey_add_user_hash</code>
    <value>[yes|no]</value>
  </attributes>

  <attributes>
    <code>mp_df_same_url_for_all</code>
    <value>[yes|no]</value>
  </attributes>

  <attributes>
    <code>mp_df_individual_codes</code>
    <value>[yes|no]</value>
  </attributes>

  <attributes>
    <code>team_whitelist</code>
    <value>[Team ID]</value>
  </attributes>

  <attributes>
    <code>team_blacklist</code>
    <value>[Team ID]</value>
  </attributes>

</product>
```

- mp_df_survey_country (mandatory)
  With this you can choose the country where our Clickworkers need to be located to participate in your survey. Valid country-codes are: at (Austria), be (Belgium), gb (Great Britain), es (Spain), pt (Portugal), pl (Poland), nl (Netherlands), it (Italy), in (India), fr (France), de (Germany), ca (Canada), us (USA), ch (Suisse). Mandatory if no whitelist provided.

- mp_df_survey_payment (mandatory)
  Here you enter the Clickworker's fee to participate in your survey. We will add 40% to the amount for your invoice. Please offer a reasonable fee to make sure enough of our Clickworker's are motivated to participate in your survey. The number will be interpreted as **Cent** (1/100) from your configured customer currency.

- mp_df_survey_age_from (mandatory)
  Here you can enter the minimum age of the Clickworker's to participate (**18-99**). Mandatory if no whitelist provided.

- mp_df_survey_age_to (mandatory)
  Here you can enter the maximum age of the Clickworker's to participate (**18-99**). Mandatory if no whitelist provided.

- mp_df_survey_gender (optional)
  Here you can enter the wanted gender of the Clickworkers to participate. Valid values are **f** (female) and **m** (male) or **a** (all).

- mp_df_survey_expiration (optional)
  Here you enter until when your survey shall be online (if not yet finished). Enter in the format yyyy-mm-dd (e.G.: 2015-01-02)

- mp_df_survey_code (mandatory)
  To ensure that our Clickworkers complete the survey, you have to show a code at the end of the survey. Our Clickworkers will enter this code on our platform to proof that they participated in the survey. Here you have to enter exactly the code you will show at the end of the survey. This code is the same for all participants (the use of individual codes is also possible – see below 'mp_df_individual_codes').

- mp_df_survey_add_user_hash (optional)
  Here you configure if the we shall append an unique identifier of the Clickworker calling the URL of the survey. Allowed values are 'yes' or 'no' with the following result:
  - yes: http://www.mysurvey.com/ -> http://www.mysurvey.com/?user=af37289db32b2c779273a982828fe8e79a08b782
  - no: http://www.mysurvey.com/ -> http://www.mysurvey.com/

  Be aware that this identification is for statistical usage only.

- mp_df_same_url_for_all
  If the URL to the externally hosted Survey is the same for all participants, configure 'yes'. In this case you will be able to easily enter the needed number of participants and the URL in one task creation request only.

- mp_df_individual_codes
  If you are able to provide an individual URL for each Participant it would also be possible to store an individual confirmation code for each URL. In this case a unique confirmation code would be shown at the end of the survey, which the Clickworker would need to enter within our Platform to get paid for his participation. This makes only sense if 'mp_df_same_url_for_all' (see previous bullet point) has been configured as 'no'. If you choose 'yes' for the individual confirmation-codes an additional form-element named 'code' needs to be part of your task creation request. Within this element you enter the confirmation code for each URL separately. If a Clickworker picks up the Job with the particular URL our system expects the Clickworker to enter the code configured here within the same task creation request. Only if the Clickworker's Code and the Code from the task creation request are the same the Clickworker is getting paid.

- team_whitelist (optional)
  Users from whitelist (team) will be included to the survey. If no demography is provided, then only the users on the whitelist can participate.

- team_blacklist (optional)
  Users from blacklist (team) will be excluded from the survey.

3.6.2 The form

In general in case of external hosted surveys there are only one or two input field necessary to show the external Url (and maybe transfer the individual confirmation code or the number of requested participants) to our Clickworkers:

```
<form>
  <elements>
    <is_output>false</is_output>
    <titles>
      <de>URL</de>
      <en>URL</en>
    </titles>
    <is_mandatory>true</is_mandatory>
    <type>url</type>
    <item_code>link_to_survey</item_code>
    <sort_order>10</sort_order>
  </elements>

  <!-- next element only if 'mp_df_same_url_for_all' has been configured as 'yes' and
'mp_df_individual_codes' has been configured as 'no' in the product-configuration
above -->
  <elements>
    <is_output>false</is_output>
    <titles>
        <de>no_of_participants</de>
        <en>no_of_participants</en>
    </titles>
    <is_mandatory>true</is_mandatory>
    <type>hidden</type>
    <item_code>participants</item_code>
    <sort_order>20</sort_order>
  </elements>
```

```
<!-- next element only if 'mp_df_same_url_for_all' has been configured as 'no' and
'mp_df_individual_codes' has been configured as 'yes' in the product-configuration
above -->
  <elements>
    <is_output>false</is_output>
    <titles>
        <de>Code</de>
        <en>Code</en>
    </titles>
    <is_mandatory>false</is_mandatory>
    <type>hidden</type>
    <item_code>code</item_code>
    <sort_order>20</sort_order>
  </elements>

</form>
```

3.6.3 Putting it all together

To prepare a survey with female Cickworkers from Germany in the age from 18-60, paying 100 Eurocent each we would create a task-template like this:

```
<task_template>
  <code>unqiue_template_id_1234</code>
  <name>Survey3000</name>
  <titles>
  <de>Eine Umfrage über Handies</de>
  <en>A survey about mobiles</en>
  </titles>
  <descriptions>
  <de>Bitte klicken Sie auf die URL und nehmen Sie an der Umfrage teil.</de>
  <en>Please click on the link and participate in the survey.</en>
  </descriptions>
  <product>
  <link href="/api/marketplace/v2/products/Survey" rel="product"
     type="application/xml"/>

  <attributes>
    <code>mp_df_survey_country</code>
    <value>de</value>
  </attributes>

  <attributes>
    <code>mp_df_survey_payment</code>
    <value>100</value>
  </attributes>

  <attributes>
    <code>mp_df_survey_age_from</code>
    <value>18</value>
  </attributes>

  <attributes>
    <code>mp_df_survey_age_to</code>
    <value>60</value>
  </attributes>
```

```
<attributes>
  <code>mp_df_survey_gender</code>
  <value>f</value>
</attributes>

<attributes>
  <code>mp_df_survey_expiration</code>
  <value>2015-03-30</value>
</attributes>

<attributes>
  <code>mp_df_survey_code</code>
  <value>1234</value>
</attributes>

<attributes>
  <code>mp_df_survey_add_user_hash</code>
  <value>no</value>
</attributes>

<attributes>
  <code>mp_df_same_url_for_all</code>
  <value>yes</value>
</attributes>

<attributes>
  <code>mp_df_individual_codes</code>
  <value>no</value>
</attributes>

</product>

<form>
<elements>
  <is_output>false</is_output>
  <titles>
  <de>URL</de>
  <en>URL</en>
  </titles>
  <is_mandatory>true</is_mandatory>
  <type>url</type>
  <item_code>link_to_survey</item_code>
  <sort_order>10</sort_order>
</elements>

<elements>
  <is_output>false</is_output>
  <titles>
      <de>no_of_participants</de>
      <en>no_of_participants</en>
  </titles>
  <is_mandatory>true</is_mandatory>
  <type>hidden</type>
  <item_code>participants</item_code>
  <sort_order>20</sort_order>
</elements>
```

```
<!—Following element not needed as ‘mp_df_same_url_for_all’ has been configured as
‘yes’ and ‘mp_df_individual_codes’ has been configured as ‘no’ above.
  <elements>
    <is_output>false</is_output>
    <titles>
        <de>Code</de>
        <en>Code</en>
    </titles>
    <is_mandatory>false</is_mandatory>
    <type>hidden</type>
    <item_code>code</item_code>
    <sort_order>20</sort_order>
  </elements>
-->

  </form>
</task_template>
```

3.6.4 Upload a task for the survey

With the above constructed task-template we can immediately post a task with the following configuration:

```
<task>
    <customer_ref>Handy-Umfrage</customer_ref>
    <template>
        <link
href="/api/marketplace/v2/customer/task_templates/unqiue%255Ftemplate%255Fid%255F1234"
rel="task_template" type="application/xml" />
    </template>
    <input>
        <items>
            <code>link_to_survey</code>
            <content>http://www.mysurvey.de/survey3000</content>
        </items>
      <items>
        <code>participants</code>
        <content>1000</content>
      </items>
<!—Following element not needed as ‘mp_df_same_url_for_all’ has been configured as
‘yes’ and ‘mp_df_individual_codes’ has been configured as ‘no’ within the task-
template. But if it would be configured the other way around it would be needed and it
would replace the former item ‘participants’
      <items>
        <code>code</code>
        <content>6789</content>
      </items>
-->
    </input>
</task>
```

As a response the following XML document will be send by our API:

```
<task_response>
    <request_status>
        <id>7a++TncqQ/QUgUTGQXy6PiHu/Rs=</id>
        <valid>true</valid>
        <status_code>200</status_code>
        <status_text>OK</status_text>
    </request_status>
    <task>
        <link type="application/xml" href=
"http://localhost:3000/api/marketplace/v2/customer/tasks/174811" rel="self"/>
        <id>174811</id>
        <customer_ref>Handy-Umfrage</customer_ref>
        <template>
            <link type="application/xml" href=
"http://localhost:3000/api/marketplace/v2/customer/task_templates/unqiue_template_id_1
234" rel="self"/>
            <product>
                <link type="application/xml" href=
"http://localhost:3000/api/marketplace/v2/products/Survey" rel="self"/>
            </product>
        </template>
        <net_amount>1.4</net_amount>
        <tax_amount>0.27</tax_amount>
        <amount>1.67</amount>
        <currency>EUR</currency>
        <state>confirmed</state>
        <inputs>
            <code>link_to_survey</code>
            <content>http://www.mysurvey.de/survey3000</content>
        </inputs>
        <inputs>
            <code>participants</code>
            <content>1000</content>
        </inputs>
    </task>
</task_response>
```

As you can see we invoice 140% of the Clickworker's fee (1.4 * 100 Cent = 1.40€).

The above given task will be rendered on the clickworker platform as follows:

If you want 1000 Clickworkers to participate in the survey you may need to post 1000 tasks through our API for technical reasons. We make sure that only distinct Clickworkers will participate. If you have choosen ‘mp_df_same_url_for_all’ as ‘yes’ within your task-template you would be able to request 1000 participants with the post of one task only.

4 How-to for specific usage scenarios

Within this chapter the following scenarios will be described:

- How to get task-results if notifications cannot be used
- How to find tasks using the customer specific task reference

4.1 How to get task-results if notifications cannot be used

If you cannot use notification for processing the events CUSTOMER_INPUT_REQUIRED and/or the TASK_COMPLETED, you must take care tracking result sign off yourself. As explained in 3.4 any Task detail can be retrieved by an http-GET request to the corresponding URL (e.g.:

https://[environment].clickworker.com/api/marketplace/v2/customer/tasks/2405829?show_cloned_task=true).

For practical reasons this approach must be extended by a feature to get an overview of all tasks and their corresponding states. A http-GET request to the URL:

https://[environment].clickworker.com/api/marketplace/v2/customer/tasks would show all tasks from the customer as a list in the following format:

```
<tasks>
  <link href=
   "https://[environment].clickworker.com/api/marketplace/v2/customer/tasks/[task_id]"
   rel="task" type="application/xml"/>
  <customer_ref>[CUSTOMER_REF]</customer_ref>
  <template>
    <link href=
    "https://[environment].clickworker.com/api/marketplace/v2/customer/task_templates/
    [tt_code]" rel="task_template" type="application/xml"/>
    <product>
    <link href=
    "https://[environment].clickworker.com/api/marketplace/v2/products/TextCreate"
    rel="product" type="application/xml"/>
    </product>
  </template>
  <state>finished</state>
  <created_at>2013-10-17T09:22:16+02:00</created_at>
</tasks>
```

The request-URL can be configured by the use of request-Parameter:

parameter	meaning	values
state	Filters tasks by specified state.	Feedback [All Tasks which require a customer buyoff] Finished
from	Filters tasks by created_at date	dd.mm.yyyy
to		

To request a list of tasks that are waiting for the customer's approval, the URL would be:

https://[environment].clickworker.com/api/marketplace/v2/customer/tasks?state=feedback

To request a list of tasks created during a specific timeframe, the URL would be:

https://[environment].clickworker.com/api/marketplace/v2/customer/tasks?from=10.10.2013&to=14.10.2013

4.2 How to get tasks by customer reference

Within the examples of the Task-upload (see 3.2.1) is shown that customers can pass-through their own identifier if desired. With this identifier all tasks from a specific customer can be filtered using the following URL:

https://[environment].clickworker.com/api/marketplace/v2/customer/tasks/search?customer_ref=[customer_ref].

As a result the same list of tasks is presented to the customer as described in the section before.

5 Form elements

As described in 3.1.3 the <form> element of the Task-template defines the element that carries the in- and output for each text creation process.

For example there is an output element defined:

```
<elements>
  <is_output>true</is_output>
  <titles>
  <de>Textblock</de>
  <en>Paragraph</en>
  </titles>
  <is_mandatory>true</is_mandatory>
  <type>text_area</type>
  <item_code>paragraph_bottom</item_code>
  <sort_order>30</sort_order>
</elements>
```

As the customer wanted this element to carry text data the <type> here has been defined as ‘text_area’. There are several other ‘types’ available to carry specific types of data. This is true for input and for output elements.

The following table lists all available types of form elements and explains their specific use case:

<type>	**use case**	**options**	
text_field	In- or output of single line text.	is_mandatory	Field must be filled (valid for input elements during Task-creation or for output elements during result creation)?
		min_length	Minimum input length in characters.
		max_length	Maximum input length in characters.
text_area	In- or output of multi line text.	is_mandatory	Field must be filled?
		min_length	Minimum input length in characters.
		max_length	Maximum input length in characters.
		is_richtext	Defines, whether this field supports HTML rich text. The default is “false”.
number	In- or output of numbers.	is_mandatory	Field must be filled?
		min_value	The minimum value allowed.
		max_value	The maximum value allowed.
date	In- or output of dates.	is_mandatory	Field must be filled?

		min_date	Defines the earliest date that can be entered.
		max_date	Defines the latest date that can be entered
keyword	Input of keywords which should be included in the text.	is_mandatory	Field must be filled?
		min_occurrence_ref	This option defines the Task's data item code to read from the minimal occurrence count of the keyword.
		max_occurrence_ref	This option defines the Task's data item code to read from the maximal occurrence count of the keyword.
		reference_code	Enter the code of a data item here to check for keywords only in this item. Otherwise, leave blank or use the special value "all" to scan for keywords in all suitable fields.
email	In- or output of eMail addresses.	is_mandatory	Field must be filled?
url	In- or output of clickable URLs.	is_mandatory	Field must be filled?
		with_protocol	Defines whether the *clickworker* has to enter the protocol prefix (e.g. http://)
drop_box multi_select check_box radio_button	In- or output of a selection list with multiple alternatives.	is_mandatory	Field must be filled?
		alternatives	List of alternatives to pick from. Every alternative has one attribute: • The alternatives value that will be used when setting the Form Element's value and label.

An example of <type> ‘drop_box’ is given as:

```
<elements>
  <type>drop_box</type>
  <titles>
    <en>Please choose</en>
    <de>Bitte wählen Sie</de>
  </titles>
  <item_code>selected_language</item_code>
  <options>
    <code>alternatives</code>

    <!-- ************first entry************ -->

    <value>
       <value>de</value>
    </value>

    <!-- ************second entry************ -->

    <value>
       <value>fr</value>
    </value>

  </options>
</elements>
```

6 FAQ-section

6.1 What if I don't do a buyoff?

Customers are supposed to execute the buyoff, in order to confirm results and ensure that quality does not deteriorate. Alternatively you may request sign-off to be done by clickworker on behalf of you for an extra fee.

We have the option to configure any API Project in a way that the buyoff is done by clickworkers. This way the created texts are 'ready for use' without the intermediate action of the customer. Get in touch with us for more information.

6.2 What is the maximum length of the Task-template description?

There is no hard limit for this. However we strongly recommend keeping the working instructions here as short and clear as possible. A clear and short working instruction has significant impact to the throughput and quality of the created texts.

6.3 Can TextCreate be configured to create more than one output-language as per Task-template?

No. If the output is needed for multiple language, a Task-template for each output language needs to be created.

6.4 Buyoff – How much time do I have for the buyoff or the rejection?

Tasks waiting for buyoff are blocked and waiting for completion. According to the clickworker terms and for fair treatment of our crowd, and approval will be **done automatically after 7 days**.

If this happens the customer gets the results either through the TASK_COMPLETED.

6.5 What if I cannot provide a callback URL for notifications?

The only option left is to use polling and request the task results manually by http-GET requests (see 3.4).

6.6 What is a cloned task and how is the notification handled for this?

Due to the process design of the Text product, there may be situations when a task submitted by the customer gets 'cloned' (we create a copy and restart the text creation process from the beginning). In this scenario the results will be created for another task-id. Since you will only hold the reference of the 'original' id, the parameter 'show_cloned_task=true' should be part of every task request (see 3.4). In the near future we want to ensure that the customer gets only known task ids. In the meanwhile we recommend using the <customer_ref> element (see 3.2.1) to associate the unknown task id with an initial text creation request.