**Integrating Codyt with CMS systems**

**A best practices guide**

Codyt is the recommended type of project when it comes to linking up Wordbee with a CMS system. This document enumerates key API methods you might want to use to implement a typical integration.

When we refer to “connector” in this document, we mean the software that sits in between Wordbee and the CMS system. For example the “connector” extracts data from the CMS and transfers these to Wordbee. The “connector” needs to be developed and is not part of the Wordbee offering.

Use this document in conjunction with the Wordbee API documentation as well as the Wordbee API reference page. The latter is accessible inside Wordbee Translator: click “Settings” then “Developer API”, then click the help link.

## Prerequisites

To avoid that the connector accesses jobs or other data that are not controlled by the connector, please setup the system as follows:

* Add a new custom field to projects such as “Handled by” (pickup list type, no custom text, optional).
* Add “My CMS” (such as “Sharepoint”, “Wordpress” etc.) to the picklist.
* The connector now needs the ID of the custom field (e.g. CustomStr16). Look up the ID in the custom field settings online or use the API method *(GET) settings/customfields?token={TOKENID}*.
* If the connector creates projects from templates, then make sure that all projects have this custom field set to “My CMS”. As such all the projects created will contain the proper custom field value.
* We further recommend that you name all template projects something like “CMS template – XYZ” to make their purpose clear to users. Once setup change the status of the template projects to “Archived”. It will make them read-only.

The use case supposes that translations are extracted from Wordbee once the workflow is finished. In some cases, you might want to override this behavior. For example, you might give a user the possibility to transfer a translated document to the CMS even when the workflow is not finished.

This can be implemented with the use of labels on a job. Whenever the label is set, it can be used by the connector as a request for transfer: Create a “Transfer to CMS” custom job label. Users can then flag documents that shall be transferred back to the CMS even when the workflow is still running.

* Create the label in Wordbee. It can be an on/off or a multiple choice label.
* Use *(GET) settings/customlabels?token={TOKENID}* to get the label codes required when filtering jobs by label.

## Sending files to Wordbee

|  |  |
| --- | --- |
| Create or find project | Find project with reference:  *(GET) projects?token={TOKENID}&count=1&reference={REFERENCE}*  If the project does not exist, create a new one:  *(POST) projects?token={TOKENID}*  *&projecttype=2*  *&clientcompanyid={CLIENT COMPANY ID}*  *&sourcelocale={SOURCELOCALE}*  *&targetlocales={TARGETLOCALES}*  *&deadline={DEADLINE}*  *&reference={ REFERENCE}*  *&status={STATUS}*  *&receiveddate={RECEIVEDDATE}*  *&templateprojectid={PROJECT ID}*  For In Progress set STATUS to value 0.  The client company id (your platform) can be hardcoded, it will never change but it may be different in the test and the production platform. Use API call *(GET)* *master/company?token={TOKENID}* which returns the platform “master” company id.  To get the ID of the template project, use the “projects” method with the “reference” parameter set.  The “receiveddate” is optional. If not specified it will be set to the current date/time. You might want to set the CMS date here. Do not forget to convert to UTC and ISO format (see API doc).  Note: Wordbee does not prevent you from creating multiple projects with the same reference. |
| Upload files to project | Once you have the project id, you can upload files to the source language folder in the project.  Use method:  *(POST) projects/{PROJECTID}/files/{LOCALE}/file?token={TOKENID}&name={NAME}*  *&overwrite={OVERWRITE}*  “Locale” is the language folder. For uploading source files specify the source language code.  The “name” parameter includes the relative path like in “myfile.doc”, “folder1/folder2/myfile.doc”. The system creates any folders if they do not yet exist.  Drop the “overwrite” parameter unless you want to replace a file (if the file is already marked for translation you cannot replace it anymore). |
| Mark file for translation | This operation will prepare the file for a workflow. It extracts the texts, stores them into the database and kicks off workflows. The workflow specified in the project will be assigned (e.g. this may attach a translation job to the file).  Use this method:  *(PUT) projects/{PROJECTID}/documents/codytdocument2?token={1}*  *&name={2}*  *&version={3}*  *&sourcelocale={4}*  *&targetlocales={5}*  *&parserdomain={6}*  *&parserconfig={7}*  *&workflow={8}*  *&workflowstart={9}*  *&docreference={10}*  *&nosegmentation={11}*  See the API doc “Projects and Jobs” / “Codyt” for the parameters.  2: The previously uploaded document (e.g. “folder1/folder2/myfile.doc”)  3: Optional document version (free text string, for display purposes only).  6: See documentation for the available codes or use *(GET) settings/parserdomains?token={TOKENID}* for available options.  7: The name of the text extraction settings (in Wordbee go to “Settings”, choose file format and view the list of settings.  8: Set “ProjectDefault”  9: Drop this parameter so that workflow starts immediately  10: Drop or see documentation.  11: Drop. Segmentation should always be done.  Return values:  If the method returns -1 you are done. Otherwise you must poll the system status in a loop until the operation is done (or fails). Check the API documentation. |
|  |  |
| Other | There are various API methods that you might find useful for this step:   * Other file management methods (delete, rename, move, zip, unzip, etc.). See API documentation, chapter “Document library” |

## Assigning suppliers/workers

Per each file inserted into the project, the system creates a workflow with all the jobs needed. For example, the system may have added a translation and a revision job for the uploaded file.

Depending on the workflow some, jobs are either automatically assigned or not. In the former case, there is nothing more to do and the workflows are ready. In the latter case you would need to assign the unassigned jobs with the API.

To successfully assign a job, you need to have this information:

1. The job id (mandatory)
2. The supplier company id (mandatory)
3. The supplier person id (mandatory if supplier is internal user / in-house staff)
4. The service id which tells the system how to calculate costs (optional)

The basic approach is:

* Find all unassigned jobs for the previously uploaded file by applying a filter on the project id, the language couple and the file name.
* Then assign the job to a company/person and optionally specify the cost id

The procedure may change obviously if you know in advance which jobs need to be assigned to whom. For example, if you know that all files of type XYZ need to be assigned to John. The following description is for cases where you do not know.

|  |  |
| --- | --- |
| Find yet unassigned jobs | Use method:  *(GET) projects/{PROJECTID}/tasks?token={TOKENID}*  *&sourcelocale={SOURCELOCALE}*  *&targetlocale={TARGETLOCALE}*  *&status={STATUS}*  *&filter={FILTER}*  The language parameters can be dropped.  Now set the “filter” to enumerate unassigned jobs for a file:  *&filter=(Status\_= 0) AND (DocumentName = ”{name of file}”)*  See doc “API Objects” > “APICodytTask” for other fields you can filter  For each job returned, note down these properties:   * BeeDocumentId (the unique ID of the previously uploaded file) * TaskId (the unique job identifier) * TaskCode (to know if it is a translation, revision, etc.)   You will need them to subsequently assign suppliers. |
| Assign jobs | To assign a job, use this method:  *projects/{PROJECTID}/documents/*  *{DOCUMENTID}/tasks/{TASKID}/suppliers?token={TOKENID}*  *&companyid={COMPANYID}*  *&personid={PERSONID}*  Simply specify the company id and the person id. The document id is the “BeeDocumentId” mentioned further up.  Assigning to company or person?  When assigning to inhouse staff specify both company id and person id. When assigning to external vendors specify company id only (person id is optional then).  Get list of all or just qualified suppliers  You cannot assign any company/person to any job. Supplier profiles specify who can carry out what kind of jobs in which languages. For example, a typical translator may not qualify to do proofreading.  To get all internal suppliers (aka inhouse staff):  *master/persons?token={TOKENID}&from={FROM}&count={COUNT}*  To get all external suppliers:  *(GET) companies?token={TOKENID}&from={FROM}&count={COUNT}*  *&issupplier=true*  To get just the internal and external suppliers that are qualified to be assigned to a given job:  *(GET) projects/{PROJECTID}/documents/{DOCUMENTID}/*  *tasks/{TASKID}/suppliers?token={TOKENID}* |
| Other | There are various other methods that may be useful: For example to find the supplied services per supplier, the pricing details, etc. |
| Missing | It is currently not possible to specify the “service id” when assigning a job with the API. The service id references the unitary cost and discounts to be applied.  In the current API, the system will automatically attach the cost details found in the supplier’s default price list.  This may be ok or not ok in your case. |

## Getting translated files – Scenario A

There are various ways how you can poll Wordbee for completed work. We propose one approach below.

Another approach would be to poll each project individually. This approach may be more efficient if the total number of projects, active at any point in time, is below 50. Though, it would be inefficient with several hundreds of active projects.

Poll frequency: Generally, we recommend avoiding polling too frequently. Some customers poll every 10 minutes, others once per hour.

|  |  |
| --- | --- |
| Find completed workflows | Get all completed workflows:  *(GET) projects/workflows?token={TOKENID}*  *&sourcelocale={SOURCELOCALE}*  *&targetlocale={TARGETLOCALE}*  *&status={STATUS}*  *&statusdatefrom={DATEFROM}*  *&filter={FILTER}*  *&from=0*  *&count=100*  Set {STATUS} to 1 for finished workflows only.  See API doc “API Objects” > “ApiCodytRevisionSetLocale” for the various values.  Use {DATEFROM} to check only workflows that were finished at or after this date. This avoids that you query all historical data.  Set filter to *&filter=ProjectCustomStrXX=”My CMS”* to filter only on projects that are managed by the connector. See Prerequisites chapter in this document.  This method returns each finished workflow together with the file name and the project id. You will also get the document id if you need it for tracking purposes.  If later, a user re-opens the workflow and closes it again, this method will return the workflow again because the status date will have been updated. |
| Find flagged files | Use this method to find unfinished workflows that had been flagged by a user to transfer to the CMS. We discussed that the user would use a “Custom label” for flagging.  *(GET) projects/tasks?token={TOKENID}*  *&sourcelocale={SOURCELOCALE}*  *&targetlocale={TARGETLOCALE}*  *&filter={FILTER}*  *&from=0*  *&count=100*  Use {DATEFROM} to check only workflows that were finished at or after this date. This avoids that you query all historical data.  Set filter to *&filter=(ProjectCustomStrXX=”My CMS”)* to filter only on projects that are managed by the connector. See Prerequisites chapter in this document.  Now set the “filter” to enumerate unassigned jobs for a file:  *&filter= … AND (Labels.Contains(”{label db code}”)*  The “label db code” is obtained from the API: Call *(GET) settings/customlabels?token={TOKENID}* and look up the label in the list. Navigate to the option you want to filter for. <DBCode> contains the db value.  If required, set a “filter” on the workflow status. This example excludes completed workflows. To be seen if such filter is required for the use case.  *&filter= … AND (WorkflowStatus\_ != 1)*  See API doc “API Objects” > “ApiCodytRevisionSetLocale” for the various values.  Set filter on the workflow update date. To reduce the load on our database system, please indicate the UTC date of your last poll. The date time below is UTC.  *&filter= … AND (WorkflowStatusDate >= DateTime(2013, 5, 20)*  Alternatively you may want to filter on other properties such as the job status.  This method returns file name and project id. You will also get the document id if you need it for tracking purposes. |
| Clear flags/labels | To clear flagged files (i.e. reset the custom label), use this method:  *(PUT) projects/tasks/{TASKID}?token={TOKENID}*  *&command={command}*  {command} is a JSON object to update task fields.  Set a label: { “labels”: [ { “id”: 10, “value”: 1 }, … ] }  Clear a label: { “labels”: [ { “id”: 10, “value”: null }, … ] } |
| Get list of translated files | To find the deliverable or an attachment in the target languages, use this method:  *(GET) projects/{projectid}/filegroup?token={tokenid}*  *&locales={locales}*  *&name={name}*  *&documentid={documentid}*  Specify either {name} or {documentid}. No need to set both.  Locales can be one or more languages such as “en”, “en,fr,es” to select a single or multiple languages. Include the source language to find attachments to the source document.  The result is a list:   * It can contain either one or zero deliverables (IsAttachment = false) * It can contain zero, one or more attachments (IsAttachment = true)   You need to decide which of the translation files to send to the CMS. The record with IsAttachment = false should be given preference.  Example where the translation in English consists of a deliverable and one attached file.  <ArrayOfApiFileGroupItem>  <ApiFileGroupItem>  <Date>2013-05-17T10:38:33.8096895Z</Date>  <FileEncoding>utf-8</FileEncoding>  <FileName>**folder1\file.doc**</FileName>  <FormatInfo>.doc</FormatInfo>  <IsAttachment>**false**</IsAttachment>  <IsSource>false</IsSource>  <Locale>en</Locale>  </ApiFileGroupItem>  <ApiFileGroupItem>  <Date>2013-05-17T10:39:09.4047254Z</Date>  <FileEncoding i:nil="true"/>  <FileName>**folder1\file.doc\_\mydeliverable.doc**</FileName>  <FormatInfo i:nil="true"/>  <IsAttachment>**true**</IsAttachment>  <IsSource>false</IsSource>  <Locale>en</Locale>  </ApiFileGroupItem>  Once you have the file(s) to return, you can proceed to download them. |
| Download files | Use this method to download files:  *(GET) projects/{PROJECTID}/files/{LOCALE}/file?token={TOKENID}*  *&names={NAMES}*  See API documentation “Document library” > “File management” for details.  *Note: Download one file at a time (and not a list) because Zip files do not well support archives with non-ASCII file names.* |

If you want the connector to reconstruct the translated file automatically, use the method below. Otherwise, make sure that the reviser/proofreader or project manager, set the reconstructed translated file as part of the workflow.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Automatically build translated file | This method constructs the translated file from the translated segments. It builds the file and stores it to the deliverable location.  Please note that building the translated file may fail for various reasons: The most frequent is that a user forgot to fix markup. Therefore, a mechanism must be put in place to notify the user of a “problem” and ask for action. Before calling this method, you should check if the deliverable was not already uploaded by the user (or from the API).  Use this method to download files:  *projects/{PROJECTID}/files/{LOCALE}/file/translation?token={TOKENID}*  *&name={NAME}&targetlocale={TARGETLOCALE}&targetname={TARGETNAME}*  *&targetEncoding={TARGETENCODING}&targetFormat={TARGETFORMAT}*   |  |  |  | | --- | --- | --- | | tokenid | The connection token | required | | projectid | The project id | required | | locale | The original file's source language | required | | targetlocale | The language of the translated file | required | | targetname | If empty then the translated document will be saved with the same path/name as the original document (default behavior). Otherwise specify a target path/name. The file extension must match that of the original document. | optional | | targetEncoding | Do not set by default. Permits to customize the file encoding of the target file for certain document formats such as web pages. | optional | | targetFormat | Do not set by default. Permits to customize internal file format options depending on document format. Contact Wordbee for more information. | optional | |