Odoo provides a service allowing you to automate the processing of your invoices. The service scans your document using an Optical Character Recognition (OCR) engine and then uses AI-based algorithms to extract the fields of interest such as the total, the due date, or the invoice lines. More functional information can be found on the demo page (https://www.odoo.com/page/invoice-automation).
This service is a paid service. Each invoice processing will cost you one credit. Three different sized packs can be bought on iap.odoo.com (https://iap.odoo.com/iap/in-app-services/259?sortby=date).
You can either use this service directly in the Odoo Accounting App or through the API. The Extract API which is detailed in the next section allows you to integrate our service directly into your own projects.

Invoices

The extract API use the JSON-RPC2 (https://www.jsonrpc.org/specification) protocol. The diffent routes are located at the following address: **https://iap-extract.odoo.com**.

Expected successful flow

1. Call /iap/invoice_extract/parse to submit your invoices (one call for each invoice). On success, you receive a `document_id` in the response.
2. You then have to regularly poll /iap/invoice_extract/get_results to get the document's parsing status.
3. Once the result received, you can validate it by calling /iap/invoice_extract/validate and sending the expected values. This step is optional but greatly helps the system to improve.

These 3 routes are detailed in this section. The HTTP POST method should be used for all of them. A python implementation of the full flow can be found here (../_downloads/ca8fc9e87f0da0b974fa1b9c014ecbc0/extract_api_implementation.py) and a token for integration testing is provided in the integration testing section.

Routes

`/iap/invoice_extract/parse`

Description

Request a processing of the document from the OCR. The route will return a **`document_id`** you can use to obtain the result of your request.

Request Body

`jsonrpc` (required)
Must be exactly “2.0”.

`method` (required)
Must be “call”.

`id` (required)
An identifier established by the client. It allows the client to keep track of which response goes with which request. This makes asynchronous calls easier.

`params`

`account_token` (required)
The token of the account from which credits will be taken. Each successful call costs one token.

`version` (optional)
The version will determine the format of your requests and the format of the server response. Some results can be unavailable in older versions. For the current version 1.2.0, send ‘120’. If not specified, the latest version will be used.

`documents` (required)
The invoice must be provided as a string in the ASCII encoding. The list should contain only one string. If multiple strings are provided only the first string corresponding to a pdf will be processed. If no pdf is found, the first string will be processed. This field is a list only for legacy reasons. The supported extensions are *pdf*, *png*, *jpg* and *bmp*.

`user_infos` (required)
Information concerning the person to whom the invoice is intended. This informations is not required in order for the service to work but it greatly improves the quality of the result.

`user_company_vat` (optional)
VAT number of the client.

`user_company_name` (optional)
Name of the client’s company.

`user_company_country_code` (optional)
Country code of the client. Format: ISO3166 alpha-2 (https://www.iban.com/country-codes).

`user_lang` (optional)

The client language. Format: *language_code + _ + locale* (ex: fr_FR, en_US).

`user_email` (optional)

The client email.

JSON

```
{
    "jsonrpc": string,
    "method": string,
    "params": {
            "account_token": string (hex),
            "version": int,
            "documents": [string],
            "user_infos": {
                    "user_company_vat": string,
                    "user_company_name": string,
                    "user_company_country_code": string,
                    "user_lang": string,
                    "user_email": string,
            },
    },
    "id": string (hex),
}
```

Response

`jsonrpc`

A string specifying the version of the JSON-RPC protocol. It will be "2.0".

`id`

The identifier you set in the request body.

`result`

`status_code`

The code indicating the status of the request. **`status_code`** is 0 in case of success. Other **`status_code`** are detailed in the table below.

`status_msg`

A string giving verbose details about the request status.

`document_id`

Only present if the request is successful.

> The API does not actually use the JSON-RPC error scheme. Instead the API has its own error scheme bundled inside a successful JSON-RPC result.

status_code	status_msg
0	Success
2	An error occured

status_code	status_msg
3	You don't have enough credit
6	Unsupported file format
9	Server is currently under maintenance. Please try again later.

JSON

```
{
    "jsonrpc": string,
    "id": string,
    "result": {
        "status_code": int,
        "status_msg": string,
        "document_id": int,
    }
}
```

/iap/invoice_extract/get_results

Description

Request the results of the documents ids obtained with the /parse route. Can either return the results or a "request pending" message.

Request Body

`jsonrpc` (required)
Same as for /parse.

`method` (required)
Same as for /parse.

`id` (required)
Same as for /parse.

`params`:

`version` (required)
Same as for /parse.

`documents_ids` (required)
The list of `document_id` for which you want to get the current parsing status.

JSON

```
{
    "jsonrpc": string,
    "method": string,
    "params": {
        "version": int,
        "documents_ids": [int]
    },
    "id": string (hex),
}
```

Response

`jsonrpc`

Same as for /parse.

`id`

Same as for /parse.

`result`

Dictionary where each key is a document_id. For each **`document_id`** :

`status_code`

The code indicating the status of the request. **`status_code`** is 0 in case of success. Other **`status_code`** are detailed in the table below.

`status_msg`

A string giving verbose details about the request status.

`results`

Only present if the request is successful.

> **⚠ Warning**
> result keys are strings despite the fact that the document_ids given in the request body are integers.

status_code	status_msg
0	Success
1	Not ready
2	An error occured
9	Server is currently under maintenance. Please try again later.

JSON

```
{
    "jsonrpc": string,
    "id": string,
    "result": {
        "document_id_1": {
            "status_code": int,
            "status_msg": str,
            "results": [{"feature_1_name": feature_1_result,
                         "feature_2_name": feature_2_result,
                         …
                        }]
            },
        "document_id_2": {
            "status_code": int,
            "status_msg": str,
            "results": [{"feature_1_name": feature_1_result,
                         "feature_2_name": feature_2_result,
                         …
                        }]
            },
            ...
    }
}
```

feature_result

Each field of interest we want to extract from the invoice such as the total or the due date are also called features. An exhaustive list of all the extracted features can be found in the table below.

For each feature, we return a list of candidates and we spotlight the candidate our model predicts to be the best fit for the feature.

selected_value

The best candidate for this feature.

words

List of all the candidates for this feature ordered by decreasing score.

JSON

```
{
    "selected_value": candidate_12,
    "words": [candidate_12, candidate_3, candidate_4,...]
}
```

candidate

For each candidate we give its representation and position in the document. Candidates are sorted by decreasing order of suitability.

content

Representation of the candidate.

coords

`[center_x, center_y, width, height, rotation_angle]` . The position and dimensions are relative to the size of the page and are therefore between 0 and 1. The angle is a clockwise rotation measured in degrees.

page

Page of the original document on which the candidate is located (starts at 0).

JSON

```
{
    "content": string|float,
    "coords": [float, float, float, float, float],
    "page": int
}
```

Feature name	Specifities

`SWIFT_code`	**content** is a dictionary encoded as a string.
	It contains information about the detected SWIFT code (or BIC (https://www.iso9362.org/isobic/overview.html)).
	Keys:
	`bic`
	detected BIC (string).
	`name` **(optional)**
	bank name (string).
	`country_code`
	ISO3166 alpha-2 country code of the bank (string).
	`city` **(optional)**
	city of the bank (string).
	`verified_bic`
	True if the BIC has been found in our DB (bool).
	Name and city are present only if verified_bic is true.

`VAT_Number`	**content** is a string
`country`	**content** is a string
`currency`	**content** is a string
`date`	**content** is a string Format : *YYYY-MM-DD HH:MM:SS*
`due_date`	Same as for `date`
`global_taxes`	**content** is a float **candidate** has an additional field `amount_type` . Its value is always percent. **selected_values** is a list of candidates.

Feature name	Specifities
`global_taxes_amount`	**content** is a float
`invoice_id`	**content** is a string
`subtotal`	**content** is a float
`total`	**content** is a float
`supplier`	**content** is a string

`feature_result` for the `invoice_lines` feature

It follows a more specific structure. It is basically a list of dictionaries where each dictionary represents an invoice line. Each value follows a feature_result structure.

JSON

```
[
    {
        "description": feature_result,
        "discount": feature_result,
        "product": feature_result,
        "quantity": feature_result,
        "subtotal": feature_result,
        "total": feature_result,
        "taxes": feature_result,
        "total": feature_result,
        "unit": feature_result,
        "unit_price": feature_result
    },
    ...
]
```

`/iap/invoice_extract/validate`

Description

Route that validates the different features of an invoice. The validation step is an optional step but is strongly recommended. By telling the system if it were right or wrong for each feature you give an important feedback. It has no direct impact but it helps the system to greatly improve its prediction accuracy for the invoices you will send in the future.

Request Body

`jsonrpc` (required)

Same as for /parse.

`method` (required)

Same as for /parse.

params

documents_id (required)

Id of the document for which you want to validate the result.

values

Contains the validation for each feature. The field **merged_line** indicates if the **invoice_lines** have been merged or not.

> You don't have to validate all the features in order for the validation to succeed. However **/validate** can't be called multiple times for a same invoice. Therefore you should validate all the features you want to validate at once.

JSON

```
{
    "jsonrpc": string,
    "method": string,
    "params": {
        "document_id": int,
        "values": {
            "merged_lines": bool
            "feature_name_1": validation_1,
            "feature_name_2": validation_2,
            ...
        }
    },
    "id": string (hex),
}
```

validation

A **validation** for a given feature is a dictionary containing the textual representation of the expected value for this given feature. This format apply for all the features except for **global_taxes** and **invoice_lines** which have more complex validation format.

JSON

```
{ "content": string|float }
```

validation for **global_taxes**

content is a list of dictionaries. Each dictionary represents a tax:

amount

Amount on which the tax is applied.

tax_amount

Amount of the tax.

tax_amount_type

Indicates if the **tax_amount** is a percentage or a fixed value. The type must be specified using the literal string "fixed" or "percent".

tax_price_include

Indicates if **amount** already contains the tax or not.

JSON

```
{"content": [
    {
        "amount": float,
        "tax_amount": float,
        "tax_amount_type": "fixed"|"percent",
        "tax_price_include": bool
    },
    ...
]}
```

validation for `invoice_lines`

lines is a list of dictionaries. Each dictionary represents an invoice line. The dictionary keys speak for themselves.

JSON

```
{"lines": [
    {
        "description": string,
        "quantity": float,
        "unit_price": float,
        "product": string,
        "taxes_amount": float,
        "taxes": [
                    {
                        "amount": float,
                        "type": "fixed"|"percent",
                        "price_include": bool
                    },
                    ...
                ],
        "subtotal": float,
        "total": float
    },
    ...
]}
```

Response

`jsonrpc`

Same as for /parse.

`id`

Same as for /parse.

`result`

`status_code`

The code indicating the status of the request. `status_code` is 0 in case of success. Other `status_code` are detailed in the table below.

`status_msg`

A string giving verbose details about the request status.

status_code	status_msg
0	Success
12	Validation format is incorrect

JSON

```
{
    "jsonrpc": string,
    "id": string,
    "result": {
        "status_code": int,
        "status_msg": string,
    }
}
```

Integration Testing

You can test your integration by using *integration_token* as **`account_token`** in the /parse request.
Using this token put you in test mode and allows you to simulate the entire flow without really parsing a document and without being billed one credit for each successful invoice parsing.
The only technical differences in test mode is that the document you send is not parsed by the system and that the response you get from /get_results is a hard-coded one.
A python implementation of the full flow can be found here (../_downloads/ca8fc9e87f0da0b974fa1b9c014ecbc0/extract_api_implementation.py).