

SwiftCorrect Installation and User Guide

SwiftCorrect Overview

The Syncer and Verifier

- **Syncer:** Automatically updates vendor data (names, IDs) from CSV files with Vantage's catalogue based on your preferred schedule.
- **Verifier:** Uses fuzzy searches to match vendor names/IDs, automatically corrects guesses, provides easy-to-review corrections, saves both original and corrected invoice details, and improves search accuracy over time.

What's Inside the Verifier?

3 Types of Data:

- Original data (what Vantage found).
- Suggested data (from fuzzy search).
- Corrected data (if the user fixes it).

How This Helps You:

- Fix ERP Data: Link corrected vendor names/IDs to invoice details. Export clean lists to update old ERP data.
- Auto-Correct Future Invoices: Uses past corrections to avoid repeating mistakes.
- Improve Fuzzy Search: Gets better as it learns from user fixes.

Installation Steps

Configuration File

Adjust the configuration in the config.toml file. Open the config.toml file you will find:

```
debug = true
log-dir = '.'
passphrase = 'The-big-secret'
verbose = true
```

```
[vantage]
client-id = "
client-secret = "
tenant-id = "
```

```
username = "  
password = "  
  
[database]  
name = 'SwiftCorrect.db'  
  
[sync-csv-suppliers-to-abbyy-and-database]  
filepath = '.'  
catalogid = 'SwiftCorrectSuppliersCSV'  
  
[server]  
port = 7777
```

You can change the following configurations

ABBYY [Vantage] Connection Credentials

- Client-id
The OAuth client identifier assigned when you registered your integration app in ABBYY Vantage.
- Client-secret
The secret key paired with your Client ID for OAuth authentication.
- Tenant-id
The unique GUID representing your ABBYY Vantage tenant.
- Username
The email address used to log in to your ABBYY Vantage account.
- Password
The password associated with that ABBYY Vantage user.

Target Data Catalog Name and File Path [sync-csv-suppliers-to-abbyy-and-database]

- catalogid

SwiftCorrect is expecting the data catalogue name to be SwiftCorrectSuppliersCSV if you want to change the data catalogue name, write the expected name here between the single quotes

- filepath

The sync service reads from a CSV file containing the records to be imported. Provide:

File path is the same where the SwiftCorrect is installed. Its name is SwiftCorrectSuppliersCSV.csv. If you want to change the file path or file name write the expected name here between the single quotes

[Server] Configurations

- Port
Change to the server porty number of your choice. Currently, it is 7777

CSV Format Requirements

- The file must follow the structure of the “Document Issuer Companies” catalog.
- Required columns (no null values permitted):
 - Issuer Company ID
 - Name
 - Country

Example CSV header:

Issuer Company ID, Name, City, Street, State or Province, Country, Postal Code, Tax ID, National Tax ID, Bank Account, Bank Code, IBAN, Company Correlation ID

Installing HTTPS

ABBYY Vantage enforces the use of **HTTPS** to communicate with external end points. Use a reverse proxy like Nginx or Apache to expose the server port and enable secure HTTPS access with a server certificate.

How to Run the Server for the First Time

Running the Server

Use one of the following commands to run the server, depending on your needs:

Run Locally (Temporary Run):

```
./SwiftCorrect --workdir . --debug --verbose
```

Install as a Service (Persistent):

- On Linux:
Open a terminal and run the following command:

```
./SwiftCorrect --workdir . --debug --verbose --service install
```
- On Windows:
Open **Command Prompt as Administrator**, then run:

```
SwiftCorrect --workdir . --debug --verbose --service install
```

Important: Make sure you run Command Prompt **as Administrator**, or the service installation will fail.

Important:

Use the --debug or --verbose flags only, when necessary, based on technical or business requirements.

Verifying the Server

Once the server starts, check the logs. You will see an HTTP URL like the following:
`http://localhost:7777`

Instructions for Using Fuzzy Search and Vendor Confirm from a Vantage Process

These instructions guide you through installing the server, configuring custom activities in the ABBYY Vantage Process Skill Editor, and adding code for fuzzy search and vendor confirmation.

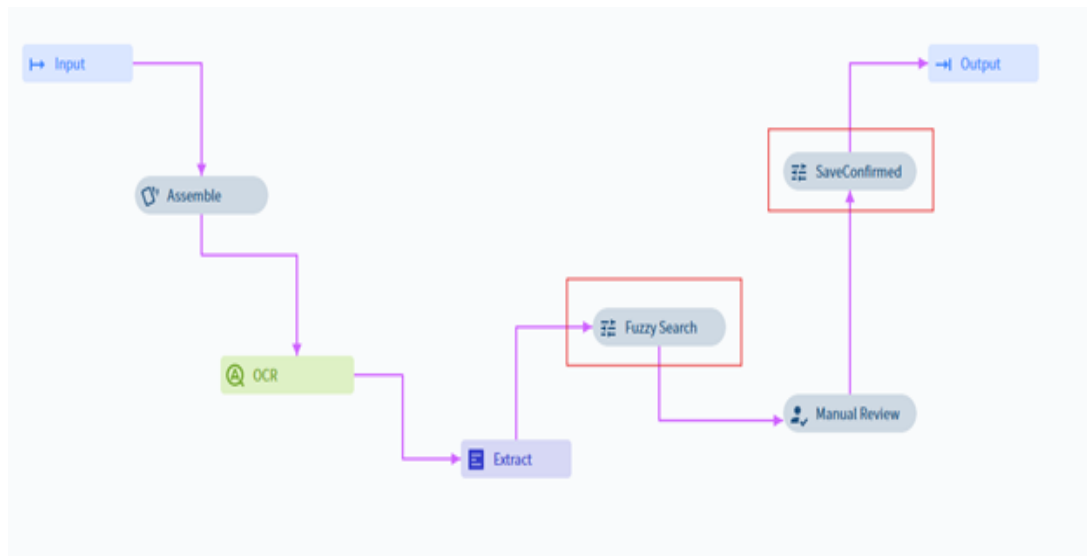
Install the Server

1. Deploy the server application on the customer's machine.
2. After installation completes, obtain the server URL (e.g., <https://localhost:8080>).

Configure Custom Activities in the Process Skill Editor

1. Open the Process Skill Editor in your application.
2. Create two new Custom Activities:
 - **Fuzzy Search Activity:** This activity performs approximate matching to find potential vendor records.
 - **Save Confirmed Vendor Activity:** This activity updates the system with the vendor selected from the fuzzy search results.

Refer to the following illustration for guidance:



Add Code to the Fuzzy Search Activity

In the **Fuzzy Search Activity**, insert the following code snippet to implement vendor lookup using fuzzy matching:

```

for (let i = 0; i < Context.Transaction.Documents.length; i++) {
    let vendorNameField = Context.Transaction.Documents[i].GetField("Vendor/Name");
    if (vendorNameField && vendorNameField.Value) {
        vendorNameField.Value = vendorNameField.Value.toString()
            .replace(/\r\n/g, ' ')
            .replace(/\s+/g, ' ')
            .trim();
        var req = Context.CreateMultipartFormDataRequest();
        req.Url = "<<SERVERURL>>";
        req.Method = "POST";
        req.SetHeader("Authorization", "Basic
NGE1NTJ2TUZYzIEY5ZDRlMzhlLTgxMTEtNDJfMC04MzE2NGNiLWI2YWY4NDC2ZHI1N0ItMTU0N0JlMzU
3Mjk5OTFCOQ==");
        let transactionID = Context.Transaction.Id;
        let vendorTaxIDField = Context.Transaction.Documents[i].GetField("Vendor/Tax ID");
        let vendorAddressField = Context.Transaction.Documents[i].GetField("Vendor/Address");
        let vendorCountryField = Context.Transaction.Documents[i].GetField("Vendor/Country");
        let vendorCityField = Context.Transaction.Documents[i].GetField("Vendor/City");
        let vendorPostalField = Context.Transaction.Documents[i].GetField("Vendor/Postal Code");
        let vendorIDField = Context.Transaction.Documents[i].GetField("Vendor/ID");
        var content = JSON.stringify({
            transactionID: transactionID,
            fields: {
                "Vendor ID": vendorIDField.Value,
                "Vendor Name": vendorNameField.Value,
                "Vendor Tax ID": vendorTaxIDField.Value,
                "Vendor Address": vendorAddressField.Value,
                "Vendor Country": vendorCountryField.Value,
                "Vendor City": vendorCityField.Value,
                "Vendor Postal Code": vendorPostalField.Value
            }
        });
        req.AppendStringContent(content, "vendorPayload");
        req.Send();
        var jsonResponse = JSON.parse(req.ResponseText);
        var updatedFields = jsonResponse.fields;
        if (updatedFields) {
            if (updatedFields["update"] === false) {
                // If updated is false, only update Vendor Name and Vendor ID
                if (updatedFields["Vendor Name"] !== undefined) vendorNameField.Value = updatedFields["Vendor
Name"];
                if (updatedFields["Vendor ID"] !== undefined) vendorIDField.Value = updatedFields["Vendor ID"];
            } else {
                if (updatedFields["Vendor Name"] !== undefined) vendorNameField.Value = updatedFields["Vendor
Name"];
                if (updatedFields["Vendor Tax ID"] !== undefined) vendorTaxIDField.Value = updatedFields["Vendor
Tax ID"];
                if (updatedFields["Vendor Address"] !== undefined) vendorAddressField.Value = updatedFields["Vendor
Address"];
                if (updatedFields["Vendor Country"] !== undefined) vendorCountryField.Value = updatedFields["Vendor
Country"];
                if (updatedFields["Vendor City"] !== undefined) vendorCityField.Value = updatedFields["Vendor City"];
                if (updatedFields["Vendor Postal Code"] !== undefined) vendorPostalField.Value =
updatedFields["Vendor Postal Code"];
                if (updatedFields["Vendor ID"] !== undefined) vendorIDField.Value = updatedFields["Vendor ID"];
            }
        }
    }
}

```

```

    }
  }
}

```

Add Code to the Save Confirmed Vendor Activity

In the **Save Confirmed Vendor Activity**, insert the following code to save the selected vendor to your system:

```

for (let i = 0; i < Context.Transaction.Documents.length; i++) {
  let vendorNameField = Context.Transaction.Documents[i].GetField("Vendor/Name");

  if (vendorNameField && vendorNameField.Value) {
    vendorNameField.Value = vendorNameField.Value.toString()
      .replace(/\r\n/g, ' ')
      .replace(/^\s+/, '')
      .trim();

    var reqVendor = Context.CreateMultipartFormDataRequest();
    reqVendor.Url = "<<ServerURL>>"
    reqVendor.Method = "POST";
    reqVendor.SetHeader("Authorization", "Basic
NGE1NTJ2TUZYzIEY5ZDRlMzhlLTgxMTEtNDJfMC04MzE2NGNiLWI2YWY4NDc2ZHI1N0ltMTU0N0JlMzU
3Mjk5OTFCOQ==");

    let transactionID = Context.Transaction.Id;
    let vendorTaxIDField = Context.Transaction.Documents[i].GetField("Vendor/Tax ID");
    let vendorAddressField = Context.Transaction.Documents[i].GetField("Vendor/Address");
    let vendorCountryField = Context.Transaction.Documents[i].GetField("Vendor/Country");
    let vendorCityField = Context.Transaction.Documents[i].GetField("Vendor/City");
    let vendorPostalField = Context.Transaction.Documents[i].GetField("Vendor/Postal Code");
    let vendorIDField = Context.Transaction.Documents[i].GetField("Vendor/ID");

    var reqVendorContent = JSON.stringify({
      transactionID: transactionID,
      fields: {
        "Vendor ID": vendorIDField.Value,
        "Vendor Name": vendorNameField.Value,
        "Vendor Tax ID": vendorTaxIDField.Value,
        "Vendor Address": vendorAddressField.Value,
        "Vendor Country": vendorCountryField.Value,
        "Vendor City": vendorCityField.Value,
        "Vendor Postal Code": vendorPostalField.Value
      }
    });

    reqVendor.AppendStringContent(reqVendorContent, "vendorPayload");
    reqVendor.Send();
  }
}

```

Verify the Integration

Execute a test workflow that invokes both custom activities. Upload an invoice and manually review the results to confirm the correct vendor. On subsequent runs for the same extracted user, the system will automatically return the previously confirmed vendor.