ABBYY FlexiCapture

Connector for Appian Guide

Table of Contents

About the ABBYY FlexiCapture Connector for Appian

ABBYY FlexiCapture enables users to perform high-quality full-text recognition, classify documents, detect fields and capture their data, and convert files into various formats (e.g. searchable PDF).

ABBYY FlexiCapture Connector for Appian is intended for loading document files and captured data into the Appian.

Architecture

The ABBYY FlexiCapture Connector for Appian includes the following components:

- DmsConnector.dll, a library of classes used to convert data from an abstract DMS into ABBYY FlexiCapture data.
- AfcDmsConnector.dll, a library accessed by the ABBYY FlexiCapture export script. This library accepts ABBYY FlexiCapture objects as input, converts them, and passes them on to the ABBYY FlexiCapture Connector for Appian.
- AppianConnector.dll, a connector for Appian. This library provides a simplified programming interface to interact with the Appian server.
- Newtonsoft.Json.dll, a support library. This library provides functionality to work with JSON strings. It is used by AppianConnector.dll.
- ABBYYFlexiCapturePlugin.jar, a plug-in that allows you to load files into Appian folders.

The libraries listed above run on the ABBYY FlexiCapture side. The ABBYY FlexiCapture Connector for Appian also includes a **ABBYYFlexiCapturePlugin.jar** plug-in, which is installed on the Appian side and allows you to load files into Appian folders.

System requirements

The ABBYY FlexiCapture Connector for Appian supports:

- ABBYY FlexiCapture 12 Distributed or Standalone 12.0.2.1420.
- ABBYY FlexiCapture Cloud.
- Appian 17.1.

Export will be performed on the ABBYY FlexiCapture Processing Stations.

The ABBYY FlexiCapture Connector for Appian requires the following external component to be installed on all of the computers where export will be performed and on the Appian server:

Microsoft .NET Framework 4.5.

Important! To be able to export documents from FC to the Appian server, you need to open the port on the Appian server via which the web server can be accessed. By default, this is port 8080 (if HTTP is used) or port 443 (if HTTPS is used).

Installation

The ABBYY FlexiCapture Connector for Appian is distributed as a ZIP archive that includes all the required components and samples.

Configuring the export of data from ABBYY FlexiCapture to Appian

Before setting up the connector, please make sure that you have following software installed:

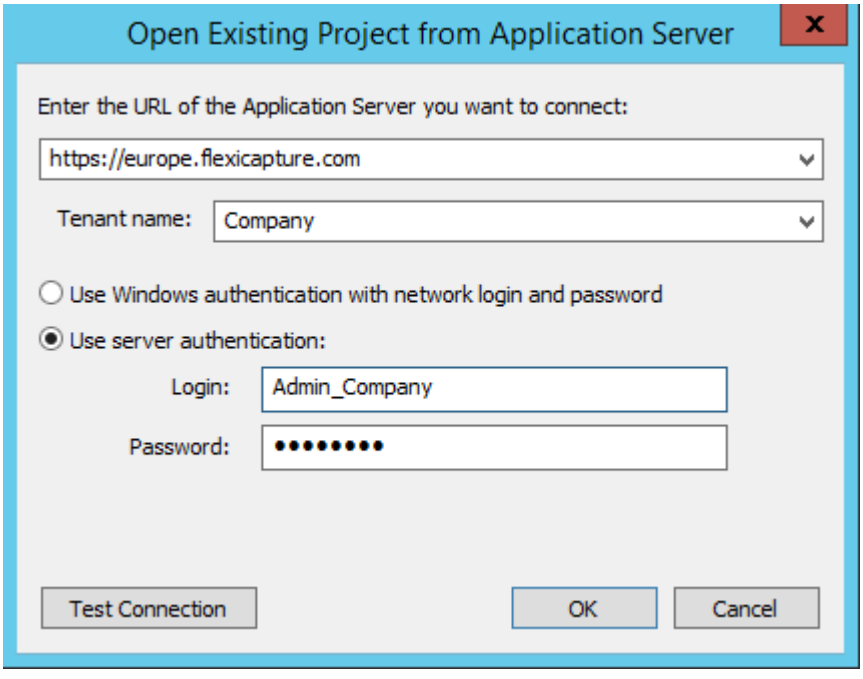
- ABBYY FlexiCapture 12 Project Setup Station (required if you are using ABBYY FlexiCapture Distributed Edition).

or

- Administrator Station (if you are using ABBYY FlexiCapture Standalone Edition).

You will also need access rights to Appian.

If you are using ABBYY FlexiCapture Cloud, you will additionally need the Project Setup Station desktop application in order to configure your project. To open a project uploaded to ABBYY FlexiCapture Cloud, open the Project Setup Station, open the **File** menu, and click **Open Project from Server…**. In the dialog box that open, specify the address of the ABBYY FlexiCapture server, specify the name of your company in the **Tenant name** field, select the **Use server authentication** option, and enter your credentials. Next, click **Test Connection** to check that you have entered the correct data. Finally, click **OK** to open the project.



The steps described above can also be used to upload a project from a local folder.

In ABBYY FlexiCapture, documents are processed within projects. ABBYY FlexiCapture will identify the type of each document and apply the appropriate Document Definition that has been prepared specifically for that type. Each Document Definition includes one or more export profiles to be used for exporting document images and data. If you are not familiar with the aforementioned concepts, please

refer to the ABBYY FlexiCapture documentation, where you will find a detailed description of the capture process.

To set up export of data from ABBYY FlexiCapture to Appian, complete the following steps:

- Connect the libraries of the ABBYY FlexiCapture Connector for Appian to your ABBYY FlexiCapture Document Definition.
- Create an export profile in your Document Definition.
- Specify export settings and map fields.

Connecting the libraries to a Document Definition

Before you can configure document export, you must connect the libraries to your ABBYY FlexiCapture project. To connect the libraries, complete the following steps:

1. Open the project in the ABBYY FlexiCapture **Project Setup Station** or the ABBYY FlexiCapture **Administrator Station**.
2. Click **Project** → **Document Definitions...** to open the list of Document Definitions available for the project, select the Document Definition for which you want to specify new export settings, and click **Edit...**.
3. Click **Document Definition** → **Document Definition Properties...** to open the properties of the Document Definition and then click the **.Net References** tab.
4. Click the **Add...** button on the **.Net References** tab. In the **Add Assembly** dialog box, select **Attached file** from the **Type** drop-down list, and in the **Reference** field, specify the path to the DLL file that you wish to add. Repeat this step for the **AfcDmsConnector.dll, DmsConnector.dll AppianConnector.dll**, and **Newtonsoft.Json.dll** files from the ZIP archive with the connector files.
5. Save the Document Definition to apply the changes.

Creating an export profile

ABBYY FlexiCapture exports data based on the export profiles in the Document Definitions.

To create an export profile for Appian:

1. On the Project Setup Station or on the ABBYY FlexiCapture Administrator Station, open your ABBYY FlexiCapture project.

Note: Make sure that all the necessary libraries are connected to the Document Definition.

2. Click **Project** → **Document Definitions...** to open the list of Document Definitions available for the project, select the Document Definition for which you want to specify new export settings, and click **Edit...**.
3. Click **Document Definition** → **Export Settings...**.
4. Click the **Add...** button to start the export profile creation wizard.

5. In the **Select the Type of Destination** step, select **Custom export (script)** from the **Type** drop-down list and then select **Errors are irrelevant** in the **Document condition** field. Leave the other settings unchanged or modify them as described in ABBYY FlexiCapture Help.

6. In the **Script Export** step, click the **Edit Script...** button, select **C# .Net** script language, and paste into the script editor one of the two sample scripts provided in the connector files: **ExportScript - Appian - Configure by script.cs** or **ExportScript - Appian - Configure by XML.cs**. For detailed instructions on using each of these scripts, see Using a script to specify export settings and Using an XML file to specify export settings below. You can always edit this script later, but for the moment, save the changes and close the script editor. Detailed instructions for setting up your export script will be provided below.

7. In the **Select the Destination Name** step, specify a name for your export profile and click **Finish**.

8. The newly created export profile will appear in the list of available export profiles in the export settings dialog box. To make ABBYY FlexiCapture use this profile on Export stage, select the box next to the profile in the **Enabled** column.

9. Click **OK** and close the Document Definition editor. Next, click the **Publish** button to publish your Document Definition.

Note: If documents are being processed within the project at the time when you add your new export profile, update them to the latest version by pressing **Alt+Shift+U** or by selecting **Update to Latest Version** from their shortcut menu.

Using a script to specify export settings

One advantage of using a script is that all the export settings will be automatically used by all the ABBYY FlexiCapture stations from which data are exported to Appian. When using an XML file, you can only specify static settings, whereas scripts allow you to specify different file naming rules, file locations, and field mappings depending on the data contained in the exported document.

When working with ABBYY FlexiCapture Cloud, you also need to use a script to specify export settings, since ABBYY FlexiCapture Cloud does not provide the ability to store an XML file with configuration on processing stations.

However, to modify the export settings, you will have to make changes to the script and, consequently, publish a new version of the Document Definition.

Please use the script named **ExportScript - Appian - Configure by script.cs** from the connector files.

Using an XML file to specify export settings

One advantage of using an XML file is that you don’t have to edit the script code or the Document Definition. All modifications can be made by editing the XML file, which is much simpler than editing the script code. However, the XML file should be located on every ABBYY FlexiCapture processing station. And you will also have to synchronize it manually on all of the ABBYY FlexiCapture stations from which data is to be exported to Appian.

To specify export settings in an XML file:

1. As your export script, use the **ExportScript - Appian - Configure by XML.cs** file from the ABBYY FlexiCapture Connector for Appian files.

2. Create an XML file named **<Project name>_<Document Definition name>.xml**. This naming scheme will allow you to have multiple XML files for different Document Definitions.
3. In the registry of the computer with a processing station, locate HKEY_LOCAL_MACHINE\SOFTWARE\ABBYY\FlexiCapture\12.0\Connectors\Appian and in its "XmlFolder" string value field, specify the path to the folder where the settings file is to be stored.
4. In the registry of the computer with a processing station, locate the HKEY_LOCAL_MACHINE\SOFTWARE\ABBYY\FlexiCapture\12.0\Connectors\ and set its "Appian" string value field to **true**.

Note: For an example of an XML configuration file with detailed comments, please see the **default.config.xml** file that is located in the ABBYY FlexiCapture Connector for Appian files.

Specifying export settings and field mapping

This section describes the fields of the **ExportScript - Appian - Configure by script.cs** script, and **default.config.xml** configuration file.

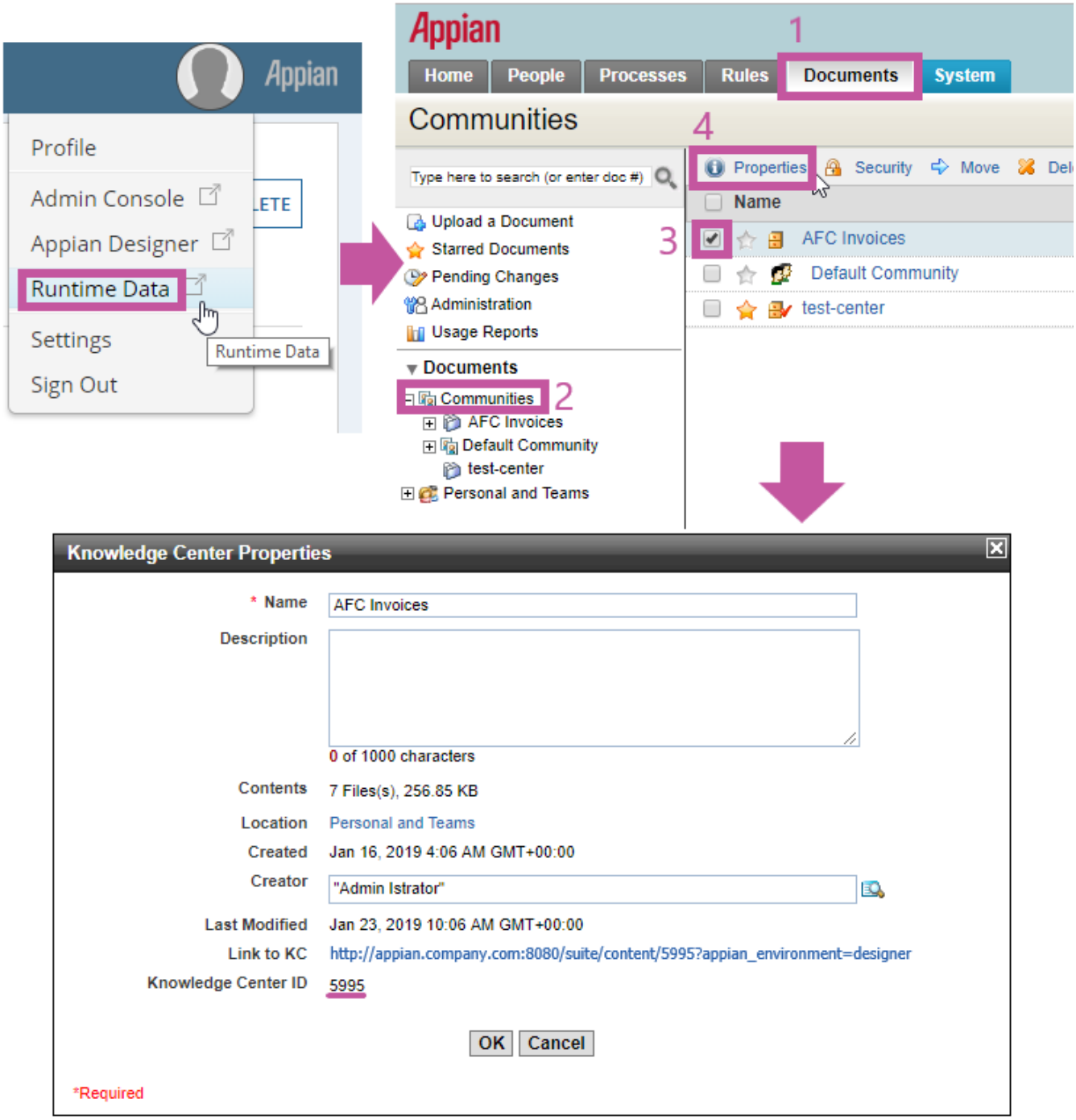
Server contains the address of Appian Server and the port used for the connection (the default port is 8080). For example, `<Server>http://computername.domain.com:8080</Server>`.

Username contains the name of the user in Appian. For example, `<Username>username</Username>`.

Password contains the user's password. For example, `<Password>password</Password>`.

Repository contains the identifier of the knowledge center in Appian. For example, `<Repository>5995</Repository>`.

Note: To look up the the ID of a knowledge center, click the **Runtime Data** in the workspace and select the desired knowledge center in the **Documents** section. At the top of the workspace, you will see the actions that can be performed on the selected knowledge center. Click Properties at the top of the workspace. A **Knowledge Center Properties** dialog box will open displaying the ID of the knowledge center:



FolderPath contains the name of the folder to be created in Appian. For example, `<FolderPath>&lt;Batch&gt;</FolderPath>`.

FileName contains the name of the document that will be created at the time of export. For example, `<FileName>&lt;Batch&gt;_&lt;Identifier&gt;</FileName>`.

Note: The name of the document may have an unchanging static part and a changing dynamic part. The dynamic part is defined by tags, which are replaced by the ABBYY FlexiCapture document property values at the time of export. Each tag must be enclosed in angle brackets. In the XML file:

- '<' stands for '<';
- '>' stands for '>'.

Possible tags for the dynamic part of the name:

- *Project* for the name of the project in ABBYY FlexiCapture;
- *Batch* for the name of the batch;
- *DocumentDefinition* for the name of the Document Definition in ABBYY FlexiCapture;
- *SourceFile* for the name of the source file;
- *Identifier* for the document identifier in the batch;
- *BatchType* for the name of the batch type;
- *Time* for the current time in yyyy-MM-dd_HH-mm-ss format.

Note: These dynamically replaceable tags are case-sensitive and the register of the characters used in the tags above should be preserved.

The following is an example of a document name with a static part:

```
<FileName>My_static_prefix_&lt;DocumentDefinition&gt;_&lt;Identifier&gt;and_postfix</FileName>
```

The following is an example of a document name without a static part:

```
<FileName>&lt;Project&gt;_&lt;Identifier&gt;</FileName>
```

DmsMetadataTemplate contains the name of the record type in plural form which will be visible in Tempo. Each ABBYY FlexiCapture Document Definition must have its own record type in Appian. For example,

```
<DmsMetadataTemplate>
  <Name>Account Payable Records</Name>
</DmsMetadataTemplate>
```

FieldPairs contains a collection of field pairs.

If your export settings are stored in an XML file, map the fields as follows:

```
<FieldPairs>
 <FieldPair>
   <AfcField>Invoice\InvoiceNumber</AfcField>
   <DmsField>InvoiceNumber</DmsField>
 </FieldPair>
</FieldPairs>
```

where:

- *AfcField* is the full path to the FlexiCapture Document Definition field from which data are to be read at the time of export;
- *DmsField* is Appian data type field, where data will be written at the time of export.

Note: Only for export settings specified in an XML file. For export settings specified in a script, map the fields as follows:

```
Dictionary<string, string> pairs = new Dictionary<string, string>()
{
    { @"Invoice Layout\InvoiceNumber", "InvoiceNumber" },
    ...
};
connectorSettings.MappedFields.FieldPairs = new MappedFields(pairs).FieldPairs;
```

where the first value is the full path to the field in FlexiCapture, and the second is a name of the field in data type.

Note: When mapping table values, be sure to create an auxiliary data type in Appian and map the fields as follows:

```
Dictionary<string, string> pairs = new Dictionary<string, string>()
{
    ...
    { @"Invoice Layout\LineItems\Description", @"LineItems\Description" },
    ...
};
connectorSettings.MappedFields.FieldPairs = new MappedFields(pairs).FieldPairs;
```

where the second value of LineItems is the name of the field in the main data type that stores the array of objects of the auxiliary data type, and Description is the name of the mapped field in the auxiliary data type.

Deploying the ABBYYFlexiCapture application on the Appian server

The ABBYY FlexiCapture Connector for Appian distribution package includes a sample Appian application named **ABBYYFlexiCapture.zip**. There are two pre-configured record types in the ABBYYFlexiCapture application: **Account Payable Records** to store data captured from invoices and **Credit Card Applications** to stored data captured from credit card applications.

To import ABBYYFlexiCapture onto the Appian server:

1. Install the **ABBYYFlexiCapturePlugin.jar** plug-in on the Appian server by copying it from the ABBYY FlexiCapture Connector for Appian distribution package into **APPIAN_INSTALL/_admin/plugins/**.

Note: To find out the version of the plug-in, open the **Admin Console** and click **SYSTEM** → **Plug-ins** in the left-hand pane. A page will open displaying the version of the plug-in next to its name (for example ABBYY FlexiCapture 12 Plug-in v1.0.0.44).

2. Open Appian Designer and click **Import Application**. In the dialog box that opens, select the **ABBYYFlexiCapture.zip** archive in the ABBYY FlexiCapture Connector for Appian distribution package and click **Import**.

Note: You may see the following error messages when importing the application:

- *dataStore _a-0000de8f-43d8-8000-3120-01ef9001ef90_3255 5984 "Invoices": The import of dataStore [id=5984 uuid=_a-0000de8f-43d8-8000-3120-01ef9001ef90_3255] could not be finalized:*

The data store "Invoices" [id=5984] was saved but cannot be published: The data source [jdbc/Appian] cannot be reached or its configuration is invalid. Details: javax.naming.NameNotFoundException: jdbc/Appian -- service jboss.naming.context.java.jdbc.Appian (DataSourceException) (APNX-1-4178-000) (APNX-1-4071-009).

- *dataStore _a-0000de8f-43d8-8000-3120-01ef9001ef90_3361 5985 "CreditApplicationsMain": The import of dataStore [id=5985 uuid=_a-0000de8f-43d8-8000-3120-01ef9001ef90_3361] could not be finalized: The data store "CreditApplicationsMain" [id=5985] was saved but cannot be published: The data source [jdbc/Appian] cannot be reached or its configuration is invalid. Details: javax.naming.NameNotFoundException: jdbc/Appian -- service jboss.naming.context.java.jdbc.Appian (DataSourceException) (APNX-1-4178-000) (APNX-1-4071-009).*

Despite these error messages, the ABBYYFlexiCapture application will been imported successfully. The error messages simply inform you that there is no data source named "jdbc/Appian" on the Appian server. You can always create this data source after importing the application (see the instructions below).

After you have imported the ABBYYFlexiCapture, complete the following steps:

1. Create a data source.
2. Publish the data stores. To do this, open the ABBYYFlexiCapture application in Appian Designer, select the **Data Store** option in the left-hand pane, and open the appropriate data store. In the dialog box that opens, select the previously created data source and click **Verify**. You will see a warning message saying "No matching tables found!" Select the **Create tables automatically** option and click **Save & Publish**.

Note: The ABBYYFlexiCapture application includes two data stores — "Invoices," intended for storing data from invoices, and "CreditApplicationsMain," intended for storing data from credit card applications. You will need to perform the above step for each of these data stores.

3. Create a knowledge center folder.

Creating objects in Appian applications

This section provides instructions for creating objects and modifying Appian applications.

Modifying the ABBYYFlexiCapture application

This section describes the modifications you can make to the sample ABBYYFlexiCapture application. To be able to store a different type of documents in the application, you need to create a new record type and make some other changes.

Creating an infrastructure for storing and displaying data in Appian

The purpose of a record type is to display information from data store in Tempo. Therefore, you need to create an infrastructure for storing and displaying data in Appian. To do this:

1. Create a database table (data type).
2. Create a data source.

3. Create a data store.
4. Create a record type.
5. Create a constant of type "Data Store Entity" to link the data store and the data type.
6. Edit the AFC_GetDataTypeByRecordTypeName and AFC_GetConstantDSEByRecordTypeName rules.
7. Create a folder of type "Knowledge Center" where to store processed files received from ABBYY FlexiCapture.

Creating a data type

1. In Appian Designer, open the ABBYYFlexiCapture application and click **New** → **Data Type**.
2. Select **Create from scratch**, complete the required **Name** field, and click **Create & Edit**.
3. In the dialog box that opens, create database table fields by clicking the **New Field** button. The ABBYY FlexiCapture Connector for Appian will be populating this table with data and then the record will be loading data from this table into Appian.

Important! Your table must contain a primary key field named "FolderId" of type "Number(Integer)". This field will function as a unique primary key and will link the data in the table with the knowledge center folder.

4. Before the data type table in Appian can accept table rows from ABBYY FlexiCapture, you must create an auxiliary data type for fields corresponding to table columns in the Document Definition. Next, create a field in the data type, specify the name of the auxiliary data type as its type, and select the **Array** option.
5. When you have made all the necessary changes to the table, click **Save**.

Creating a data source

1. Open the Appian Administration Console and click **Data Sources** in the **Integration** section on the left.
2. On the **Data Source Management** page that opens, click **New Data Source**.
3. In the **Configure Data Source** dialog box, complete the following required fields:
   - **Name** – a name for the new data source (e.g. jdbc/Appian);
   - **Type** – the type of the new data source (e.g. SQL Server);
   - **Username/Password** – the credentials of the database user;
   - **Connection String** – the string to be used to connect to the database (e.g. jdbc:sqlserver://appian.company.com:1433;databaseName=appian_db).
4. Click **Test Connection** to check that you have entered the correct data. If the test is successful, click **Save**.

Creating a data store

1. In Appian Designer, open the ABBYYFlexiCapture application and click **New** → **Data Store**.
2. Complete the **Name** field and click **Create & Edit**.
3. On the **Data Management** tab of the Data Store editor:
   a. Select a data source.
   b. Click the **Add Entity** button and add a data type.
   c. Click the **Verify** button to verify your data store.
   d. Click **Save & Publish**.

Creating a record type

1. In Appian Designer, open the ABBYYFlexiCapture application and click **New** → **Record Type**.
2. Complete the **Name** and **Plural Name** fields and click **Create & Edit**.
3. In the window that opens:
   a. In the **Data** section, select **Data Store Entity** and add your data store and your data type.
   b. Configure the display of your record list (see Configuring the record list).
   c. Configure the display of the **Summary** tab (see Configuring the record view).

Creating a constant of type "Data Store Entity"

This constant links your data store to your data type. When a new record is created, this link is used to determine the record type of the record.

1. In Appian Designer, open the ABBYYFlexiCapture application and click **New** → **Constant**. The constant will be created from scratch and its default type will be **Data Store Entity**.
2. Specify a name for your constant. The **Data Store** and **Data Type** fields will be populated by the previously created objects.
3. Save the changes to the default AFC Rules and Constants folder or to any other folder of your choice.

Editing the rules

Once you have added the new record type and constant, you need to modify the expression rule by adding conditions for the new record type.

1. In Appian Designer, open the ABBYYFlexiCapture application and select the **Expression Rule** option in the left-hand pane. Locate and click the **AFC_GetConstantDSEByRecordTypeName** rule to start editing it.
2. In the code editor, replace "null" with the following "if" condition for the new record type and constant:

```
if(ri!RecordTypeName == "Account Payable Records",
    cons!Constant_InvoiceProsessing_DSE,
    if(ri!RecordTypeName == "Credit Card Applications",
        cons!Constant_CreditApplication_DSE,
        if(ri!RecordTypeName == "New Records", /* In RecordTypeName, specify the name of
the new record type in the plural. */
            cons!New_Constant, /* After cons!, specify the name of the constant you created
earlier. */
            null
        )
    )
)
```

If an input RecordTypeName is the same as the name of a stored record type (in the plural), the DataStoreEntity constant will be returned that links the data store and the data type.

3. Now you need to edit the **AFC_GetDataTypeByRecordTypeName** rule. In Appian Designer, open the ABBYYFlexiCapture application and select the **Expression Rule** option in the left-hand pane. Locate and click the **AFC_GetDataTypeByRecordTypeName** rule to start editing it. In the code editor, replace the "null" with the following "if" condition for the new record type and the full name of the data type:

```
if(ri!RecordTypeName == "Account Payable Records",
    'type!{urn:com:appian:types}Invoice_Header',
    if(ri!RecordTypeName == "Credit Card Applications",
        'type!{urn:com:appian:types}Credit_ApplicationMain',
        if(ri!RecordTypeName == "New Records", /* In RecordTypeName, specify the name of
the new record type in the plural. */
            'type!{urn:com:appian:types}New_Data_Type', /* In the type! field, specify the
full name of the data type and the namespace. */
            null
        )
    )
)
```

If an input RecordTypeName is the same as the name of a stored record type (in the plural), a string will be returned containing the full name of the data type you created (you can look up the namespace in the data type properties). To display the properties of a data type, open the desired data type in the ABBYYFlexiCapture application. Clicking the name of the data type in the top left corner will open the **Data Type Properties** window where you can see the namespace.

Creating a knowledge center

To store document images in Appian, you need to create a knowledge center folder.

1. In Appian Designer, open the ABBYYFlexiCapture application and click **New** → **Folder**.
2. Select **Knowledge Center** as the type for your folder, complete the **Knowledge Center Name**, and click **Create**.

The ID of this folder will be passed in the repository field in the ABBYY FlexiCapture Connector for Appian export script settings. When data are exported from ABBYY FlexiCapture, image files will be saved to this folder.

```
<!--DMS repository name-->

<Repository>5995</Repository>
```

You can look up the ID of a knowledge center on the **Runtime Data** tab of the **Knowledge Center Properties** dialog box.

Configuring custom Appian applications to enable interaction with the ABBYY FlexiCapture Connector for Appian

Creating a new record type

To enable a custom Appian application to interact with the ABBYY FlexiCapture Connector for Appian, you will need to create the entities described below. Prior to creating these entities, make sure that the **ABBYYFlexiCapturePlugin.jar** plug-in is installed on the Appian server. To install the plug-in on the Appian server, simply copy it from the connector distribution package into this folder: **APPIAN_INSTALL/_admin/plugins/**. Now you need to:

- Create a data type.
- Create a data source.
- Create a data store.
- Create a record type.
- Create a constant of type "Data Store Entity" to link the data store to the data type.
- Create an AFC_GetConstantDSEByRecordTypeName rule.
- Create an AFC_GetDataTypeByRecordTypeName rule.
- Create a folder of type "Knowledge Center" where to store processed files received from ABBYY FlexiCapture.
- Create a Web API method for adding a new document into the knowledge center.
- Create a Record Web API method for populating the database table.

Creating an AFC_GetConstantDSEByRecordTypeName rule that will return a constant of type "Data Store Entity"

The CreateRecord method of the Web API requires a rule that returns constant of type "Data Store Entity." This rule will determine the data store and data type where a record should be created.

1. In Appian Designer, open the application and click **New** → **Expression Rule**.
2. Select **Create from scratch**, complete the **Name** and **Save In** fields, and click **Create & Edit**.
3. In the **Rule Inputs** dialog box, click the **+ icon** and create a new input variable of type "Text" and name it "RecordTypeName".
4. Paste the following code into the code editor on the left:

```
if(ri!RecordTypeName == "New Records", /* In the RecordTypeName parameter, specify the name of the new record type in the plural.  */

  cons!New_Constant, /* After cons!, specify the name of the constant of the record type. */

  null /* If there is more than one record type in the application, replace "null" with the same "if" condition as above, specifying the name of the second record type and the name of its constant. */

)
```

If an input RecordTypeName is the same as the name of a stored record type (in the plural), the DataStoreEntity constant will be returned that links the data store and the data type.

Creating an AFC_GetDataTypeByRecordTypeName rule that will return the full name of the data type

The CreateRecord method of the Web API requires a rule that returns the full name of the data type.

1. In Appian Designer, open the application and click **New** → **Expression Rule**.
2. Select **Create from scratch**, complete the **Name** and **Save In** fields, and click **Create & Edit**.
3. In the **Rule Inputs** dialog box, click the **+ icon** and create a new input variable of type "Text" and name it "RecordTypeName".
4. Paste the following code into the code editor on the left:

```
if(ri!RecordTypeName == "New Records", /* In the RecordTypeName parameter, specify the name of the new record type in the plural, the full name of the data type, and the namespace of the record type. */
  'type!{urn:com:appian:types}NewDataType',
  null
/* If there is more than one record type in the application, replace "null" with the same "if" condition as above, specifying the name of the second record type. the full name of the data type, and the namespace of the record type. */
)
```

If an input RecordTypeName is the same as the name of a stored record type (in the plural), a string will be returned containing the full name of the data type; otherwise, *null* will be returned (you can look up the namespace in the data type properties). To display the properties of a data type, open the desired data type in the ABBYYFlexiCapture application. Clicking the name of the data type in the top left corner will open the **Data Type Properties** window where you can see the namespace.

Creating a Web API method for adding a new document to the knowledge center

1. In Appian Designer, open the application and click **New** → **Web API**.
2. Enter "UploadDocument" (without the quotes) into the **Name** and **Endpoint** fields. Change **HTTP Method** to **POST** and click **Create & Edit**.
3. Close the template selection dialog box and paste the following code into the code editor:

```
with(
  local!value: http!request.body,
  a!httpResponse(
    statusCode: 200,
    headers: {},
    body: uploaddocument(local!value)
  )
)
```

Note: The uploaddocument(String query) method will be displayed in the code editor only if there is a **ABBYYFlexiCapturePlugin.jar file** in **APPIAN_INSTALL/_admin/plugins/**. This method sends a JSON string with the file contents directly to the JAVA method you created and gets back a JSON string with the ID of the folder where the document was placed.

Creating a Web API method for populating the database table

1. In Appian Designer, open the application and click **New** → **Web API**.
2. Enter "CreateRecord" (without the quotes) into the **Name** and **Endpoint** fields. Change **HTTP Method** to **POST** and click **Create & Edit**.

3. Close the template selection dialog box and paste the following code into the code editor:

```
with(
  local!value: cast(
    rule!AFC_GetDataTypeByRecordTypeName(http!request.queryParameters.RecordTypeName),
    a!fromJson(http!request.body)
  ),
  a!writeToDataStoreEntity(
    dataStoreEntity:
rule!AFC_GetConstantDSEByRecordTypeName(http!request.queryParameters.RecordTypeName),
    valueToStore: local!value,
    onSuccess: a!httpResponse(
      statusCode: 200,
      headers: {
        a!httpHeader(name: "Content-Type", value: "application/json")
      },
      body: a!toJson(
        fv!storedValues
      )
    ),
    onError: a!httpResponse(
      statusCode: 500,
      headers: {
        a!httpHeader(name: "Content-Type", value: "application/json")
      },
      body: a!toJson(
        {
          error: "There was an error writing to the data store"
        }
      )
    )
  )
)
```

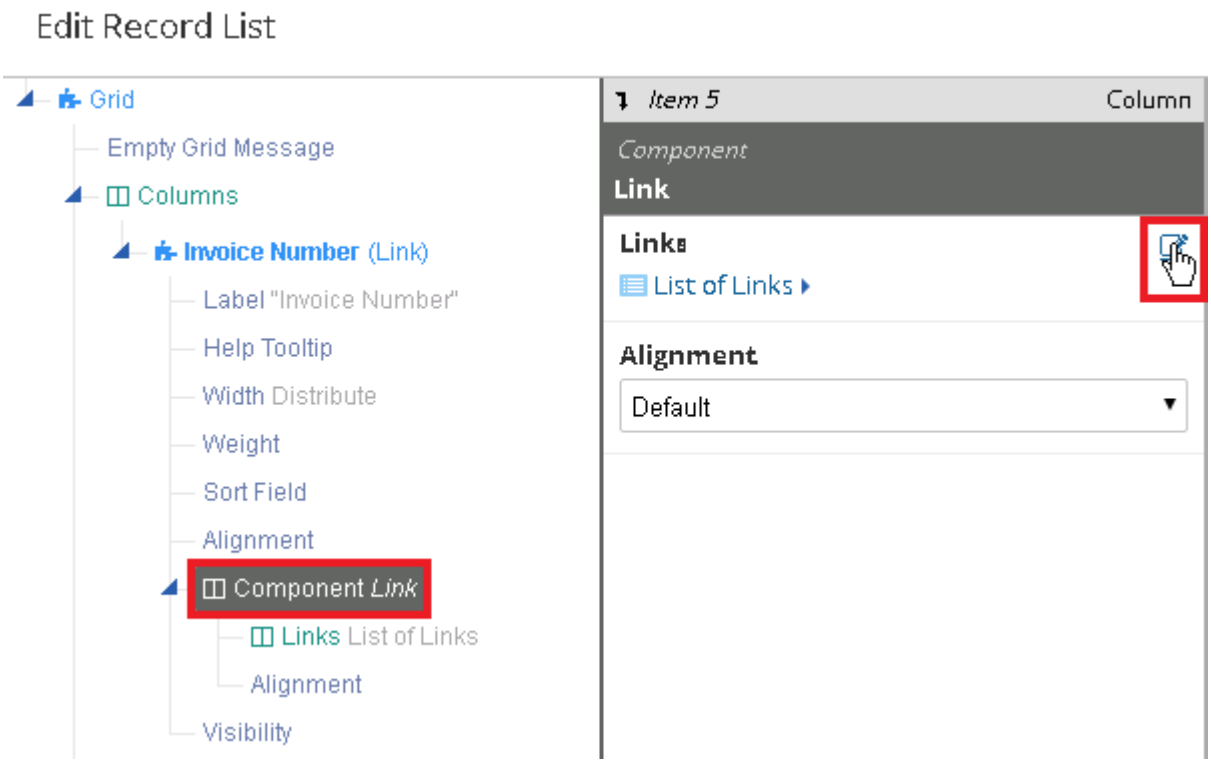
- The `rule!AFC_GetDataTypeByRecordTypeName(http!request.queryParameters.RecordTypeName)` rule will accept as input a parameter with the name of the record type (received by the Web API method from the ABBYY FlexiCapture Connector for Appian) and return the full name of the data type.
- The `rule!AFC_GetConstantDSEByRecordTypeName(http!request.queryParameters.RecordTypeName)` will return the Data Store Entity constant.

4. In the right-hand pane, click the **New Query Parameter** button and add a new parameter named **RecordTypeName** into the query string. Finally, select the **Set as default test value** option and click **Save**.

Configuring the display of a record type

Configuring the record list

1. In Appian Designer, open the application and click **Record Type**.
2. In the **Record List** section, click the **Edit Record List** link to edit the records. Some of the columns were already added automatically when you created the new record type. To add a new column, click **Add Column** and specify:
   - **Label** – the name of the new column;
   - **Select Component** – the type of data to be displayed in the column (you can configure the data source by clicking the component's name).
3. In the **Edit Record List** window, expand the **Columns** node, then expand the column with a link that you want to edit, and select **Component Link**. In the pane on the right, click the **Edit as Expression** icon next to **Links** (see screenshot below). This will open s code editor.



4. Paste the following code into the code editor:

```
a!recordLink(
  label: rf!InvoiceNumber, /* After label: rf!, type the name of the field whose value
should be a clickable link. */
  recordType: rp!type,
  identifier: rp!id
)
```

Clicking the link will open the record in the summary view.

For detailed instructions on working with the Record List, please refer to the Appian online documentation.

Configuring the record view

Users can view detailed information about each record. By default, information about each record is displayed in the summary view, but you can create your own custom views using **New View** (see the Appian online documentation for detailed instructions).

To make the data from a table row appear in the summary view:

1. Start editing the record type.
2. In the **Record View Details** section click **Summary** in the **Views** table. An **Edit View** window will open.
3. Paste the following code into the **Interface** field:

```
     =a!dashboardLayout(
  firstColumnContents: {
    a!textField(
      label: "Invoice Number", /* label - This is the name of the field in the view. */
      labelPosition: "ADJACENT", /* labelPosition - This is an alignment. */
      value: rf!InvoiceNumber, /* value - This is a link to the field in the corresponding
data type. */
      readOnly: true /* readOnly - This attribute indicates whether the field can be edited or
not.  */
    ),
    a!textField(
      label: "Invoice Date",
      labelPosition: "ADJACENT",
      value: rf!InvoiceDate,
      readOnly: true
```

```
    ),
    a!textField(
      label: "Total",
      labelPosition: "ADJACENT",
      value: rf!Total,
      readOnly: true
    )
  }
)
```

For detailed instructions on modifying the dashboard layout, please refer to the Appian online documentation.

4. Click **OK** to save the changes that you have made to the record type.