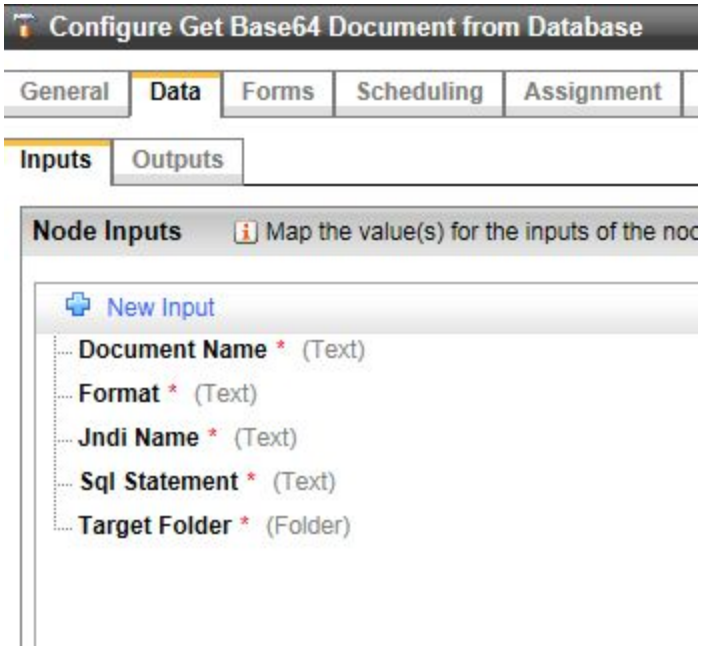
Base64DB Instructions

Base64QueryToDocument

This smart service will convert SQL statement results from Base64 to an Appian document, returning a document Id.

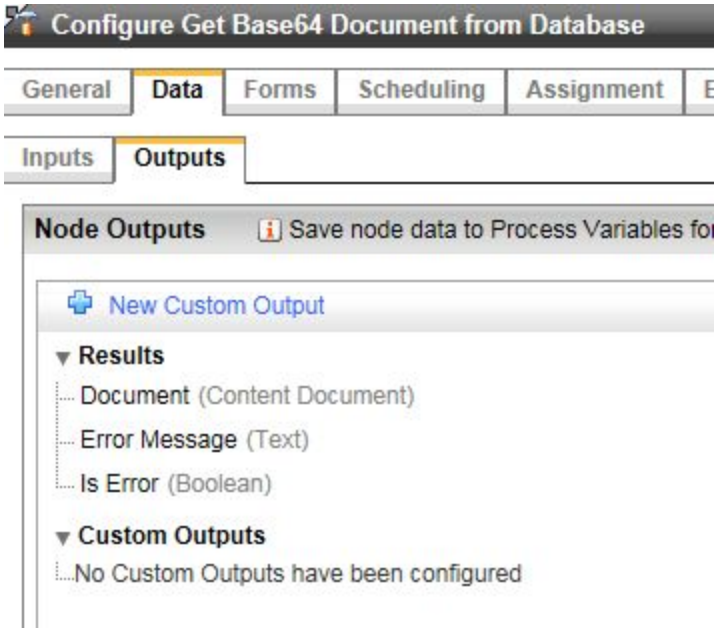
Inputs



- **Document Name** - what filename the document should be saved as
- **Format** - the file type for the document to be saved - ex, “txt”

- **Jndi Name** - the jndi name of the database where the query should be executed against. - ex: “jdbc/AppianBusinessDB”
- **Sql Statement** - the string SELECT statement to execute against the database. ***It should return a single value of a base64 string***. Ex: “SELECT base64_string from table_name where id = 24;”
- **Target Folder** - The Appian folder where the document should be saved to

Outputs



- **Document -** the document with converted base64 string (plaintext)
- **Error Message** - error message. Null if there Is Error is false.
- **Is Error** - boolean flag that indicates whether there was an error

Usage Example

Execute the following SQL script:

```
----
CREATE TABLE employees_test (
      id int,
      base64str varchar(1000)
);

INSERT INTO employees_test VALUES (1, 'ZW5jb2RlZCB0ZXh0');
----
```

Create a document folder where you want the converted document to be saved.

Customize the smart service node as follows:

- **Document Name** - plaintext
- **Format** - txt
- **Jndi Name** - the jndi name of the database
- **Sql Statement** - SELECT base64str from employee_test WHERE id = 1;
- **Target Folder** - Folder created above

Run the smart service node. Navigate to the target folder and a file named “plaintext.txt” should exist. Open the file. It should contain the text “**encoded text**”, which is the decoded base64 string inserted into the database.

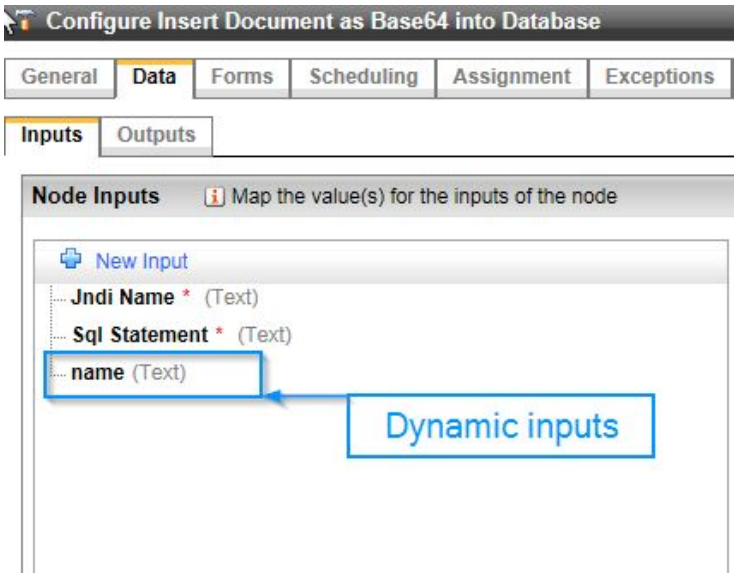
Error Handling

- Nonexistent column name - error message “unknown <column_name> in field list”
- SQL Statement returns multiple rows - error message “Your SQL statement returns more than one row. Please revise the statement to return only one row.”
- Malformed SQL statement (does not begin with “select <column_name> from”) - error message “Your SQL statement is invalid. Please use the format: SELECT <BASE64_COL_NAME> FROM <TABLE_NAME>”

DocumentToBase64Database

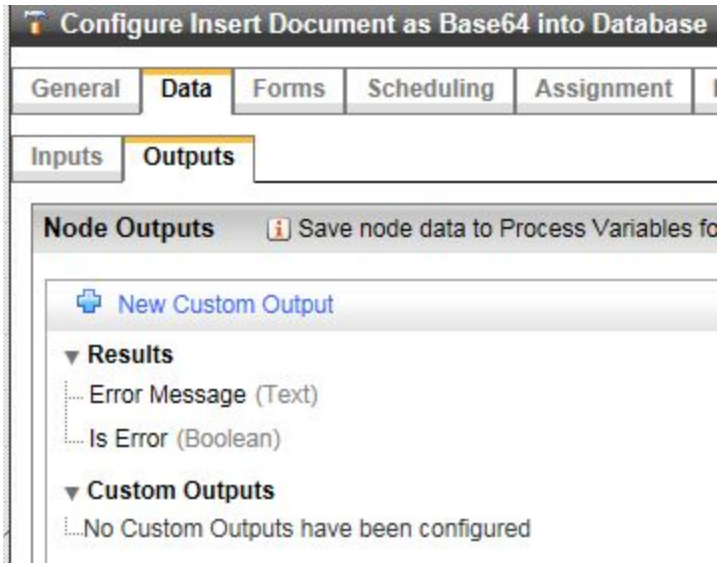
This smart service can receive an Appian document id, convert to a base64 string and save it directly to the database.

Inputs



- **Jndi Name** - the jndi name of the database where the query should be executed against. - ex: “jdbc/AppianBusinessDB”
- **Sql Statement** - the string SQL Insert statement which is formatted to include placeholders for variables - ie “INSERT INTO table VALUES(:name)”
- **[ANY DYNAMIC INPUTS]** - Additional inputs, named as placeholders in the SQL statement

Outputs



- **Error Message** - error message. Null if there Is Error is false.
- **Is Error** - boolean flag that indicates whether there was an error

Usage Example

Execute the following SQL script:

```
----
CREATE TABLE book (
      base64str varchar(1000),
      title varchar(100)
);
----
```

Create a text file with the words: "**encoded text**". Upload this file to Appian and note the document ID.

Configure the smart service node with the appropriate JNDI name for the **Jndi Name** input.

For the **Sql Statement** input, use:

- `INSERT INTO book VALUES (:documentId, :title);`

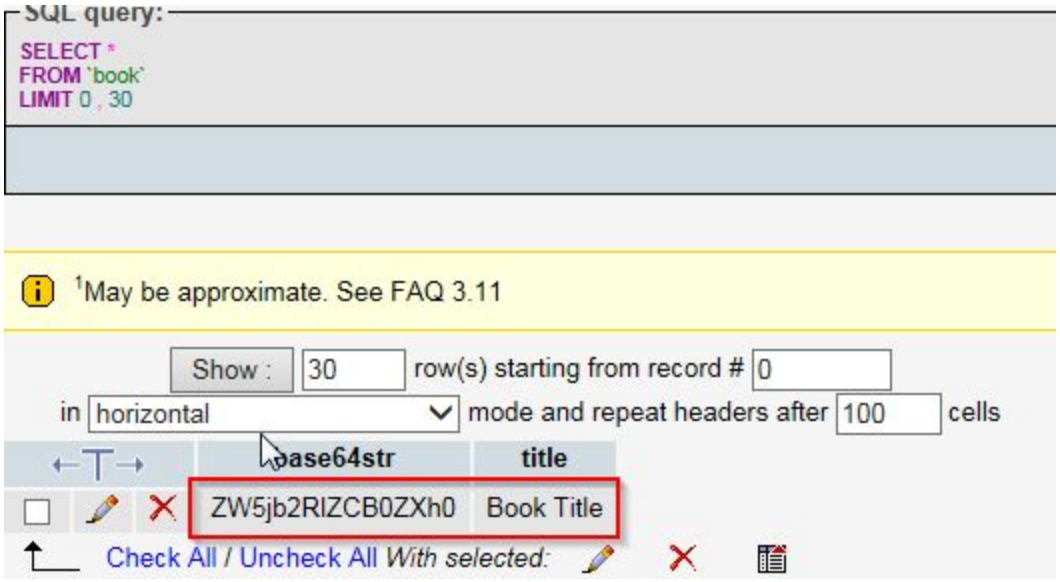
Add an ACP named **documentId** of type Number (Integer). The value should be the document ID of the document uploaded above. This ACP must be named documentId for the node to complete the conversion.

Add an ACP named **title** of type Text. The value should be:

- "Book Title"

NOTE: the order of the dynamic inputs is not important.

Run the smart service. Navigate to the book table. It should appear as follows:



The column **base64str** is the base64 encoding of the file's text (**encoded text**). Note that the title column is unaffected.