## Contents

## Introduction

There are various ways in which you can integrate TextPlan with your existing systems, depending on your needs. These are the main scenarios:

- To load data from your system into a TextPlan form you have the following options:
  - Open a form in TextPlan and load one or data models that your system exported (or you created manually) in either XML or Excel format, either from a file on your file system or from the TextPlan database. In the latter case an administrator must first *import* the data model(s) into TextPlan.
  - Your system can open a TextPlan form in the browser using an HTTP-*post* request, specifying some or all of the data to prepopulate the form with in the post data. The user can then change and add data and then generate a document as normal.

- To export TextPlan form data to your system you have the following options:
  - You can add a "Send" button to your form that sends the entered data, possibly along with a generated PDF document, to your own web service.
  - An administrator can *export* the currently entered data either in XML or Excel format. The main use of this functionality is to provide you with an example of the XML or Excel format that you can use to *import* data into TextPlan.

- If you want to use TextPlan solely to generate a document based on data that you provide, i.e. use it as a web service, you can request it to generate and return a document with the data that you supply.

## Importing XML and Excel data

An XML or Excel file that can be imported into TextPlan contains one or more *data model sets*, each of which contains one or more *data model*, each of which contains a set of input values that can be loaded into a form's fields.
The simplest way to understand the format is to *export* a form's currently entered data as either an XML or Excel file. An administrator can do this using the *"Export"* menu item on the *"Admin functions"* menu of a form's data entry page. The exported Excel file will contain remarks for all relevant cells explaining the various formats and possibilities.

If you have an XML or Excel file in the correct format, a user can load its content using the *"Load"* menu item on the *"Options"* menu (option *"Load from file"*).

- If the file contains only one data model, that model is loaded immediately.
- If the file contains one data model set with multiple data models, a panel appears from which the user can select a data model to load. Note that a data model can contain data for some or all fields. So one model could e.g. be used for customer data and another for product data.
- If the file contains multiple data model sets, a panel appears from which the user can select the data model set to use, after which he can select the data model to load.

An administrator can also *import* an XML or Excel file's content in the TextPlan database, using the *"Import"* menu item on the *"Admin functions"* menu. This will store the data model set(s) in the TextPlan database, from which he and other users (if it is shared) can load it (also using the *"Load"* menu item on the *"Options"* menu).

## Direct integration with your system

If you want to start or use TextPlan directly from your system, you have two options:

- You can open a TextPlan form in the browser using an HTTP-***post*** request, specifying the data to prepopulate the form with in the post data. The user can then change and add data and then generate a document as normal.
- You can use TextPlan as a web service, by requesting to generate and return a document with the data that you supply.

In both scenario's you need to tell TextPlan to which text blocks and variables the specified data applies. This can be done in two ways:

- A text block or variable has an "original name". This is the text that was highlighted in the template document. Note that although initially form fields have the same display name as the "original name", these names can be changed on the form configuration page for display purposes.
- Each text block and variable is also assigned a unique ID by TextPlan.

When identifying a text block or variable, you can use either the original name or the TextPlan ID. Note that 2 text blocks or variables can have the same original name, but they will always have different IDs. So if you use duplicate original names in your template document, it is better to use the ID, since you can't specify a value for the second text block or variable using the duplicate original name. If both the ID and the original name are specified, the ID is used.

For variables in *repeatable* text blocks, the ID must be followed by an *instance index*. *"variableId:0"* specifies the first instance of the variable, *"variableId:1"* specifies the second instance, and so on. Variables in repeatable text blocks cannot be specified by using the original name.
A text block is "repeatable" if it can occur more than once in the generated document. This is configured on the form's configuration page.

The simplest way to see all IDs and original names used in a form is to export the form to either an XML or Excel file. You can do this on the data entry page, by selecting *"Export"* from the *"Admin functions"* menu.

## Prepopulate a TextPlan form with data

A TextPlan form can be opened in two ways by using an HTTP *GET* request:

- By specifying its *ID*. For this to work, the user must be logged in to TextPlan and have access to the form.
  The URL in the browser is "https://my.textplan.com/Create/Form/***<id>***".
- By specifying its *public GUID*. By using the public GUID, the user does not have to log in to TextPlan to use the form. A form's GUID is shown on the form's management page.
  The URL in the browser is "https://my.textplan.com/Create/Form?guid=***<guid>***".

If you want to pass form data to the request, you should use an HTTP ***POST*** request, where either the ID or the GUID is specified as post data, along with the other fields.
Using the URL "https://my.textplan.com/Create/Form" you can pass the following post data:

- **userData** – required. A string containing the form data in JSON format. The format is explained below.
- **test** - optional. If true, the page is opened with test mode set to on. Default is false.

- **locale** - optional. If specified, this locale will be used for the form's user interface (not for the formatting of data), if supported. A locale specifies a language and a country, e.g. "en_US" for US English.

Aside from these parameters, you must either specify the form's ID or GUID.
If you specify the GUID, pass these parameters:

- **guid** - required. The form's GUID.

If you specify the ID, pass these parameters:

- **id** - required. The form's ID.
- **login** - required (except when the user is already logged in in the browser). The login/email address of a user with access to the form.
- **password** - required (except when the user is already logged in in the browser). The password of the user.

Note: if you specify an ID and login, the user is logged in with the specified credentials and can do all the same things as if he logged in manually. If you use the GUID method, the user only has access to this form, just like any anonymous user with access to the public link.

**The data model**
The data model to pass as value of the **userData** parameter is a string containing some or all of the form data in JSON format.

The data model specifies for zero or more text blocks whether they are used or not, and for zero or more variables their (textual) value. This data is processed as if the user entered it himself in the form's user interface, in the order as specified, so it should be in a valid *input* format, not the output format to use in the generated document.

You decide yourself how to build the JSON string. In order to define the format we use the following C# classes:

```
public class UserData
{
      public TextBlockData[]         TextBlocks;
      public VariableData[]          Variables;
}

public class TextBlockData
{
      public string                  Id;   // Specify either 'Id' or 'Name'.
      public string                  Name; // Specify either 'Id' or 'Name'.
      public bool                    Used;
      public int                     NrInstances;
}

public class VariableData
{
      public string                  Id;   // Specify either 'Id' or 'Name'.
      public string                  Name; // Specify either 'Id' or 'Name'.
      public int                     Type; // Optional.
      public string                  Value;// Text input for the form field.
}
```

You should either specify the 'Id' or the 'Name'. If both are specified, 'Id' is used.

For a text block the ‘Used’ property is true if this text block is used ("on") and false if it's not. The default value is true.

A text block’s ‘NrInstances’ property must be set to a value >= 0 for *repeatable text blocks*, i.e. text blocks that can occur more than once in the generated document. Whether a text block is repeatable or not is configured on the form configuration page (by default a text block is not). For not-repeatable text blocks this value must be set to -1, which is the default.
If a value > 0 is specified, i.e. if the text block is repeatable, all IDs of this text block’s variables *must* be specified as *variableId:InstanceIndex*, where *instanceIndex* is a value >= 0.

Optionally, you can specify a variable’s 'Type', as an integrity check. If specified, the 'Value' is only taken over if the variable is configured to have the same type. Supported types are:

- 0: Any (don’t check, this is the default)
- 1: Text
- 2: Number
- 3: Date
- 4: Choice
- 11: Boolean (Yes or no)

The **userData** parameter is a serialized JSON string of an object of type UserData.
An example of a JSON data model is the following:

```
{
	'TextBlocks' : [
		{ 'Id' : '1', 'Used' : 'true' },
		{ 'Id' : '2', 'Used' : 'false',  },
		{ 'Name' : 'TextBlock3', 'Used' : 'true' }
	],
	'Variables' : [
		{ 'Id' : '1', 'Value' : 'value1' },
		{ 'Name' : 'Variable2', 'Value' : 'Value2' }
	]
}
```

The JSON string of this data model is:

```
"{'TextBlocks':[{'Id':'1','Used':'true'},{'Id':'2','Used':'false'},{'Name':'TextBlo
ck3','Used':'true'}],'Variables':[{'Id':'1','Value':'value1'},{'Name':'Variable2','
Value':'Value2}]}"
```

## Request TextPlan to generate and return a document

You can use TextPlan as a web service, by requesting to generate and return a document with the data you supply.
To do this, send an HTTP ***POST*** request to the URL “https://my.textplan.com/Create/ GenerateOutputDocument” and specify the following post data:

- **dataModel** - required. A string containing the data model in JSON format. The format is explained below.

- **test** - optional. If true, a test document is created (with a TextPlan watermark). Default is false.
- **format** - the type of document to generate. Supported types are:
  - 10: Doc (Word 2003)
  - 20: Docx
  - 30: RTF
  - 40: PDF
  - 60: ODT
  - 70: Text
  - 104: JPeg

Aside from these parameters, you must either specify the form's ID or GUID.
If you specify the GUID, pass these parameters:

- **guid** - required. The form's GUID.

If you specify the ID, pass these parameters:

- **id** - required. The form's ID.
- **login** - required (except when the user is already logged in in the browser). The login/email address of a user with access to the form.
- **password** - required (except when the user is already logged in in the browser). The password of the user.

Note: if you specify an ID and login, the user is logged in with the specified credentials in order to generate the document, and logged out before the document is returned.

**The data model**
The data model to pass as value of the **dataModel** parameter is a JSON string specifying which text blocks are included and the value of all variables. Variables that are not specified will have an empty string as value.
Note that the variable values are used 'as is', they are not formatted in any way, so you should format the values as you want them to appear in the generated document.

You decide yourself how to build the JSON string. In order to define the format we use the following C# classes:

```
public class DataModel
{
      public TextBlock[]            TextBlockDatas;
      public Variable[]             Variables;
}

public class TextBlock
{
      public string                 Id;   // If starts with '@' it's a name.
      public bool                   Used;
      public int                    NrInstances;
}

public class Variable
{
      public string                 Id;   // Specify either 'Id' or 'Name'.
      public string                 Name; // Specify either 'Id' or 'Name'.
```

```
        public string                   Value;// The text to appear in the document.
}
```

The text block data is the same as the `TextBlockData` class for prepopulating a form, except that the text block's name can be specified in the 'Id' property, by starting it with a '@'. If a text block's 'Id' does not start with a '@' it is a normal text block ID. Example: "@start" specifies the text block with the original name "start", while "123" specifies the text block with the ID "123".

For variables you should either specify the 'Id' or the 'Name'. If both are specified, 'Id' is used.

The **dataModel** parameter is a serialized JSON string of an object of type DataModel.
An example of a JSON datamodel is the following:

```
{
        'TextBlockDatas' : [
                { 'Id' : '1', 'Used' : 'true' },
                { 'Id' : '2', 'Used' : 'false',  },
                { 'Id' : '@TextBlock3', 'Used' : 'true' }
        ],
        'Variables' : [
                { 'Id' : '1', 'Value' : 'value1' },
                { 'Name' : 'Variable2', 'Value' : 'Value2' }
        ]
}
```

The JSON string of this datamodel is:

```
"{'TextBlockDatas':[{'Id':'1','Used':'true'},{'Id':'2','Used':'false',},{'Id':'@Tex
tBlock3','Used':'true'}],'Variables':[{'Id':'1','Value':'value1'},{'Name':'Variable
2','Value':'Value2'}]}"
```

## Data collection: send entered form data to your own web service

You can configure your TextPlan form to show a “Send” button, which sends the entered data to an email address and/or to your own web service, instead of or in addition to the default “Create document” button.

This is configured in the form’s “Form properties” dialog, on the “Data collection” tab. Here you can specify the following:

- Whether to show the “Send” button for logged-in users and/or for anonymous users.
- The text of the “Send” button.
- An optional description to show before sending. If this is not left blank, the user is first shown a dialog with this message before actually sending the data.
- Whether the user can download a PDF generated using the sent data.
- Whether a generated PDF is sent alongside the entered data.
- And email address and/or a web service URL where to send the data to.

If an email address is specified, the entered data is emailed in an attachment  to that address, in encrypted TextPlan format. The receiver can save the attached file to his file system, open the corresponding TextPlan form, and load the data into the form.

If a web service URL is specified, the data is posted in JSON format to the specified URL.
In order to define the format of the data that is sent, we use the following C# classes:

```
public class WebHookData
{
      public string                   Key;
      public string                   Code;
      public bool                     Test;
      public WebHookTextBlock[]       TextBlocks;
      public WebHookVariable[]        Variables;
      public string                   Pdf;
}

public class WebHookTextBlock
{
      public string                   Id;
      public bool                     Used;
      public int                      NrInstances;
}

public class WebHookVariable
{
      public string                   Id;
      public int                      Type;
      public string                   Value;      // Text input of the form field.
}
```

The data that is posted to the web service is a serialized JSON string of an object of type WebHookData.
The properties of the classes WebHookTextBlock and WebHookVariable are as described in the chapter *“Prepopulate a*

*TextPlan form with data"* for the classes TextBlockData and VariableData.
The other properties of the WebHookData class have the following meaning:

- **Key**: reserved, currently always "TextPlan".
- **Code**:
  If the form's URL contains  a "code" parameter, the value of this parameter is passed to the web service.
  This can be used to report to the web service which action or customer the data concerns.
  For example, if your system provides your customer with a TextPlan form using a public URL, it can append "&code=<customerId>" to the URL that is provided to the customer. When the customer presses the "Send" button, "<customerId>" is sent to your web service in the "Code" field.
- **Test**:
  This is set to true if the "Send" button was pressed while the form was in *test mode*. This is an indication that your web service should not process the data as production data. A TextPlan account administrator can use test mode to test the form without being charged for outputs.
- **Pdf**:
  If it is configured in the "Data collection" tab that a generated PDF based on the entered data must also be send, the PDF file's data is included in this field, as a Base64-encoded string.

## Test code

The following HTML file shows how to prepopulate a form with data and how to use TextPlan as a web service.

```
<!DOCTYPE html>
<!--
        This HTML page shows how to prepopulate a form with data and how to use TextPlan as a web service.
        In this example we use the "Example Appointment" form with the public GUID "fcd286f6-a047-4560-bcaf-
259e3ec99289".
-->
<html>
        <head>
                <meta charset="utf-8" />
                <meta name="viewport" content="width=device-width" />
                <script type="text/javascript" src="http://ajax.aspnetcdn.com/ajax/jquery/jquery-
1.8.2.min.js"></script>
        </head>

        <body>

                <button type="button" style="width: 300px" onclick="handleOpenTextPlanForm()">Open form with
data</button><br/><br/>
                <button type="button" style="width: 300px" onclick="handleGenerateDocument()">Generate
document</button>

                <form id="formOpenTextPlanForm" method="post" novalidate="novalidate">
                        <input id="formOpenTextPlanFormGuid" type="hidden" name="guid"/>
                        <input id="formOpenTextPlanFormUserData" type="hidden" name="userData"/>
                </form>

                <form id="formGenerateDocument" method="post">
                        <input id="formGenerateDocumentGuid" type="hidden" name="guid"/>
                        <input id="formGenerateDocumentDataModel" type="hidden" name="dataModel"/>
                        <input id="formGenerateDocumentFormat" type="hidden" name="format"/>
                        <input id="formGenerateDocumentTest" type="hidden" name="test"/>
                </form>

                <script type="text/javascript">
                        // The public GUID of the form.
                        var theGuid = "fcd286f6-a047-4560-bcaf-259e3ec99289";
                        // Data to pass to "Create/Form".
                        // Note: the text block data for linked and not-repeatable text blocks that are always on
does not have to be specified.
                        var thePopulateFormData = "{'TextBlocks':[],'Variables':[{'Id':'1','Type':1,'Value':'John
Smith'},{'Id':'-
1','Type':11,'Value':'Yes'},{'Id':'2','Type':3,'Value':'12012014'},{'Id':'3','Type':1,'Value':'1200'},{'Id':'4',
'Type':1,'Value':'Tessa Yang'}]}";

                        // Data to pass to "Create/GenerateOutputDocument".
                        // Note: only repeatable text blocks and text blocks that are not used have to be
specified.
                        var theGenerateDocumentData = "{'TextBlockDatas':[{'Id':'@No
appointment','Used':false,'NrInstances':-1}],'Variables':[{'Id':'1','Type':1,'Value':'John Smith'},{'Id':'-
1','Type':11,'Value':'Yes'},{'Id':'2','Type':3,'Value':'December 1,
2014'},{'Id':'3','Type':1,'Value':'12:00'},{'Id':'4','Type':1,'Value':'Tessa Yang'}]}";

                        function handleOpenTextPlanForm() {
                                $("#formOpenTextPlanForm").attr("action", "https://my.textplan.com/Create/Form");
                                $("#formOpenTextPlanFormGuid").val(theGuid);
                                $("#formOpenTextPlanFormUserData").val(thePopulateFormData);
                                $("#formOpenTextPlanForm").submit();
                        }

                        function handleGenerateDocument() {
                                $("#formGenerateDocument").attr("action",
"https://my.textplan.com/Create/GenerateOutputDocument");
                                $("#formGenerateDocumentGuid").val(theGuid);
                                $("#formGenerateDocumentDataModel").val(theGenerateDocumentData);
                                $("#formGenerateDocumentFormat").val(40);    // PDF
                                $("#formGenerateDocumentTest").val(true);
                                $("#formGenerateDocument").submit();
                        }
                </script>

        </body>
```

```
</html>
```