# Form Designer

## User Guide

### Version: 7.8

Written by: Documentation Team, R&D
Date: Wednesday, March 25, 2026

HYLAND™

# Documentation Notice

The information and software described in this document are furnished only under a separate agreement and may only be used or copied according to the terms of such agreement. It is against the law to copy the software except as specifically allowed in such agreement. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright law, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Hyland Software, Inc. and/or one of its affiliates.

Hyland, OnBase, Alfresco, Nuxeo, Content Innovation Cloud, and other product or brand names are registered and/or unregistered trademarks of Hyland Software, Inc. and its affiliates in the United States and other countries. All other trademarks, service marks, trade names and products of other companies are the property of their respective owners.

© 2026 Hyland.

The information in this document may contain technology as defined by the Export Administration Regulations (EAR) and could be subject to the Export Control Laws of the U.S. Government including for the EAR and trade and economic sanctions maintained by the Office of Foreign Assets Control as well as the export controls laws of your entity's local jurisdiction. Transfer of such technology by any means to a foreign person, whether in the United States or abroad, could require export licensing or other approval from the U.S. Government and the export authority of your entity's jurisdiction. You are responsible for ensuring that you have any required approvals prior to export.

DISCLAIMER: This documentation contains available instructions for a specific Hyland product or module. This documentation is not specific to a particular customer or industry. All data, names, and formats used in this document's examples are fictitious unless noted otherwise. This document may reference websites operated by third parties. In such a case, Hyland has no control or liability for the content of such third-party websites. The inclusion of such a link shall not constitute an endorsement or affiliation with such a third-party website; the reference is provided for information purposes only. If you have questions about discrepancies in this document, please contact Hyland. Hyland customers are responsible for making their own independent assessment of the information in this documentation. This documentation: (a) is for informational purposes only, (b) is subject to change without notice, (c) is confidential information of Hyland Software, Inc. and its affiliates, and (d) does not create any commitments or assurances by Hyland. This documentation is provided "as is" without representation or warranty of any kind. Hyland expressly disclaims all implied, express, or statutory warranties. Hyland's responsibilities and liabilities to its customers are controlled by the applicable Hyland agreement. This documentation does not modify any agreement between Hyland and its customers.

# Table of Contents


**Documentation Notice** ... **3**

**What is Form Designer?** ... **7**

**What is Perceptive Forms Server?** ... **7**

**Get started with Form Designer** ... **7**

What is a form? ... 7

What are form types? ... 8

About extending form functionality ... 8

What are properties in forms? ... 9

Deploy a form overview ... 10

Toolbars in Form Designer ... 10

*File Toolbar* ... *10*

*Edit Toolbar* ... *11*

**Use Example Forms** ... **12**

Modify an example form overview ... 12

What are example forms? ... 12

Form control ... 13

*Form control example files* ... *13*

*Form control example demo.xsl file* ... *13*

*Form control example demo.xml file* ... *14*

Form control example demo.xml ... 14

*Form control example lookups.xml file* ... *15*

*Example shared formstylesheet.css file* ... *15*

Row management ... 17

*Row management example files* ... *17*

*Row management example ap_invoice.xsl file* ... *17*

*Row management example ap_invoice.xml file* ... *19*

**Create forms** ... **19**

Create a form ... 19

Open a saved form design ... 19

Copy a form ... 20

Preview a form ... 20

View another form while designing ... 20
Filter the form properties ... 20
Select a style ... 20
Copy text from an application ... 20
Select the font for the form ... 21
About data synchronization ... 21
Synchronize data ... 21
About verifying data in fields ... 21
Verify data in fields ... 22
Make fields required ... 22
Save a form design ... 22
Export form files ... 23
Set form privileges ... 23
Associate a form with a workflow queue ... 23
**Customize forms** ... **24**
Create a customized form section ... 24
Create a section with two columns ... 24
Reorder composite properties ... 24
Add a field ... 25
Add text to a form ... 25
Add a text area to a form ... 25
Add a link to a form ... 25
Add a picture to a form ... 26
Add an edit box to a form ... 26
Add a background to a property ... 26
Modify a border ... 26
**Publish forms** ... **27**
About publishing forms ... 27
Create a sequence ... 27
Publish a form as a document in Perceptive Content ... 28
Publish a form in a folder in Perceptive Content ... 28
Publish a form as a document to Forms Server ... 28
Publish a form as a folder to Forms Server ... 29

**Configure Form Designer** ..... **30**
What is imagenowforms.xml? ..... 30
imagenowforms.settings parameters ..... 31
*[PerceptiveContentForms] Perceptive Content Server settings* ..... *31*
*[PerceptiveContentForms] Application server settings* ..... *32*
*[Locale]* ..... *33*
Prevent the URL from appearing on the browser status bar ..... 33
What are data and configuration parameters? ..... 34
Data parameters ..... 34
Configuration parameters ..... 35
Transform parameters ..... 54

# What is Form Designer?

Form Designer gives you the ability to create, preview, and publish forms.

Form Designer allows you to customize elements of a form (such as images, links, and section titles) without requiring the use of scripting or technical development.

Form Designer enables you to select from custom properties created in Form Designer, including adding fields to a form. Creating fields from custom properties optionally allows the system to complete custom properties using the data a user enters in the form fields. This synchronization makes the form field data searchable.

Form Designer is divided into the following functional areas/

- The Properties pane shows all custom properties that you can utilize to create fields on your forms.
- The Toolbox pane contains all the tools that you can use to customize your form, such as adding links and images.
- The tab in the center of the screen is the work area where you construct the form.
- The Property Attributes pane enables you to define the movement of data between the ImageNow Server and the form and reorder composite properties.

You can use scripting or technical development to make more advanced changes to your forms.

# What is Perceptive Forms Server?

Perceptive Forms Server enables you to view a form in your browser using a URL. You can only access forms published as stand-alone documents or folders.

You deploy forms accessed with a URL in Forms Server. When you use Form Designer to publish the form, the imagenowforms user automatically receives access to view and submit a form accessible with a URL. You can create URLs that point to specific form types. Each time a user launches a URL, a new instance of the specified form type appears, allowing a user to view, complete, and submit the form into the Perceptive Content document or folder repository. For more information, refer to the Perceptive Forms Server Installation and Setup Guide.

# Get started with Form Designer

## What is a form?

Perceptive Content Forms are electronic forms that allow the user to enter data into customized fields.

Forms are an optional, core component of Perceptive Content. Whether forms are available for your use depends on your implementation of Perceptive Content. To determine if you have any forms available in your implementation, contact your Perceptive Software representative.

Your administrator can tell you which forms are available, including any workflow queues with which they are associated. You can complete a form by typing data into the fields on the form. Form data might also be pre-populated by other Perceptive Content components.

# What are form types?

Form types can affect the way the system indexes, stores, and displays forms.

Forms can displayed as a stand-alone document or on a folder or as supplemental information for a document or folder. Depending on your system configuration, each field can be mapped to a corresponding custom property. After you fill in the fields and save the form, the system can automatically complete the values for the corresponding custom properties on ImageNow Forms Server, allowing you to search for forms just as you would search for documents. When you open a document, this form appears independently of a document in ImageNowViewer.

Stand-alone forms allow you to capture additional data for your documents or folders. When you open a document, a stand-alone form appears independently of a document in ImageNowViewer. You can select a stand-alone form as the folder type when you create a folder. The form appears independently of a document in the Folder Viewer.

Supplemental forms allow you to capture additional data for your documents or folders. By default, in ImageNowViewer in the Forms pane or in ImageNowViewer in the Forms tab, the user can select any form and associate it with any document. You can use supplemental forms in the following ways.

- When viewing a document in ImageNowViewer, supplemental forms appear in the Forms pane. You can select from the available forms and fill out the fields to provide additional information about a document.
- When viewing a document in Folder Viewer, supplemental forms appear in the Forms tab. You can select from the available forms and fill out the fields to provide additional information about a folder.

# About extending form functionality

To use forms, you must install an Perceptive Content Forms license. If you need a Forms license, contact your Perceptive Software representative.

Using standard HTML, XSL, and XML, you can use external editing tools, such as Microsoft FrontPage or Adobe Dreamweaver to extend the functionality of forms that you create using Form Designer. Form Designer builds a Presentation Template and one Data Definition file. You can expand this by building multiple Presentation Templates for each Data Definition file. You can provide external Cascading Style Sheets (CSS) files or add in-line CSS to the XSL file. In addition, you can use JavaScript files for standard numeric functions, data validation functions, or other common functions, such as context-sensitive help.

You deploy a modified form by loading the form files onto the ImageNow Forms Server. If you are deploying a form to use with the ImageNow Forms Server, you associate the form with a folder type or document type. If you are deploying a form to use in the Forms pane, you can optionally associate the form with any appropriate workflow queues.

The following files are used in forms in Perceptive Content, although you can optionally use CSS, graphics, and JavaScript.

- `Presentation Template`. This XSL file describes how to present the XML data in the form. You can use many presentations with a single XML source file, so you can create different formatting or views of the data for each form you create. Each presentation can have different security settings. Through these settings, you can control access to your forms. Additionally, external ODBC access is provided through iScript, a scripting language available with Perceptive Content.
- `Data Definition`. This XML file contains the schema used to save data instances in data content record files. You can also add data to elements in this file to pre-populate fields in your form the first time it is displayed to a user.
- `Data Content Record`. This XML file contains individual instances of form data based on a corresponding data definition file. The data captured during form processing is stored for viewing, editing, searching, and archiving purposes. Perceptive Content creates and maintains the data content record files for you based on your data definition file. Perceptive Content stores these records as sub-objects.

If you choose not to use Form Designer to create your initial form files, you can use external editing tools to customize example forms. If you are using external editing tools to create form files, the system does not support files encoded with UTF-8 with BOM.

# What are properties in forms?

Form custom properties are property fields that can be populated with data relating to a form.

The custom property type affects the behavior of fields in the form and determines the format of the information a user must enter to complete the field. You can use the following types of custom properties to create different types of fields in a form.

- `Composite`. A composite property type can contain a set of custom properties. This property type allows a user to enter multiple values for each property in the set.
- `Date`. The accepted number format that a user can enter when completing date fields is based on the settings chosen when creating the custom property.
- `Flag`. When you add this custom property type to a form, the field appears as a drop-down list. The system can automatically display either a selected or a cleared check box based on the settings defined for the custom property.
- `List`. When you add this custom property type to a form, a list box is displayed on the published form. On the form, the user can select one value from the existing custom property.
- `Number`. Number fields allow both positive and negative numbers. The accepted number format that a user can enter when completing number fields is based on the settings chosen when creating the custom property.
- `String`. The text fields support all printable ASCII characters within the character set.
- `User`. When you add this custom property type to a form, a list appears on the published form.

# Deploy a form overview

The following steps outline the high-level procedures you need to perform to deploy forms accessed with a URL in Forms Server. When you use Form Designer to publish the form, the imagenowforms user automatically receives access to view and submit a form accessible with a URL. Note that if you choose to deploy forms access with a URL, you must ensure that your network is configured with the appropriate security. To deploy a form, complete the following sequence of procedures.

1. Create a form using **Form Designer**.
2. Publish the form using one of the following methods and create the document or folder type in **Form Designer**.
   - Publish a form as a document in Perceptive Content.
   - Publish a form as a document to Forms Server.
   - Publish a form as a folder to Forms Server.
   - Publish a form in a folder in Perceptive Content.
3. Define security for the form using any of the following options.
   - Set form privileges.
   - Set privileges for documents or folders associated with forms and any workflow queues.
   - Grant the Perceptive Content Forms group access to the workflow queue where the Forms Viewer applet is configured to route the forms
   - Grant the Perceptive Content Forms group privileges to the form that you associate with the folder type or document type.
   - Grant the Perceptive Content Forms group access to the folder type or document type.

   **Example** For a folder displayed as a form, the group needs both the **Drawer > Content > Open** and **Drawer > Content > Create/Append** privileges for the drawer that stores the folder.
4. Optional. Associate a form with a workflow queue.
5. Embed the forms URL in your portal or business application.

# Toolbars in Form Designer

The following tables describe the File and Edit Toolbars.

## File Toolbar

| Tool | Name | Description |
|---|---|---|
| | New | Open a blank form. |
| | Open | Select a form from the list. |

| | | |
|---|---|---|
| | Save | Save the selected form. |
| | Export Design | Export saved design files from the server. |
| | Form Preview | View the way the form appears after publication. |
| | Publish | Create a new document type to associate to the form. |
| | Print Preview | Open the Print Preview dialog box. |
| | Print | Print the selected form. |

## Edit Toolbar

| Tool | Name | Description |
|---|---|---|
| | Undo | Undo the last update. |
| | Redo | Replace the update after you perform an undo action. |
| | Cut | Remove text. |
| | Copy | Copy text to the clipboard. |
| | Paste | Paste text after you copy to the clipboard. |
| | Font | Open the Font dialog box. |
| | Bold | Make selected text bold. |
| | Italicize | Italicize selected text. |
| | Underline | Underline selected text. |

# Use Example Forms

## Modify an example form overview

To customize an existing form example provided in Perceptive Content, or to modify and extend the functionality for a form created in Form Designer, complete the following sequence of procedures.

**Prerequisite** Before completing this procedure, you must include an XML declaration in the XSL and XML file that includes the encoding. For Unicode environments, your encoding must be set to UTF-8.

You can create as many presentations to use with one XML source file as you need.

1. Modify a presentation XSL file.
2. Optional. To embed an external CSS file in your XSL file, in the `<head>` section, insert `<link href="file-name.css" rel="stylesheet" type="text/css" />` where *filename* is the name of your CSS file.

   **Example** `<link href="psi_style.css" rel="stylesheet" type="text/css" />`
3. Optional. To embed an external JavaScript file, in the `<head>` section of your form XSL file, insert `<script LANGUAGE=" JavaScript" SRC="scriptname.js"></script>` where *scriptname* is the name of your script.

   **Example** `<script language="JavaScript" SRC="external.js"></script>`
4. Optional. Build a graphics file.

   **Example** `<img src="healthcare.jpg" alt="Healthcare Image"></img>`

   **Note:** You can include PNG, GIF, or JPG graphic files. Use as many graphic files as you need.
5. Create a data definition XML schema file.
6. Optional. Test a form.

## What are example forms?

If you choose not to useForm Designer to create forms, Form Designer includes example forms that can help you create a form using an external editor.

Each of the examples contains the XML, XSL, and CSS files that comprise a form.

Copy the example code from these help topics into text files and use them to create your own forms in Perceptive Content.

# Form control

## Form control example files

The Form Control Example shows you how to write code to use the different form controls supported by Perceptive Content in your forms. It also provides information about how to write lookups to data in other files. This form example shows how to code each of the form controls offered by Perceptive Content in XSL using the common illustration of shipping information such as the Ship To, Ship Date, and other shipping information displayed using such form controls as a list box, text box, radio buttons, check box, and a text area control. The Ship To field is a drop-down list box that is dynamically populated using scripting in the XSL file that looks up the vendor in the lookups.xml file.

| Directory | Description |
|---|---|
| demo.xsl | The demo.xsl file contains the HTML form controls, and XSL code examples you can follow to use these controls in your own forms with Perceptive Content. |
| demo.xml | The demo.xml file contains example elements so that you can follow this structure when you are coding your own form files. |
| lookups.xml | The lookups.xml file contains code example of data lookups to give you an idea of how you can code your own data lookups with your own applications or systems. When you deploy your form, add this xml file as part of the form presentation files. |
| formstylesheet.css | The formstylesheet.css file is an example of a shared form file. This file gives you an idea of how you can use CSS to style your own forms. Shared form files in Perceptive Content enable you to use the same file with many different forms so that you can single-source the files that are used with more than one form. |

## Form control example demo.xsl file

This XSL file is part of the Form Control Example form. It contains the HTML form controls and XSL code examples you can follow to use these controls in your own forms files.

```
<?xml version="1.0" encoding="UTF-8"?><xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:template match="/page">
<html><head><link href="FormStyleSheet.css" rel="stylesheet" type="text/css"
/><title>Form</title></head> <body><form name="frmForm"> <table
cellSpacing="0" cellPadding="0" border="0"> <tr> <td class="fieldcell"
colspan="4"> <img src="Perceptive.jpg"></img><br/><br/> </td> </tr> <tr> <td
```

```
class="fieldcell" style="width:100px">Ship To:</td> <td class="fieldcell">
<xsl:variable name="ship_to"> <xsl:value-of select="SHIP_TO"/>
</xsl:variable> <select name="cboVendorName" id="cboVendorName"
class="fieldcell" style="width:75px" tabindex="1"> <xsl:attribute
name="value"> <xsl:value-of select="SHIP_TO"/> </xsl:attribute> <xsl:for-each
select="document('Lookups.xml')/lookups/vendors/row"> <option> <xsl:attribute
name="value"> <xsl:value-of select="value"/> </xsl:attribute> <xsl:value-of
select="value"/> </option> </xsl:for-each> </select> </td> </tr> <tr> <td
class="fieldcell" style="width:100px">Ship Date:</td> <td class="fieldcell">
<input type="text" name="txtSHIP_DATE" class="fieldcell" style="width:150px">
 <xsl:attribute name="value"> <xsl:value-of select="SHIP_DATE"/>
</xsl:attribute> </input> </td> </tr> <tr> <td class="fieldcell"
style="width:100px">Ship Via:</td> <td class="fieldcell" style="width:275px">
<xsl:variable name="ship_via"> <xsl:value-of select="SHIP_VIA"/>
</xsl:variable> <div> <xsl:attribute name="value"> <xsl:value-of
select="SHIP_VIA"/> </xsl:attribute> <xsl:choose> <xsl:when test="$ship_
via='FedEx'"> <input type="radio" name="rdoSHIP_VIA" checked="true"
class="ChecksAndRadios">FedEx</input> </xsl:when> <xsl:otherwise> <input
type="radio" name="rdoSHIP_VIA" class="ChecksAndRadios">FedEx</input>
</xsl:otherwise> </xsl:choose> <xsl:choose> <xsl:when test="$ship_via='UPS'">
 <input type="radio" name="rdoSHIP_VIA" checked="true"
class="ChecksAndRadios">UPS</input> </xsl:when> <xsl:otherwise> <input
type="radio" name="rdoSHIP_VIA" class="ChecksAndRadios">UPS</input>
</xsl:otherwise> </xsl:choose> <xsl:choose> <xsl:when test="$ship_via='DHL'">
 <input type="radio" name="rdoSHIP_VIA" checked="true"
class="ChecksAndRadios">DHL</input> </xsl:when> <xsl:otherwise> <input
type="radio" name="rdoSHIP_VIA" class="ChecksAndRadios">DHL</input>
</xsl:otherwise> </xsl:choose> </div> </td> </tr> <tr> <td class="fieldcell"
style="width:100px">Shipping Instructions:</td> <td class="fieldcell">
<textarea rows="8" cols="150" class="fieldcellsmaller" style="width:400px"
id="txaSHIP_IN" name="txaSHIP_IN"> <xsl:value-of select="SHIP_IN"/>
</textarea> </td> </tr> <tr> <td class="fieldcell"
style="width:100px">Payment Received?</td> <td class="fieldcell"> <input
type="checkbox" name="chkPAYMENT_RECEIVED" class="ChecksAndRadios">
<xsl:attribute name="value"> <xsl:value-of select="PAYMENT_RECEIVED"/>
</xsl:attribute> </input> </td> </tr></table> </form></body> </html>
</xsl:template></xsl:stylesheet>
```

## Form control example demo.xml file

This XML file is one of two included in the Form Control Example form. It contains example elements so that you can follow this structure when you are coding your own form files.

### Form control example demo.xml

```
<?xml version="1.0" encoding="UTF-8"?><page> <SHIP_TO></SHIP_TO> <SHIP_
DATE></SHIP_DATE>` <SHIP_VIA></SHIP_VIA> <SHIP_IN></SHIP_IN> <PAYMENT_
RECEIVED></PAYMENT_RECEIVED></page>
```

## Form control example lookups.xml file

This is a second XML file in the Form Control Example form. It contains code examples of data lookups to give you an idea of how to code your own data lookups with your own applications or systems. When you deploy your form, add this XML file as part of the form presentation files.

```
<?xml version="1.0" encoding="UTF-8"?><lookups> <vendors> <row>
<value></value> </row> <row> <value>ACME</value> </row> <row> <value>Jet
Electric</value> </row> <row> <value>Blue Beach</value> </row> <row>
<value>Master Crafters</value> </row> </vendors></lookups>
```

## Example shared formstylesheet.css file

This CSS file is part of the Form Control Example form. It is an example of a shared form file. The file gives you an idea of how to use CSS to style your own forms. Shared form files in Perceptive Content enable you to use the same file with many different forms so that you can single-source the files that are used with more than one form.

```
* { margin: 0px; padding: 1px} body { font-family: tahoma, sans-serif; font-
size: 12px; line-height: 15px; text-align: left; } h1 { background-image: url
(banner1.png); background-repeat: no-repeat; text-indent: -9999px; display:
block; margin: 0px; padding: 0px; height: 99px; width: 200px; float: right; }
h2 { font-family: tahoma, sans-serif; font-size: 14px; font-weight: bold;
text-transform: uppercase; color: #FFFFFF; background-color: #91B0D5; letter-
spacing: 5px; display: block; padding-top: 5px; padding-right: 5px; padding-
bottom: 5px; padding-left: 10px; float: none; line-height: normal; height:
auto; margin: 0; border-top: solid #333 1px; border-bottom: solid #333 1px; }
h3 { font-size: 16px; color: #91B0D5; text-transform: uppercase; height:
auto; width: 300; display: block; margin: 0; padding: 10px 0px 5px; } h4 {
font-size: 14px; color: #91B0D5; height: auto; width: 300; display: block;
margin: 0; padding: 10px 0px 5px; } h5 { font-size: 12px; color: #91B0D5;
height: auto; width: 300; display: block; margin: 0; padding: 10px 0px 5px; }
#container .inner-left table { font-size: 12px; padding: 2px; } #container
.inner-left #form2 select { font-size: 8px; background-color: #EEEEEE; text-
align: center; TABLE { color: #000000; width: 600px; border: 3px solid white;
 border-collapse: collapse; } TH { font-size: 10px; border: 1px solid black;
padding-left: 4px; text-align: center; } TD { padding-left: 4px; } HR {
width: 625px; } .subtotalcaption { font-size: 10px; font-weight: bold; text-
align: right; padding-left: 1px; padding-right: 1px; } .subtotal { font-size:
10px; border-left: 2px solid black; border-right: 2px solid black; border-
bottom: 2px solid black; border-top: 1px solid black; padding-left: 1px;
padding-right: 1px; } .subtotal input { width: auto; font: 11px Tahoma, sans-
serif; background-color: white; color: black; border: white; margin-right:
0px; } .total { font-size: 10px; border: 2px solid black; padding-left: 1px;
padding-right: 1px; } .total input { width: 102px; font: 11px Tahoma, sans-
serif; background-color: white; color: black; border: white; margin-right:
0px; } .fillout { font-size: 10px; border: 1px solid black; padding-left:
```

```
1px; padding-right: 1px; } .fillout input { width: auto; font: 11px Tahoma,
sans-serif; background-color: white; color: black; border: white; margin-
right: 0px; } .barcode { text-align: left; margin-right: 0px; padding-right:
0px; } .barcode input { width: 150px; font: 12px Tahoma, sans-serif;
background-color: white; color: black; border: 1px solid gray; margin-right:
1px; padding-left: 2px; padding-right: 2px; text-align: right; }
.fieldcellsmallest { width: 100px; font-size: 10px; text-align: left;
vertical-align: bottom; margin-right: 0px; padding-right: 0px; }
.fieldcellsmaller { width: 100px; font-size: 11px; text-align: left;
vertical-align: bottom; margin-right: 0px; padding-right: 0px; }
.fieldcellsmaller input { width: 200px; font: 11px Tahoma, sans-serif;
background-color: white; color: black; border: 1px solid gray; margin-right:
0px; cursor: text; } .fieldcellsmallerbold { vertical-align: top; width:
100px; font-size: 11px; font-weight: bold; margin-right: 0px; padding-right:
0px; } .fieldcellbold { width: 100px; font-size: 12px; font-weight: bold;
text-align: left; margin-right: 0px; padding-right: 0px; } .fieldcell {
width: 100px; font-size: 12px; text-align: left; margin-right: 0px; padding-
right: 0px; } .fieldcell input { width: 200px; font: 11px Tahoma, sans-serif;
 background-color: white; color: black; border: 1px solid gray; margin-right:
 0px; } .fieldcell select { width: 300px; font: 11px Tahoma, sans-serif;
background-color: white; color: black; border: 1px solid gray; margin-right:
0px; } .fieldcell textarea { width: 200px; font: 11px Tahoma, sans-serif;
background-color: white; color: black; border: 1px solid gray; margin-right:
0px; } .webnowlong input { width: 530px; } .webnowCaseNum input { width:
50px; font: 11px Tahoma, sans-serif; background-color: white; color: black;
border: 1px solid gray; margin-right: 0px; } .webnowCaseName input { width:
50px; font: 11px Tahoma, sans-serif; background-color: white; color: black;
border: 1px solid gray; margin-right: 0px; } .amount input { width: 92px;
font: 11px Tahoma, sans-serif; background-color: white; color: black; border:
1px solid gray; margin-right: 0px; text-align: right; } .labelcell { font:
9pt Tahoma,sans-serif; color: black; background-color: transparent; width:
190px; } .fieldcellLines { font: 9pt tahoma, sans-serif; color: #3670A7;
background-color: transparent; } .fieldcellLinesLabel { font: 9pt tahoma,
sans-serif; color: black; background-color: transparent; }
.fieldcellLineInput input { width: 125px; font: 9pt tahoma, sans-serif;
background-color: white; color: black; border: 1px solid gray; margin-right:
0px; } .fieldcellDate input { width: 75px; font: 8pt tahoma, sans-serif;
background-color: white; color: black; border: 1px solid gray; margin-right:
0px; text-align: right; } fieldset{ display:inline; border:none; margin-
left:0; padding-left:0; } fieldset legend { font: 9pt tahoma, sans-serif;
padding-left:0; padding-top: 1pt; margin-left:0; margin-bottom: 1; } textarea
{ font: 11px Tahoma, sans-serif; background-color: white; color: black;
border: 1px solid gray; margin-right: 0px; } input { font: 9pt tahoma, sans-
serif; background-color: white; color: black; border: 1px solid gray; }
input.ChecksAndRadios { cursor: default; width: 20px; font: 9pt tahoma, sans-
serif; border: none; vertical-align: middle; } input.red { border-left:
```

```
#cccccc; border-top: #cccccc; border-bottom: #666666; border-right: #666666;
background-color: #CC0000; font-weight: bold; font-size: 12px; color: white;
} input.green { border-left: #cccccc; border-top: #cccccc; border-bottom:
#666666; border-right: #666666; background-color: #008000; font-weight: bold;
font-size: 12px; color: white; } input.NormalButton { width: 100px; font:
12px tahoma, sans-serif; background-color: #cccccc; border-left: #black;
border-top: #black; border-bottom: #black; border-right: #black; } select {
width: 150px; margin-right: 1px; font: 9pt tahoma, sans-serif; background-
color: white; color: black; border: 1px solid gray; } label { font: 9pt
tahoma, sans-serif; padding-left:0; padding-top: 1pt; margin-left:0; margin-
bottom: 1; }
```

# Row management

## Row management example files

The Row Management Example shows you how to write code that enables users to add or delete rows from tables. This form example illustrates several text controls using a real world example of Accounts Payable vendor information fields. It also demonstrates how you can code your form so that your end users can click a button to repeat or delete data entry rows in tables. You can code the form to repeat one or several rows (a section) in tables at once according to your needs.

| Directory | Description |
|---|---|
| ap_invoice.xsl | The ap_invoice.xsl file contains an HTML table and XSL code examples you can follow to use repeating rows in your own forms with Perceptive Content. |
| ap_invoice.xml | The demo.xml file contains example elements so that you can follow this structure when you are coding your own form files. |
| formstylesheet.css | The formstylesheet.css file is an example of a shared form file. This file gives you an idea of how you can use CSS to style your own forms. Shared form files in Perceptive Content enable you to use the same file with many different forms so that you can single source the files that are used with more than one form. |

## Row management example ap_invoice.xsl file

This XSL file is part of the Row Management Example form. It contains an HTML table and XSL code examples you can follow to use repeating rows in your own forms with Perceptive Content. Refer to Row management example files to view the entire set of form files that comprise this example.

<?xml version="1.0" encoding="UTF-8"?><xsl:stylesheet version="1.0"

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"> <xsl:template match="/page"> <html> <head> <link href="FormStylesheet.css" rel="stylesheet" type="text/css" /> <title>Form</title> </head> <body> <form name="frmForm"> <table cellSpacing="0" cellPadding="0" border="0"> <tr> <td class="fieldcell" colspan="4"> <img src="Perceptive.jpg"></img><br/><br/> </td> </tr> <tr> <td class="fieldcell" nowrap="true" style="width:100px">Vendor ID:</td> <td class="fieldcell"> <input type="text" id="txtVendorCode" name="txtVendorCode" class="fieldcell" style="width:150px"> <xsl:attribute name="value"> <xsl:value-of select="VendorCode"/> </xsl:attribute> </input> </td> </tr> <tr> <td class="fieldcell" style="width:100px">Vendor Name:</td> <td class="fieldcell"> <input type="text" name="txtVendorName" class="fieldcell" style="width:150px"> <xsl:attribute name="value"> <xsl:value-of select="VendorName"/> </xsl:attribute> </input> </td> </tr> <tr> <td class="fieldcell" style="width:100px">Vendor Address:</td> <td class="fieldcell"> <input type="text" name="txtVendorAddress" class="fieldcell" style="width:150px"> <xsl:attribute name="value"> <xsl:value-of select="VendorAddress"/> </xsl:attribute> </input> </td> </tr> <tr> <td class="fieldcell" style="width:100px">Vendor City:</td> <td class="fieldcell"> <input type="text" name="txtVendorCity" class="fieldcell" style="width:150px"> <xsl:attribute name="value"> <xsl:value-of select="VendorCity"/> </xsl:attribute> </input> </td> </tr> <tr> <td class="fieldcell" style="width:100px">Vendor State:</td> <td class="fieldcell"> <input type="text" name="txtVendorState" class="fieldcell" style="width:150px"> <xsl:attribute name="value"> <xsl:value-of select="VendorState"/> </xsl:attribute> </input> </td> </tr> <tr> <td class="fieldcell" style="width:100px">Vendor Zip Code:</td> <td class="fieldcell"> <input type="text" name="txtVendorZipCode" class="fieldcell" style="width:150px"> <xsl:attribute name="value"> <xsl:value-of select="VendorZipCode"/> </xsl:attribute> </input> </td> </tr> <tr> <td class="fieldcell" style="width:134px" nowrap="true">Invoice Date (MM/DD/YYYY):</td> <td class="fieldcell"> <input type="text" name="txtInvoiceDate" class="fieldcell" maxlength="10" style="width:150px"> <xsl:attribute name="value"> <xsl:value-of select="InvoiceDate"/> </xsl:attribute> </input> </td> </tr> <tr> <td class="fieldcell" style="width:100px">Invoice Number:</td> <td class="fieldcell"> <input type="text" name="txtInvoiceNumber" class="fieldcell" maxlength="15" style="width:150px"> <xsl:attribute name="value"> <xsl:value-of select="InvoiceNumber"/> </xsl:attribute> </input> </td> </tr> <tr> <td class="fieldcell" style="width:100px">Purchase Order:</td> <td class="fieldcell"> <input type="text" name="txtPONumber" class="fieldcell" maxlength="30" style="width:150px"> <xsl:attribute name="value"> <xsl:value-of select="PONumber"/> </xsl:attribute> </input> </td> </tr> <tr> <td class="fieldcell" style="width: 100px">Project:</td> <td> <input type="text" id="txtProject" name="txtProject" class="fieldcell" style="width:150px"> <xsl:attribute name="value"> <xsl:value-of select="Project"/> </xsl:attribute> </input> </td> </tr> <tr> <td class="fieldcell" style="width:100px">Invoice Amount:</td> <td class="fieldcell"> <input type="text" name="txtVendorTotalAmt" id="txtVendorTotalAmt" class="amount" style="width:150px;text-align:right"> <xsl:attribute name="value"> <xsl:value-of select="TotalAmount"/> </xsl:attribute> </input> </td> </tr> <tr> <td class="fieldcell" colspan="2"> <hr style="width:400px"/> </td> </tr> </table> <table style="width:auto"><thead><tr> <td class="fieldcell">Item Code</td> <td class="fieldcell">Description</td> <td class="fieldcell">Unit Price</td> <td class="fieldcell">Quantity</td></tr></thead><tbody> <xsl:for-each select="LineItemDetailsBlock/accountInfo"> <tr repeat="row" delete="row"> <td class="fieldcell" nowrap="true" style="width:auto"> <input type="text" name="txtItemCode" id="txtItemCode" class="fieldcell" style="width:75px"> <xsl:attribute name="value"> <xsl:value-of select="ItemCode"/> </xsl:attribute> </input> </td> <td class="fieldcell" nowrap="true" style="width:auto"> <input type="text" id="txtDescription" name="txtDescription" class="fieldcell" style="width:75px"> <xsl:attribute name="value"> <xsl:value-of select="Description"/> </xsl:attribute> </input> </td> <td class="fieldcell" nowrap="true" style="width:auto"> <input type="text" id="txtUnitPrice" name="txtUnitPrice"

class="fieldcell" style="width:75px; text-align:right;"> <xsl:attribute name="value"> <xsl:value-of select="UnitPrice"/> </xsl:attribute> </input> </td> <td class="fieldcell" valign="middle"> <input type="text" name="txtQuantity" class="amount" style="width:75px; text-align:right;"> <xsl:attribute name="value"> <xsl:value-of select="Quantity"/> </xsl:attribute> </input> </td> </tr> </xsl:for-each> </tbody> <tfoot> <tr> <td class="fieldcell" colspan="5"> <hr style="width:400px"/> </td> </tr> <tr> <td colspan="5"> <input type="button" value="Add Section" class="green" repeat="row"/>    <input type="button" value="Delete Section" class="red" delete="row"/> </td> </tr> </tfoot> </table> </form> </body> </html> </xsl:template></xsl:stylesheet>

### Row management example ap_invoice.xml file

This XML file is part of the Row Management Example form. It contains example elements so that you can follow this structure when you are coding your own form files. Refer to Row management example files to view the entire set of form files that comprise this example.

```
<?xml version="1.0" encoding="UTF-8"?><page> <VendorCode></VendorCode>
<VendorName></VendorName> <VendorAddress></VendorAddress>
<VendorCity></VendorCity> <VendorState></VendorState>
<VendorZipCode></VendorZipCode> <InvoiceNumber></InvoiceNumber>
<PONumber></PONumber> <InvoiceDate></InvoiceDate> <TotalAmount></TotalAmount>
 <Project></Project> <LineItemDetailsBlock> <accountInfo>
<ItemCode></ItemCode> <Description></Description> <UnitPrice></UnitPrice>
<Quantity></Quantity> <DistAmount></DistAmount> </accountInfo>
</LineItemDetailsBlock></page>
```

# Create forms

## Create a form

To create a new form you can build in Form Designer, complete the following steps.

1. In **Management Console**, in the left pane, under **Select Department**, select a department from the list.
2. In the left pane, click **Forms**.
3. In the right pane, on the **Forms** tab, click **New**.
4. Type a unique name for your form and then press `ENTER`.

## Open a saved form design

To open a saved form design, perform the following steps.

You cannot open a published form using this method.

1. In **Form Designer**, on the **File** toolbar, click the **Open** button.
2. In the **Open Design** dialog box, select the name of your saved form design.
3. Click **Open**.

## Copy a form

To copy a form specific elements in a form, perform the following steps.

1. In **Form Designer**, on the **File** toolbar, click the **Open** button.
2. In the **Open Form** dialog box, select the form and click **Open**.
3. On the **File** toolbar, click the **New** button.
4. To copy the previously saved form, click the tab in the center of the screen.
5. Select any elements you want to copy.
6. On the **Edit** toolbar, click the **Copy** button.
7. To paste to the new form, in the tab in the center of the screen, click the tab.
8. On the **Edit** toolbar, click the **Paste** button.

## Preview a form

To preview the appearance of a form after publication, perform the following steps.

1. Open a form in an open design tab in the center of the screen.
2. On the **File** toolbar, click the **Form Preview** button.

## View another form while designing

To view another form while designing, complete the following steps.

1. On the **File** toolbar, click the **Open** button.
2. In the **Open Design** dialog box, select the form you want to view and click **Open**.
3. In **Form Designer**, select the tab labeled with the form name.

## Filter the form properties

To filter the custom form properties in the Properties pane, perform the following step.

- In the Properties pane in the Filter field, type the name or partial name of the custom property.

## Select a style

To change the appearance of your form, you can select a different style. To select a style, perform the following step.

- On the Edit toolbar, select an option in the Style list box.

## Copy text from an application

To copy text from an application and add it to a form, perform the following steps.

1. Copy the appropriate text from the third-party application.
2. Place your cursor in a blank section of the open **Design** tab in the center of the screen.

3. In **Form Designer**, on the **Edit** toolbar, click the **Paste** button.

# Select the font for the form

To select the font for the form, perform the following steps.

1. In an open **Design** tab in the center of the screen, ensure you do not have a field or composite set selected.
2. On the **Edit** toolbar, click the **Font** button.
3. In the **Font** dialog box, select the font type, style, size, and color.
4. Click **OK**.

# About data synchronization

Data synchronization is the process that enables you to choose how to synchronize information between the form fields and custom properties in Perceptive Content.

You can set the system to automatically update the associated custom properties in Perceptive Content based on the information a user enters in the form fields. This allows the system to index any information a user enters in form fields. You can configure the synchronization separately for each composite or custom property added to your form. When you set this automatic update, if any custom property information is entered in ImageNow Server, the form information overwrites it. You can also remove the synchronization so that the system does not update custom properties in ImageNow Server based on the information a user enters in the form fields. When you remove data synchronization, the system maintains the current custom property set in Perceptive Content regardless of the form field data.

# Synchronize data

You can set the system to automatically update the associated custom property in Perceptive Content based on the information a user enters in the form field. To synchronize data, perform the following steps.

When you set this automatic update, if any custom property information is entered in Perceptive Content, the form information overwrites it. You can also remove the synchronization so that the system does not update custom properties in ImageNow Forms Server based on the information a user enters in the form fields.

1. Select a property in an open design tab in the center of the screen.
2. In the **Property Attributes** pane, in the **Synchronize Data** list, select **Enabled** or **Disabled**.

# About verifying data in fields

You can verify data in fields to ensure that the correct information resides in the form fields and the custom properties in Perceptive Content.

When you synchronize form fields and the custom properties in Perceptive Content, the system automatically verifies that the user entered the correct data type in a field. If a user enters an incorrect value type and selects another field, the system automatically clears the data. If a user enters the correct type of value in the field and the format is incorrect, the system automatically updates the information to reflect the correct format.

For example, if a user enters an alphabetic character in a field that accepts only numbers, the system clears the value when the user selects another field. However, if the correct type of data is entered, the field may perform additional formatting. For example, if the correct number format includes a decimal, after the user enters a number without a decimal in a field, the system automatically adds a decimal value of .00.

# Verify data in fields

When you disable data synchronization, you can choose to verify or disregard the format of the information a user enters in a field. When you enable data synchronization, the system automatically verifies formatting. To verify data in fields, perform the following steps.

1. In an open design tab in the center of the screen, select a field.
2. In the **Property Attributes** pane, in the **Synchronize Data** field, select **Disabled**.
3. Do one of the following actions:
   - To verify the data of the information a user enters in a field, in the **Verify Data Format** list, select the check box.
   - To disregard the data of the information a user enters in a field, in the **Verify Data Format** list, clear the check box.

# Make fields required

To require a user to enter a value in a field before successfully submitting a form published to ImageNow Forms Server, complete the following steps.

Only forms published to ImageNow Forms Server can contain required fields. A user must complete all required fields before successfully submitting a form.

1. In an open design tab in the center of the screen, select a field or a composite set.
2. In the **Property Attributes** pane, in the **Require Fields** list, select one or more field name check boxes.

# Save a form design

To save a form design, perform the following steps.

A user with access to Form Designer can open any saved form design.

1. On the **File** toolbar, click the **Save** button.
2. In the **Save Design** dialog box, in the **Name** field, enter a file name.
3. Optional. In the **Description** field, enter text.
4. Click **OK**.

# Export form files

You can export form file to manually enhance the form using external editing tools. To export form files, perform the following steps.

You cannot export published forms.

1. Open a saved form design.
2. On the **File** toolbar, click the **Export** button.
3. In the **Browse For Folder** dialog box, select a folder destination for the files.
4. Click **OK**.

# Set form privileges

To set form privileges, complete the following steps.

1. In **Management Console**, in the left pane, under **Select Department**, select a department from the list.
2. In the left pane, click **Forms**.
3. In the right pane, in the **Form** list, select the form you want and then click **Modify**.
4. In the **Form** dialog box, in the left pane, click **Security**.
5. In the right pane, in the **Select a presentation** box, select a presentation.
6. Under **Users/Groups**, click **Add**.
7. In the **Form** dialog box, under **Users/Groups**, select the user or group you want to grant privileges to.
8. In the **Privileges** list, click the column to the left of the **Forms** privilege so that it is displayed for the following privileges as needed:
   - To view the form and its data, grant the View privilege. The View privilege is required to make the other privileges effective.
   - To modify the data for that form, grant the Modify privilege.
   - To enter data into new instances of that form, grant the Create privilege.
   - To delete data in that form, grant the Delete privilege.
   - Repeat these steps for each user and group who needs access to the form.

# Associate a form with a workflow queue

To associate a form with a workflow queue, complete the following steps.

1. In **Management Console**, in the left pane, under **Select Department**, select a department from the list.
2. In the left pane, click **Workflow**.
3. On the **Workflow** tab, select a process and then click **Modify**.
4. In the **Workflow Designer** dialog box, select a queue and then press `ENTER`.
5. In the **Queue Properties** dialog box, in the left pane, click **Forms**.
6. You can perform the following associations:

- To attach forms to this queue, click **Add**, in the **Select Forms** dialog box, select the forms you want to add, and then click **OK**.
- To remove a form from this queue, select the form in the list and then click **Remove**.
- To change the default form associated with this queue, select the form in the list and click **Set as Default**.

7. Repeat the previous step for any additional queues.
8. When you are done associating forms, click **OK**.

# Customize forms

## Create a customized form section

To create a section with customized text in your form, complete the following steps.

1. Drag **Section** from the **Toolbox** pane to an open **Design** tab in the center of the screen.
2. Select and delete the default text, and then enter new text to describe the section.
3. Drag a custom property from the **Properties** pane to the new section.

## Create a section with two columns

To add a section with two columns to a form design, perform the following steps.

1. In **Form Designer**, from the **Toolbox** pane, drag **Two Column Section** to an open **Design** tab in the center of the screen.
2. Optional. In the **Property Attributes** pane, do one of the following:

| Situation | Steps |
|---|---|
| To evenly distribute the width of both columns | • Select Balance Column |
| To size the column according to the content size | • Deselect Balance Column |

## Reorder composite properties

To reorder composite properties, complete the following steps.

1. Add a composite field to an open **Design** tab in the center of the screen.
2. In the **Design** tab, select the composite property.
3. In the **Attributes** pane, select the **Set Order** row and click the ellipsis button.
4. In the **Set Order** dialog box, select a property and click **Move Up** or **Move Down** to change the order of the property, as needed.
5. Click **OK**.

# Add a field

To add a field to a form, complete the following steps.

You can use each field on a form one time only.

1. Click and drag a custom property from the **Properties** pane to an open **Design** tab in the center of the screen.
2. Optional. To choose the configuration of the rows for a composite property, right-click and drag a composite property from the **Properties** pane to the tab in the center of the screen:
   - To display each property on individual rows, click **Singular**.
   - To display the properties as multiple fields on one row, click **Repeating**.

# Add text to a form

To add text to a form, complete the following steps.

You can add text to areas of the form that are not occupied by fields, rows, or images.

1. Place your cursor in a blank section of a open **Design** tab in the center of the screen.
2. Type the text you want to display.
3. Optional. To change the font, do the following actions:
   1. Highlight the text. On the **Edit** toolbar, click the **Font** button.
   2. In the **Font** dialog box, change the appearance of the text including color, style, and size as needed.
   3. Click **OK**.

# Add a text area to a form

To add an area where users can enter text to a form, complete the following steps.

1. Drag Text area from the **Toolbox** pane to an open **Design** tab in the center of the screen.
2. Optional. Click and drag the box to reposition it and modify the size.

   If a user enters more text than the size of the text area allows, a scroll bar appears to allow scrolling within the text area.

# Add a link to a form

To add a link to a form, complete the following steps.

1. Drag Link from the **Toolbox** pane to an open Design tab in the center of the screen.
2. Optional. To create a link from customized text, highlight text in the form and drag Link from the **Toolbox** pane to the highlighted text in the tab in the center of the screen.
3. In the **Hyperlink** dialog box, in the **Display Text** field, enter the display text for the link.
4. In the **URL** field, enter the address.
5. Click **OK**.

# Add a picture to a form

To add a picture to a form, complete the following steps.

1. Drag **Picture** from the **Toolbox** pane to an open **Design** tab in the center of the screen.
2. In the **Picture** dialog box, select the picture and click **Open**.

# Add an edit box to a form

To add at box where users can enter a single line of text to a form, complete the following steps.

1. Drag **Edit box** from the **Toolbox** pane to an open **Design** tab in the center of the screen.
2. Optional. Click and drag the box to reposition it and modify the size.

# Add a background to a property

To add a background to a property, perform the following steps.

1. In an open **Design** tab in the center of the screen, select a field or composite set.
2. In the **Property Attributes** pane, select one of the following options:
   - Optional. Under **Background**, in the **Color** field, select a color.
   - Optional. To select an image, complete the following steps.
3. Under **Background**, in the Image field, select **Open Image**.
4. In the **Picture** dialog box, select a picture.
   - Optional. Under **Background**, in the **Tile** field, select a repeating pattern for the image.
5. Click **Open**.

# Modify a border

To change the style or color of a border, perform the following steps.

1. In an open **Design** tab in the center of the screen, select a field or composite set.
2. In the **Property Attributes** pane, complete any of the following options:
   - Optional. Under **Border**, in the **Color** field, select a color.
   - Optional. Under **Border**, in the `Style` field, select a style.
   - Optional. Under **Border**, in the **Width** field, enter a positive number to represent the thickness of the border in pixels.

# Publish forms

## About publishing forms

You can publish forms by launching a publishing wizard that leads you through each step of publication, so that other users can access them in Form Designer.

You can select several options based on the publishing location and form type. The publishing wizard varies the information you are required to complete according to the selections you make. For example, if you choose to use the ImageNow Forms Server to publish a document type form online, the publishing wizard requires you to enter a document type name, select a drawer, and enter at least one document key. Alternatively, if you publish a document type form only in Form Designer, the only required information is a document type name.

## Create a sequence

When you publish a form, you can create a sequence to define the folder name of document keys. To create a sequence, perform the following steps.

1. In **Form Designer**, on the **File** toolbar, click the **Publish** button.

   Clicking the Publish button opens a publishing wizard. Complete the required information and click Next to open successive pages.
2. In the **Document Keys** page, do the following actions:
   1. Select Sequence Number in the **source** column.
   2. In the **Details** column, select **Edit Sequence**.
   3. In the **Sequence Number** dialog box, click **New**.
   4. Click **Next**.

   **Note:** To view the drop-down list in the column rows, click in the row.
3. In the **New Sequence** page, do the following actions:
   1. In the **Name** box, type a name for the sequence.
   2. From the **Base** list, select the numeral system you want to use.
   3. In the **Width** box, type the number of digits you want to use.
   4. In the **Increment** box, type the amount that you want added to the sequence.
   5. In the **Prefix** box, type a static text string that you want to precede the sequence.
   6. In the **Suffix** box, type a static text string that you want to follow the sequence.
   7. Click **OK**.

# Publish a form as a document in Perceptive Content

When you publish a form as a document in Perceptive Content, it can appear in Perceptive Content in place of a scanned image or as a form in the Form pane. To publish a form as a document in Perceptive Content, perform the following steps.

1. To open the publishing wizard, on the **File** toolbar, click the **Publish** button.
2. In the **Published Location** page, select **Perceptive Content** and click **Next**.
3. In the **New Item** page, to create the document type, complete the following substeps.
    1. In the **Type** field, select **Document**.
    2. In the **Name** field, type the name of the new document.
    3. Optional. In the **Description** field, type a description.
    4. Click **Finish**.

# Publish a form in a folder in Perceptive Content

To publish a form in a folder in Perceptive Content, complete the following steps.

1. On the **File** toolbar, click the **Publish** button.
2. In the **Published Location** page, select **Perceptive Content**, and click **Next**.
3. To create the folder type, in the **New Item** page, complete the following substeps.
    1. In the **Type** field, select **Folder**.
    2. In the **Name** field, type the name of the new folder type.
    3. Optional. In the **Description** field, type a description.
    4. Click **Finish**.

# Publish a form as a document to Forms Server

To publish a form as a document to Perceptive Forms Server, complete the following steps.

1. To open the publishing wizard, on the **File** toolbar, click the **Publish** button.
2. In the **Published Location** page, select **Forms Server and ImageNow**, and click **Next**.
3. To create the document type, in the **New Item** page, complete the following substeps.
    1. In the **Type** list, select **Document**.
    2. In the **Name** box, type the name of the new document.

        **Note:** Enterprise Software does not support using special characters, such as apostrophes, in the form name.

    3. Optional. In the **Description** box, type a description.
    4. Optional. In the **Queue** list, select a queue.
    5. Optional. Select or clear the **Allow attachments to submitted forms** check box.

        **Note:** Attachments appear as additional pages in Perceptive Content.

6. Click **Next**.
4. To populate document keys when submitting a form from an embedded URL, in the **Document Keys** page, select a drawer and select a source for at least one field. Enter the appropriate corresponding information in the **Details** column.

   To view the lists in the columns, click in the row.
5. Click **Next**.
6. Optional. To select specific file types that users can attach to the form, in the **Attachment File Types** page, clear the **Allow all attachments** check box, and complete the following substeps.
   1. To select a file type, in the **Filter by** list, select a category, select an available file type, and then click **Add**, or click **Add All** to add all file types for the category.
   2. To add a file type on a per form publication basis, click **New** and create the new extension.
7. Click **Next**.
8. Optional. To customize the size and number of files a user can attach to the form, in the **Attachment Size and Number** page, complete the following substeps.
   1. To set a custom maximum file size for attachments, select **Custom** and type a maximum size, in kilobytes.
   2. To customize the minimum number of file attachments for the form, select **Custom** and select a value.
   3. To customize the maximum number of file attachments for the form, select **Custom** and select a value.

   **Note:** The minimum number of file attachments cannot exceed the maximum.
9. Click **Next**.
10. Optional. To map URL parameters to form elements, in the **URL Parameters** page, complete the following substeps.
    1. Click **Add**.
    2. In the **Name** column, enter parameters that you want to map to form elements.
    3. In the **Form field** column, select the field in the form.
    4. Optional. Click **Add** to add rows and URL parameters.
11. Click **Finish**.

# Publish a form as a folder to Forms Server

To publish a form as a folder to Perceptive Forms Server, perform the following steps.

This type of form is embedded using a URL so that a user can access the form on a web application server as well as ImageNow Forms Server or the Forms pane.

1. On the **File** toolbar, click the **Publish** button.
2. In the **Published Location** page, select **Forms ServerandPerceptive Content**.
3. To create the project type, in the **New Item** page, complete the following substeps:
   1. In the **Type** field, select **Project**.

2. In the **Name** field, type the name of the new project type.

   **Note:** Perceptive Software does not support using special characters, such as apostrophes, in the form name.

3. Optional. In the **Description** field, type a description.
4. In the **Queue** field, select a queue.
5. Click **Next**.

4. To populate the project name when a user submits a form from an embedded URL, in the **Project Name** page, complete the following substep:
   1. To view the drop-down list in the column rows, click in the row.
5. Configure the following Source options:
   1. **System drawer** - Select a system drawer
   2. **Literal** - Type static text
   3. **Unique ID**- Generates a unique ID number
   4. **Form Field** - Select a form field
   5. **Sequence Number** - Select a predefined sequence or "Create a sequence."
   6. **Time Stamp** - Assigns the time and date when the user submits the form.
   7. Click **Next**.
6. Optional. To map URL parameters to form elements, in the **Publish URL Parameters** page, complete the following substeps:
   1. Click **Add**.
   2. In the **Name** column, enter parameters to map to form elements.
   3. In the **Form** field column, select the field in the form.
   4. Optional. Click **Add** to add rows and add additional URL parameters.
   5. Click **Finish**.

# Configure Form Designer

## What is imagenowforms.xml?

You can use the *imagenowforms.xml* file, located in the *[drive:]\inserver\etc* directory, to create and modify form definitions available for use by Forms Server.

You can have as many form definitions in the imagenowforms.xml file as you need. However, you must have a form definition in the imagenowforms.xml file for each form you plan to deploy. There are two types of form definitions: the document form definition and the folder form definition. You cannot deploy a single form definition for both a folder type and a document type.

A form definition can store two types of attributes for a form: a literal value or a dynamic value. Note that when defining a folder form, only the FolderName element of the folder form definition can accept a dynamic value. This is the only field on the folder form definition that obeys dynamic fields.

# imagenowforms.settings parameters

The following table lists the group, parameters, and sample values for the imagenowforms.settings file.

## [PerceptiveContentForms] Perceptive Content Server settings

### host

***Sample value: 168.154.21.54***

**Description**

The host name of Perceptive Content Server. You can specify the host name or the IP address.

### port

***Sample value: 6000***

**Description**

The port number of Perceptive Content Server. By default, Perceptive Content utilizes port 6000.

### heartbeat

***Sample value: TRUE or FALSE***

**Description**

Specifies whether to send a heartbeat message to the server at a regular interval to maintain the session when no information is sent.

By default, this parameter is set to FALSE. If set to TRUE, the Forms session will never time out. The recommended setting is FALSE.

### logontimeout

***Sample value: 30***

**Description**

The socket timeout for the logon process in seconds.

### defaulttimeout

***Sample value: 30***

**Description**

The socket timeout for communication with Perceptive Content Server in seconds.

### sso

***Sample value: TRUE or FALSE***

**Description**

Enables users to log into Forms Server directly from your SSO provider logon. After setting up single sign-on, you can log into your SSO provider for direct access to Forms Server.

### sso.key

***Sample value: tokenpassword***

**Description**

When a single sign-on is enabled, this setting specifies the token Forms Server uses with Perceptive Content Server to indicate successful authentication.

### sso.method

***Sample value: com.imagenowservlet.auth.RemoteUserAuthenticator***

**Description**

Based on the provider, this setting specifies the type of single sign-on.

## [PerceptiveContentForms] Application server settings

### log

***Sample value: TRUE or FALSE***

**Description**

Specifies whether to log Forms Server errors and warning messages.

When the value is TRUE, logging is enabled. When the value is FALSE, logging is disabled. The recommended setting is TRUE.

### logdirectory

***Sample value: C:\logs***

**Description**

The file directory in which Forms Server stores logged error and warning messages. This path must exist.

### log-unencrypted

***Sample value: TRUE or FALSE***

**Description**

Specifies whether log files are encrypted. The default is FALSE, which means the log files are encrypted.

**Note:** The common servlet class defaults to TRUE if the setting is blank.

### form-presentation-cache-timeout

***Sample value: 60***

**Description**

The socket timeout for caching form files in seconds. The default setting is 60.

### integration-server-path

***Sample value: http://168.154.21.86:8080/integrationserver***

**Description**

Specifies the path to Integration Server.

## [Locale]

This optional section enables you to customize the behavior of the application to perform actions such as changing the language to one different from the operating system's locale. For more information on changing the Forms Server language, refer to Change the Forms Server language.

**Note:** This feature is available when the associated Perceptive Content language pack is available.

# Prevent the URL from appearing on the browser status bar

To prevent the URL from opening on the browser status bar by integrating Forms with another web application, perform the following steps.

1. Open the web application that uses the forms link.
2. Replace the Forms link that you want to hide with the following HTML code: `<a href="http://forms url here" onMouseOver="window.status='Complete Form'; return true" onMouseOut="window.status=''; return true">View</a>`

**Result** After performing these steps, when you move your mouse over the Forms link, the status bar displays "Complete Form" rather than the URL associated with the Forms link.

# What are data and configuration parameters?

Configuration and data parameters work together to display the Perceptive Content forms information the way you want it to appear.

Configuration parameters allow you to configure the application that displays the form in the browser. You use data parameters to pre-populate form fields. The imagenowforms.xml file specifies the data parameter name and the field where the information is populated in the form.

# Data parameters

You use data parameters to pre-populate form fields. The imagenowforms.xml file specifies the data parameter name and the field where the information is populated in the form. You can reference this information from the URL or single sign-on functionality. The URL parameter specifies the data parameter name and information to be populated on the form.

## <First_Name>

**Description**

An example of a data parameter that uses a URL is a user ID. This parameter specifies the user name passed into a form field named First Name.

**Value**

John

**Examples**

```
<DataParam name= First Name  path= /form/student/first_name />
```

http://localhost:8080/psw-content-forms-server/fs?form=DocFormTest&Fist_Name=John

## <username>

**Description**

An example of a data parameter that uses a single sign-on is a username. This parameter specifies the user name passed into a form field.

**Value**

Jsmith

**Example**

```
<DataParam name= username  path= /form/student/username />
```

# Configuration parameters

You can add configuration parameters to the imagenowforms.xml file or the embedded URL. The examples that appear in the following table show a format for the configuration parameter in the XML file and in the URL. You do not need to add the parameter to both places. If you add a parameter to the XML file and to the URL, the configuration parameter takes priority over the URL value unless you add a URL override parameter to the imagenowforms.xml file.

## attachmentsAction

### Description

Specifies the action the system performs when a user clicks the button. A button can perform multiple tasks. For example, the Attachments button can launch the attachments dialog box and then save the form. The default value is attachments.

### Value

Any combination of the following action names separated by a comma: save, reset, print, or attachments.

### Examples

```
<ConfigParams><ConfigParam name="attachmentsActions"
value="attachments,save"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&attachmentsActions=attachments,save

## attachmentsBorderColor

### Description

Specifies the border color for the Attachments button that appears on the button bar at the bottom of the form. The default value is #C8C8C8.

You can supply any CSS color name or RGB value recognized by the browser. For example, white, (255,255,255), #FFFFFF.

Forms Server recognizes any three decimal values, separated with commas, as an RGB value. For example, 255, 255, 255.

### Value

CSS color name or an RGB value

**Examples**

```
<ConfigParams><ConfigParam name="attachmentsBorderColor"
value="red"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&attachmentsBorderColor=red

## attachmentsColor

**Description**

Specifies the color for the Attachments button that appears on the button bar at the bottom of the form. The default value is #F0F0F0.

You can supply any CSS color name or RGB value recognized by the browser. For example, white, (255,255,255), #FFFFFF.

Forms Server recognizes any three decimal values, separated with commas, as an RGB value. For example, 255, 255, 255.

**Value**

CSS color name or an RGB value

**Examples**

```
<ConfigParams><ConfigParam name="attachmentsColor"
value="red"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&attachmentsColor=red

## attachmentsMaximumCount

**Description**

Specifies the maximum number of files that can be attached to the form, and must be equal to, or larger than, attachmentsMinimumCount. This does not include any PDFs sent as attachments when the saveTransformAsAttachments parameter is set to true. The default is no limit.

**Value**

Any non-negative 32-bit integer

**Examples**

```
<ConfigParams><ConfigParam name="attachmentsMaximumCount"
value="5"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&attachmentsMaximumCount=5

## attachmentsMaximumSize

**Description**

Specifies the maximum size for each attachment, in kilobytes. This does not include any PDFs sent as attachments when the saveTransformAsAttachments parameter is set to true. The default is no limit.

**Value**

Any non-negative 32-bit integer

**Examples**

```
<ConfigParams><ConfigParam name="attachmentsMaximumSize"
value="2147483646"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&attachmentsMaximumSize=2147483646

## attachmentsMinimumCount

**Description**

Specifies the minimum number of files that must be attached to submit the form, and must be equal to, or smaller than, attachmentsMaximumCount. This does not include any PDFs sent as attachments when the saveTransformAsAttachments parameter is set to true. The default is no limit.

**Value**

Any non-negative 32-bit integer

**Examples**

```
<ConfigParams><ConfigParam name="attachmentsMinimumCount"
value="1"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&attachmentsMinimumCount=1

## attachmentsPosition

**Description**

Specifies the placement of the button in regards to the other buttons. The first button in a row is represented by the number 1, while 4 represents the last button in the row. The default value is 4.

**Value**

An integer between 1 and 4

**Examples**

```
<ConfigParams><ConfigParam name="attachmentsPosition"
value="1"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server /fs?form=FormName&attachmentsPosition=1

## attachmentsText

**Description**

Specifies the text for the Attachments button that appears on the button bar at the bottom of the form.

**Note:** If you remove all text from the button, an icon appears to represent the attachments action.

**Value**

Any text

**Examples**

```
<ConfigParams><ConfigParam name="attachmentsText"
value="AddFile"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&attachmentsText=Add%20File

## attachmentsTextColor

**Description**

Specifies the color for the text on the Attachments button that appears on the button bar at the bottom of the form. The default value is #F0F0F0.

You can supply any CSS color name or RGB value recognized by the browser. For example, white, (255,255,255), #FFFFFF

Forms Server recognizes any three decimal values, separated with commas, as an RGB value. For example, 255, 255, 255.

**Value**

CSS color name or an RGB value

**Examples**

```
<ConfigParams><ConfigParam name="attachmentsTextColor"
value="red"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&attachmentsTextColor=red

## attachmentsTooltip

**Description**

Specifies the text that appears when a user places the pointer over the Attachments button. The default value is attachments.

**Value**

Any text

**Examples**

```
<ConfigParams><ConfigParam name="attachmentsTooltip" value="Add
Files"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&attachmentsTooltip=Add%20Files

## attachmentsTypes

**Description**

Specifies the file extensions permitted for attachments, separated by commas. Do not include a leading period.

**Value**

File type extension

**Examples**

```
<ConfigParams><ConfigParam name="attachmentsTypes"
value="pdf,jpg,png"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&attachmentsTypes=pdf,jpg,png

## attachmentsVisible

**Description**

Specifies whether the Attachments button appears on the button bar at the bottom of the form. The default value is TRUE.

**Value**

TRUE or FALSE

**Examples**

```
<ConfigParams><ConfigParam name="attachmentsVisible"
value="FALSE"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&attachmentsVisible=FALSE

## barColor

### Description

Specifies the color of the button bar that appears at the bottom of the form. The default color is #888888.

You can supply any CSS color name or RGB value recognized by the browser. For example, white, (255,255,255), #FFFFFF.

Forms Server recognizes any three decimal values, separated with commas, as an RGB value. For example, 255, 255, 255.

### Value

CSS color name or an RGB value

### Examples

```
<ConfigParams><ConfigParam name="barColor" value="green"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&barColor=green

## displayStatus

### Description

Determines whether the system displays the form information at the bottom of the screen. The default is TRUE.

### Value

TRUE or FALSE

### Examples

```
<ConfigParams><ConfigParam name="displayStatus"
value="FALSE"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&displayStatus=FALSE

## error

### Description

Specifies the error message that displays when there is an error submitting a form. However, if the error is from Perceptive Content Server, the text from the Forms Server text does not appear.

### Value

Any text

#### Examples

```
<ConfigParams><ConfigParam name="error"
value="Error%20submitting%20form"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&error=Error%20submitting%20form

## fullScreenSuccess

#### Description

Determines whether the configuration message displays on a new screen after you submit the form. If the value is FALSE, the message appears on the button bar of the submitted form. The default is TRUE.

#### Value

TRUE or FALSE

#### Examples

```
<ConfigParams><ConfigParam name="fullScreenSuccess"
value="TRUE"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&fullScreenSuccess=TRUE

## height

#### Description

Supplies the height of the web application within the parent web page. You can use either pixels or percentages for the value. For pixels, an example is width=250px. For percentage, an example is height=50%25. The %25 is the URL escape code for %.

**Note:** Scaling the application size larger can make the buttons inaccessible outside of the viewing area.

#### Value

Pixels or percentage

#### Examples

```
<ConfigParams><ConfigParam name="height" value="250px"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&height=250px

## printActions

**Description**

Specifies the action the system performs when a user clicks the button. A button can perform multiple tasks. For example, the Print button can print and then save the form. The default value is print.

**Value**

Any combination of the following action names separated by a comma: save, reset, print, or attachments.

**Examples**

```
<ConfigParams><ConfigParam name="printActions"
value="print,save"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&printActions=print,save

## printBorderColor

**Description**

Specifies the color of the Print button that appears at the bottom bar of the form. The default color is #C8C8C8.

You can supply any CSS color name or RGB value recognized by the browser. For example, white, (255,255,255), #FFFFFF.

Forms Server recognizes any three decimal values separated with commas as an RGB value. For example, 255, 255, 255.

**Value**

CSS color name or an RGB value

**Examples**

```
<ConfigParams><ConfigParam name="printBorderColor"
value="red"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&printBorderColor=red

## printColor

**Description**

Specifies the color for the Print button that appears on the button bar at the bottom of the form. The default value is #F0F0F0.

You can supply any CSS color name or RGB value recognized by the browser. For example, white, (255,255,255), #FFFFFF

Forms Server recognizes any three decimal values separated with commas as an RGB value. For example, 255, 255, 255.

**Value**

CSS color name or an RGB value

**Examples**

```
<ConfigParams><ConfigParam name="printColor" value="red"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&printColor=red

## printPosition

**Description**

Specifies the placement of the button in regard to the other buttons. The first button in a row is represented by the number 1, while 4 represents the last. The default value is 3.

**Value**

An integer between 1 and 4

**Examples**

```
<ConfigParams><ConfigParam name="printPosition"
value="1"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&printPosition=1

## printText

**Description**

Specifies the text for the Print button that appears on the button bar at the bottom of the form.

**Note:** If you remove all text from the button, an icon appears to represent the print action.

**Value**

Any text

**Examples**

```
<ConfigParams><ConfigParam name="printText"
value="PrintForm"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&printText=Print%20Form

## printTextColor

**Description**

Specifies the color for the text on the Print button that appears on the button bar at the bottom of the form. The default value is black.

You can supply any CSS color name or RGB value recognized by the browser. For example, white, (255,255,255),#FFFFFF.

Forms Server recognizes any three decimal values separated with commas as an RGB value. For example, 255, 255, 255.

**Value**

CSS color name or an RGB value

**Examples**

```
<ConfigParams><ConfigParam name="printTextColor"
value="red"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&printTextColor=red

## printTooltip

**Description**

Specifies the text that appears when a user places the pointer over the Print button. The default value is Print Form.

**Value**

Any text

**Examples**

```
<ConfigParams><ConfigParam name="printTooltip"
value="Print"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&printTooltip=Print

## printTransform

**Description**

Determines whether the system generates and opens a PDF in a new browser window when a user initiates a print action.

**Note:** If the PDF file does not generate successfully, the Print dialog box appears to complete the print action. Also note that the PDF files display in a new browser window. To view the file, you must disable pop-up blockers.

**Value**

TRUE or FALSE

**Examples**

```
<ConfigParams><ConfigParam name="printTransform"
value="TRUE"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&printTransform=TRUE

## printVisible

**Description**

Specifies whether the Print button appears on the button bar at the bottom of the form. The default value is TRUE.

**Value**

TRUE or FALSE

**Examples**

```
<ConfigParams><ConfigParam name="printVisible"
value="FALSE"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&printVisible=FALSE

## prompt

**Description**

Specifies the text in the message that displays when a user attempts to close the window without saving changes to a form. The default message is  The form might contain unsaved data.

**Note:** The promptOnClose parameter must be set to TRUE for the prompt parameter  s text to display.

**Value**

Any text

**Examples**

```
<ConfigParams><ConfigParam name="prompt"
value="Save%20form%20now?"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&prompt=Save%20form%20now?

## promptOnClose

**Description**

Determines whether the message from the prompt parameter appears when a user attempts to close the window without saving. The default is FALSE.

**Value**

TRUE or FALSE

**Examples**

```
<ConfigParams><ConfigParam name="promptOnClose"
value="TRUE"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&promptOnClose=TRUE

## redirectUrl

**Description**

After a user clicks the button to save the form, the parameter specifies a URL address to redirect the user. Save must be the last action configured on the button.

**Note:** The URL must include the http:// preface. You must use encodings to escape special characters.

**Value**

Any valid URL

**Examples**

```
<ConfigParams><ConfigParam name="redirectUrl"
value="http://www.perceptivesoftware.com /></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&redirectUrl=http://www.perceptivesoftware.com

## resetActions

**Description**

Specifies the action the system performs when a user clicks the button. A button can perform multiple tasks. For example, the Reset button can save and then reset the form. The default value is reset.

**Value**

Any combination of the following action names separated by a comma: save, reset, print, or attachments.

**Examples**

```
<ConfigParams><ConfigParam name="resetActions"
value="save,reset"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&resetActions=save,reset

## resetBorderColor

**Description**

Specifies the color for the text on the Reset button that appears on the button bar at the bottom of the form. The default value is #C8C8C8.

You can supply any CSS color name or RGB value recognized by the browser. For example, white, (255,255,255), #FFFFFF.

Forms Server recognizes any three decimal values separated with commas as an RGB value. For example, 255, 255, 255.

**Value**

CSS color name or an RGB value

**Examples**

```
<ConfigParams><ConfigParam name="resetBorderColor"
value="red"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&resetBorderColor=red

## resetColor

**Description**

Specifies the color for the Reset button that appears on the button bar at the bottom of the form. The default value is #F0F0F0.

You can supply any CSS color name or RGB value recognized by the browser. For example, white, (255,255,255), #FFFFFF.

Forms Server recognizes any three decimal values separated with commas as an RGB value. For example, 255, 255, 255.

**Value**

CSS color name or an RGB value

**Examples**

```
<ConfigParams><ConfigParam name="resetColor" value="red"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&resetColor=red

## resetPosition

**Description**

Specifies the placement of the button in regard to the other buttons. The first button in a row is represented by the number 1, while 4 represents the last button in the row. The default value is 2.

**Value**

An integer between 1 and 4

**Examples**

```
<ConfigParams><ConfigParam name="resetPosition"
value="1"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&resetPosition=1

## resetText

**Description**

Specifies the text for the Reset button that appears on the button bar at the bottom of the form.

**Note:** If you remove all text from the button, an icon appears to represent the reset action.

**Value**

Any text

**Examples**

```
<ConfigParams><ConfigParam name="resetText"
value="Clear"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&resetText=Clear

## resetTextColor

**Description**

Specifies the color for the text on the Reset button that appears on the button bar at the bottom of the form. The default value is black.

You can supply any CSS color name or RGB value recognized by the browser. For example, white, (255,255,255),#FFFFFF.

Forms Server recognizes any three decimal values separated with commas as an RGB value. For example, 255, 255, 255.

**Value**

CSS color name or an RGB value

**Examples**

```
<ConfigParams><ConfigParam name="resetTextColor"
value="red"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&resetTextColor=red

## resetTooltip

**Description**

Specifies the text that appears when a user places the pointer over the Reset button. The default value is Reset Form.

**Value**

Any text

**Examples**

```
<ConfigParams><ConfigParam name="resetTooltip"
value="Clear"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&resetTooltip=Clear

## resetVisible

**Description**

Specifies whether the Reset button appears on the button bar at the bottom of the form. The default value is TRUE.

**Value**

TRUE or FALSE

**Examples**

```
<ConfigParams><ConfigParam name="resetVisible"
value="FALSE"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&resetVisible=FALSE

## saveActions

**Description**

Specifies the action the system performs when a user clicks the button. A button can perform multiple tasks. For example, the Save button can print and then save the form. The default value is save.

**Value**

Any combination of the following action names separated by a comma: save, reset, print, or attachments.

**Examples**

```
<ConfigParams><ConfigParam name="saveActions"
value="print,save"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&saveActions=print,save

## saveBorderColor

**Description**

Specifies the border color for the Save button that appears on the button bar at the bottom of the form. The default value is #C8C8C8.

You can supply any CSS color name or RGB value recognized by the browser. For example, white, (255,255,255),#FFFFFF.

Forms Server recognizes any three decimal values, separated with commas, as an RGB value. For example, 255, 255, 255.

**Value**

CSS color name or an RGB value

**Examples**

```
<ConfigParams><ConfigParam name="saveBorderColor"
value="red"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&asaveBorderColor=red

## saveColor

**Description**

Specifies the color for the Save button that appears on the button bar at the bottom of the form. The default value is #F0F0F0.

You can supply any CSS color name or RGB value recognized by the browser. For example, white, (255,255,255), #FFFFFF.

Forms Server recognizes any three decimal values, separated with commas, For example, 255, 255, 255.

**Value**

CSS color name or an RGB value

**Examples**

```
<ConfigParams><ConfigParam name="saveColor" value="red"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&saveColor=red

## savePosition

**Description**

Specifies the placement of the button in regard to the other buttons. The first button in a row is represented by the number 1, while 4 represents the last button in a row. The default value is 1.

**Value**

An integer between 1 and 4

**Examples**

```
<ConfigParams><ConfigParam name="savePosition" value="2"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&savePosition=2

## saveText

**Description**

Specifies the text for the Save button that appears on the button bar at the bottom of the form.

**Note:** If you remove all text from the button, an icon appears to represent the save action.

**Value**

Any text

**Examples**

```
<ConfigParams><ConfigParam name="saveText"
value="SaveForm"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&saveText=Save%20Form

## saveTextColor

**Description**

Specifies the color for the text on the Save button that appears on the button bar at the bottom of the form. The default value is black.

You can supply any CSS color name or RGB value recognized by the browser. For example, white, (255,255,255), #FFFFFF.

Forms Server recognizes any three decimal values, separated with commas, as an RGB value. For example, 255, 255, 255.

**Value**

CSS color name or an RGB value

**Examples**

```
<ConfigParams><ConfigParam name="saveTextColor"
value="red"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&saveTextColor=red

## saveTooltip

**Description**

Specifies the text that appears when a user places the pointer over the Save button. The default value is Save Form.

**Value**

Any text

**Examples**

```
<ConfigParams><ConfigParam name="saveTooltip" value="save and
submit"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&saveTooltip=save%20and%20submit

## saveTransformAsAttachment

**Description**

Determines whether the system sends a PDF as an attachment to Perceptive Content when a user performs a save action.

**Value**

TRUE or FALSE

**Examples**

```
<ConfigParams><ConfigParam name="saveTransformAsAttachment"
value="TRUE"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&saveTransformAsAttachment=TRUE

## saveVisible

**Description**

Specifies whehter the Save button appears on the button bar at the bottom of the form. The default value is TRUE.

**Value**

TRUE or FALSE

**Examples**

```
<ConfigParams><ConfigParam name="saveVisible"
value="FALSE"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&saveVisible=FALSE

## success

**Description**

Specifies the text message that displays when a form is successfully submitted.

**Value**

Any text

**Examples**

```
<ConfigParams><ConfigParam name="success"
value="Success!"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&sucess=Success!

### width

**Description**

Specifies the width of the web application within the parent web page. You can use either pixels or percentages for the value. For pixels, an example is width=250px. For percentage, an example is width=50%25. The %25 is the URL escape code for %.

**Note:** Scaling the application size larger can make the buttons inaccessible outside of the viewing area.

**Value**

Pixels or percentage

**Examples**

```
<ConfigParams><ConfigParam name="width" value="250px"/></ConfigParams>
```

http://localhost:8080/psw-content-forms-server/fs?form=FormName&width=250px

## Transform parameters

Adding the transform parameter to the imagenowforms.xml file gives you the ability to generate a PDF from the form data. You can choose to generate a PDF when a user performs a print action, a save action, or both.

You must add either the printTransform or saveTransformAsAttachment configuration parameter to add a transform parameter.

### transformProvider

**Description**

The reader sends the form data to an existing PDF reader template.

The xsl-fo leverages an existing template that specifies customized formatting and layout details.

**Note:** When you choose to use reader or xsl-fo providers, you are responsible for creating the templates that Perceptive Content leverages.

**Value**

reader or xsl-fo

## transformTemplate

**Description**

The file name of the template the system uses to generate formatting and style of the PDF file.

**Note:** The xsl-fo and reader transformProvider parameters require a template.

**Value**

The file name relative to the web application root