

# Process Director Documentation

## Implementer's Guide



Last Updated: 2024-04-11, 17:20

# Contents

---

<b>Contents</b> .....	<b>2</b>
<b>Documentation Formatting Note</b> .....	<b>8</b>
Text and Code Formatting Conventions .....	8
Icons .....	9
Other Conventions .....	9
<b>Quick Links</b> .....	<b>10</b>
<b>Getting Started With Implementation</b> .....	<b>12</b>
Intended Document Audience .....	12
Understanding Process Director .....	12
What is a process? .....	12
Parent and Child Objects .....	13
Content Types .....	15
Implementation Samples .....	16
[Sample Datasources] Folder .....	17
Case Management Folder .....	17
Charting Folder .....	17
Data Folder .....	17
Forms Folder .....	17
Knowledge Views Folder .....	18
Mobile Folder .....	19
PDF Folder .....	19
Process Folder .....	19
Scripting Folder .....	19
Tasks Folder .....	20
<b>Process Director User Interface</b> .....	<b>21</b>
Navigator UI .....	21
Legacy UI .....	21
Other Common UI Elements .....	21
Process Director Navigator User Interface .....	21
Overview .....	22
Marking Objects as Favorites # .....	35
Other Common UI Elements .....	37

	37
Navigation System # .....	37
Other Common UI Elements .....	38
Common UI Objects .....	38
Standard Icons # .....	41
Action Buttons # .....	41
Object Definition Interface # .....	42
Object Control Menu # .....	55
User Profile # .....	56
Icon Chooser .....	58
Condition Builder .....	62
Adding a Condition # .....	62
Using Arrays in Form Validation Conditions # .....	70
Client Validation # .....	72
Date Comparison Issues # .....	73
Debug Mode .....	74
Desktop Interface Workspace .....	76
<b>Process Director Content List Objects .....</b>	<b>81</b>
Business Rules .....	82
Business Rule Configuration .....	82
Custom Business Rule Parameters # .....	87
Global vs Local Business Rules .....	88
Business Rule Samples .....	89
Business Values .....	89
Related Topics .....	90
Creating Business Values .....	90
Business Value Operations .....	103
Case Management Overview .....	108
What is Case Management? .....	108
See also: .....	113
Case Definition .....	113
Implementing a Case Management Application .....	118
Charts .....	128
Configuring a Chart # .....	128
Properties Tab # .....	129

Dashboards .....	147
Properties Tab .....	148
Dashboard Definition Tab .....	150
Widget Properties .....	155
Launching Forms .....	156
Dashboard Widgets .....	156
Datasource Objects .....	174
Datasource Object Configuration # .....	175
Other Datasource Types .....	176
Datasource Caching .....	176
Common Datasource Types .....	178
Microsoft Excel Datasources .....	186
Windows File System Datasources .....	191
SharePoint Data Sources .....	195
Social Media Data Sources .....	203
DropDown Object .....	204
Table View Tab # .....	205
TextBox View Tab # .....	206
Text List Configuration # .....	206
Using Blank Values # .....	207
Associating the DropDown Object with a Dropdown Control # .....	208
Forms .....	209
Form Definition .....	210
Form Configuration .....	238
Online Form Designer .....	274
Button Tab .....	332
Button Properties Tab .....	333
Sort Tab .....	381
Button Properties Tab .....	382
Set Form Data .....	383
Conditions .....	384
Validation .....	385
Control Properties .....	386
Button Tab .....	387
Button Properties Tab .....	389



More Button Properties Tab .....	390
Form Definition Properties .....	391
Attach Tab .....	392
Attach Properties Tab .....	394
Email Templates .....	458
Goals .....	474
Configuring Goals .....	476
Knowledge Views .....	482
How Knowledge Views Work .....	484
Other Knowledge View Topics .....	488
More Information .....	508
Knowledge View Operations .....	508
Knowledge View Exporting and Reporting .....	522
Machine Learning (ML) Definition .....	528
Properties Tab # .....	529
Data Set Tab # .....	530
Transformation Tab # .....	533
Training Tab # .....	535
Visualize Tab # .....	542
Schedule Tab # .....	543
Meta Data .....	545
Power of Meta Data # .....	546
Meta Data Elements # .....	546
How Meta Data is Applied # .....	547
Meta Data Replication # .....	548
Importing / Exporting Categories & Attributes # .....	548
Creating Meta Data .....	549
Using Meta Data .....	554
Process Timelines .....	560
What is a Process Timeline Definition? .....	560
Processes and Subprocesses # .....	566
Background Operations # .....	566
Other Process Timeline Topics .....	567
Process Timeline Definitions .....	567
Process Timeline Activities .....	572

Conditions Tab # .....	587
Options Tab .....	588
Managing Process Timeline Instances .....	612
Reports .....	625
Creating a report # .....	625
Report Configuration # .....	626
Selecting Report Data # .....	630
The Report Designer # .....	632
Included Views # .....	633
Report Variables .....	636
Stream Actions Object .....	642
Properties Tab # .....	642
Data Set Tab # .....	643
Start Process Tab # .....	645
Schedule Tab # .....	647
Workflow .....	649
What is a Workflow Definition? .....	649
Creating Workflow Definitions .....	652
Workflow Step Task Types .....	664
Workflow Package .....	680
<b>Implementation Guidelines .....</b>	<b>691</b>
Best Practices for Implementations .....	691
Best Practices for Accessibility .....	699
Checking for Accessibility # .....	700
Accessibility Principles .....	700
Accessibility Principles .....	701
Managing Content .....	723
Content List # .....	724
Creating Content List Objects .....	726
Object Locking .....	733
Organizing Content List Objects .....	734
Importing/Exporting Content .....	748
Template Library .....	767
Importing Data .....	775
Direct URL Access to Objects .....	776

Working with SharePoint .....	778
Process Director Custom Utilities .....	796
Using the bpUtil Utility .....	797
Using the bpImport Utility .....	797
Using the bpEmailImport Utility .....	806
Using the Microsoft Windows Scheduler .....	814
Managing Users .....	823
User Profile # .....	824
External Users # .....	827
Permissions .....	829
Permissions Methodology on Production Systems .....	830
Setting Permissions .....	834
Permission Exceptions .....	841
Changing Existing Processes .....	841
Data Flow Analyzer .....	843
BP Logix Mobile Application .....	849
Mobile Application Users # .....	849
Importing a Form into the Mobile App Component # .....	850
Mobile Application # .....	856
Importing Form Instances Back into Process Director # .....	864
Importing Data Sources # .....	865
Applying a Data Source to a Screen # .....	868
<b>Configuring Azure for Process Director Integration .....</b>	<b>871</b>
Create a certificate to authenticate Process Director with Azure # .....	871
Add Process Director to Azure # .....	879
Conclusion .....	879
SharePoint Data Sources .....	880
Configuring a SharePoint OAuth (Tenant) Datasource # .....	880
Configure SharePoint Online permissions # .....	880
Configuring the SharePoint OAuth (Site) Datasource # .....	885
Conclusion .....	886
Sharepoint Legacy Datasource # .....	886
Other Datasource Types .....	888
Microsoft OAuth for SMTP .....	888
<b>Index .....</b>	<b>891</b>

## Documentation Formatting Note

### Text and Code Formatting Conventions

To highlight terms and concepts that have special relevance, this documentation implements several formatting conventions to make key words and terms more noticeable.

- **Control Label:** This format will identify the text labels or properties for Process Director objects, or the names of dialog boxes.  
**Example:** The **Name** text box.
- **UI Element:** This format will identify user interface elements such as buttons, tabs, or other UI objects used to perform interface operations.  
**Example:** The **Submit** button.
- **Formal Control Name:** This format will identify named Process Director controls.  
**Example:** A **Section End** control.
- **Process Director Object:** This format will identify named instances of Process Director Folders, Forms, Process Timelines, Knowledge Views, etc.  
**Example:** The **Travel Expense Approval Process Timeline**.
- **Key Terms:** This format will identify key terms and concepts introduced into the text of the document, and which are important to learn.  
**Example:** A **Case** is group of processes, transactions, or responses that define a complex activity.
- **Code:** This format will identify code samples, system variables, formulas, or other fixed programmatic syntax.  
**Example:** Type the following formula: **AirFare + Lodging**.
- **Code Option:** A section of a code sample to denote placeholder values that must be replaced by the user manually at design time.  
**Example:** {CURR\_USER, format=**FormatType**}
- **Code Comment:** A section of a code sample that is used for text comments, rather than runnable code.  
**Example:** // **This is a comment**.
- **Code Variable:** A programming object whose value is usually determined from a command written in code.  
**Example:** var **formControls** = BaseCurrentForm.FormControls;

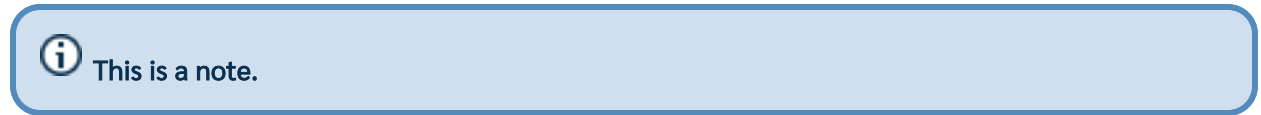
In addition to the above, extended samples of program code are presented in a special format to set them off from the rest of the text, as demonstrated below:

```
// Called after database initialized
public override void SetSystemVars(BPLogix.WorkflowDirector.SDK.bp bp)
{
    // Before we make SDK calls that access the database,
    // ensure DB has been opened
    if (bp.DBOpenComplete)
    {
        // Place custom code here
    }
}
```

Important text or warnings are presented in a special callout box for special attention:






Notes of general interest are also presented in special callout boxes:



Hopefully, the use of these formatting conventions will make it easier for you to determine the various types of objects to which the text refers.

## Icons

Some universal icons are used in the documentation. They are listed below:

ICON	NAME	DESCRIPTION
	Link	A hyperlink to the specific URL and named anchor of a topic, heading, or other item.
	Dropdown Closed	An icon that, when clicked, will expand dropdown text in a topic.
	Dropdown Open	An icon that, when clicked, will close the expanded dropdown text in a topic.

Finally, some topic headers within each online document may display a link symbol (#) when you mouse over the header. Clicking the link will navigate to that specific section of the document, which can then be bookmarked in your browser.

## Other Conventions

URLs displayed in sample will, unless used for commands or URLs used on the local host machine, use the "HTTPS" prefix by default, as modern practice has evolved to use the encryption layer to access URLs, instead of the plain-text method (HTTP) of accessing URLs.

## Quick Links

This page provides an alternate table of contents to assist you with finding links to topics that are searched or used most often. We hope this will help you find specific topics a bit faster than using the Search or normal Table of Contents features.

### General Topics

[Adding objects to the content list.](#)

[Importing and Exporting Applications](#)

[Permissions Methodology](#)

- [Setting Permissions](#)

[Security For Process Director](#)

[Best Practices For Implementation](#)

### Forms

[Editing a form](#)

[Adding Form Controls](#)

- [Basic Controls](#)
- [Additional controls](#)
- [Setting datetime control intervals](#)

[Configuring a form definition](#)

- [Form Properties](#)
- [Setting form data](#)
- [Filling dropdowns](#)
- [Adding validation rules](#)

### Process Timelines

[Editing a Process Timeline's properties](#)

[Timeline Activities](#)

[Timeline Activity types:](#)

- [User](#)
- [Custom Task](#)
- [Parent and looping](#)
- [Wait](#)
- [Notify](#)
- [Case](#)

## System Variables

[Adding system variables](#)

[Special system variable configurations](#)

[User sysvars](#)

[Form Sysvars](#)

[Timeline sysvars](#)

[Timeline Activity sysvars](#)

## Email Templates

[Creating an email template](#)

[Task completion via email](#)

[Multiple-message email templates](#)

## Business Rules

[Creating a Business Rule](#)

## Business Values

[Creating a SQL Datasource](#)

[Creating a SQL Business Value](#)

## Knowledge Views

[Configuring a Knowledge View](#)

[Associate with a specific object](#)

[Objects to return](#)

[Setting columns](#)

[Filtering the data](#)

[Exporting Knowledge Views](#)

## Goals

[Configuring](#)

[Run on a scheduled basis](#)

## Reports

[Creating a report](#)

## Developer's References

[Custom Variables](#)

## Getting Started With Implementation

Process Director provides business users with a web-based software solution with the advanced no-code/low-code capabilities they need to provide greater agility and visibility into their business processes, and create applications quickly. Built on an integrated document management and process automation system, Process Director manages, automates and reports on an organization's business processes and provides powerful storage, categorization and search technologies for all documents, electronic forms (Forms), and processes.

This document is the Implementer's Reference Guide for Process Director. To install the product, refer to the Process Director Installation Guide. For an overview of all the documents available, please refer to the [BP Logix documentation web site](#).

This topic provides a quick overview of the product terminology, how to get started, and steps to first login. Subsequent topics will get you started on your way to understanding how Process Director features and components can be utilized in your organization.

## Intended Document Audience

This Implementer's Reference Guide describes the advanced features and customization options available for Process Director and contains the information necessary to build your business processes in Process Director. It is intended as a reference for the users that will be implementing Process Director. The implementers typically are defined as those users that are familiar with the existing business processes inside your company and are familiar with the basic concepts of Process Director, such as Forms, processes and Knowledge Views.

This document doesn't address the custom scripting or external APIs that are available for Process Director. Although customization options for Process Director aren't required, they are provided to enable the product to perform unique business logic and to integrate into existing systems. Refer to the Process Director Developers Guide for information on the customization options.


## Understanding Process Director

Process Director can be used for a variety of functions. It is a content management system with integrated process modeling, document management, Forms processing, document imaging, knowledge management, business intelligence and reporting. Throughout all processes, users can be assured of secure access to shared documents, digital content and all data stored in the Process Director database.

## What is a process?

A process is a set of tasks, notifications, or activities that are completed in a specific order. In Process Director, a process is modeled primarily using a Process Timeline. So, the term "process" in the context of Process Director, is really a just a generic term for a Process Timeline.



 A legacy process model, the Workflow, is also incorporated into the product. BP Logix does not recommend the use of this process model. It remains in the product solely for backwards compatibility.

Process Timelines can implement the Run Process task to initiate a second process. The second process will run parallel with the process that starts it. The process that implements the Run Process task is called the parent process. Similarly, the process that is started by the Run Process task is called the child process.

Any number of child processes can be started by a parent process, and synchronization allows a parent process to wait for the completion of a child process before proceeding. When a process is started, it can optionally contain a copy of all the parent process' references (e.g. document, Forms, etc.) or just a reference to them.

So, let's discuss a scenario where a parent/child process would be necessary, and where the Run Process task might be useful.

Let's say that there is a specific process for obtaining approval from the organization's financial analysts, such as a requirement to obtain multiple approvals in a specific order. Then, let's assume that all financial approval processes, whether requests for travel, or capital spending, or expense reimbursement, must be approved by the financial analysts if they exceed a certain amount.

One way to implement this would be to encapsulate Financial Analyst approval into its own process by creating a Process Timeline to model it.

Once you've done so, you can create your travel, or expenditure request processes separately. In each of these parent processes, you can use the Run Process task to call the financial analyst approval process as a child process if the amount exceeds the threshold requirement.

By creating processes in this modular fashion, you can re-use a process many times as a child of other processes, rather than repetitively modeling it multiple times in different processes.

## Parent and Child Objects

There are a number of places in the documentation where the terms "parent" and "child" will be used in various contexts. In general, these terms refer to a hierarchical system of organization where the **parent** object is the higher-level object in the hierarchy, and any object that is lower in the hierarchy is called the **child**. In Process Director the terms "parent" and "child" are used in the following contexts.

**Content List:** In the [Content List](#), the highest-level object is the Partition. The Partition is the parent object, and all objects contained in the Partition are the child objects of the Partition. Most commonly, the child objects at the root level of the Partition are folders. Each folder is also the parent object of its contents, e.g., a subfolder would be the child object of a root-level parent folder. Every folder in the Content List, in turn, is both a child of its parent folder (or the Partition), and is, at the same time, the parent of its folder contents.

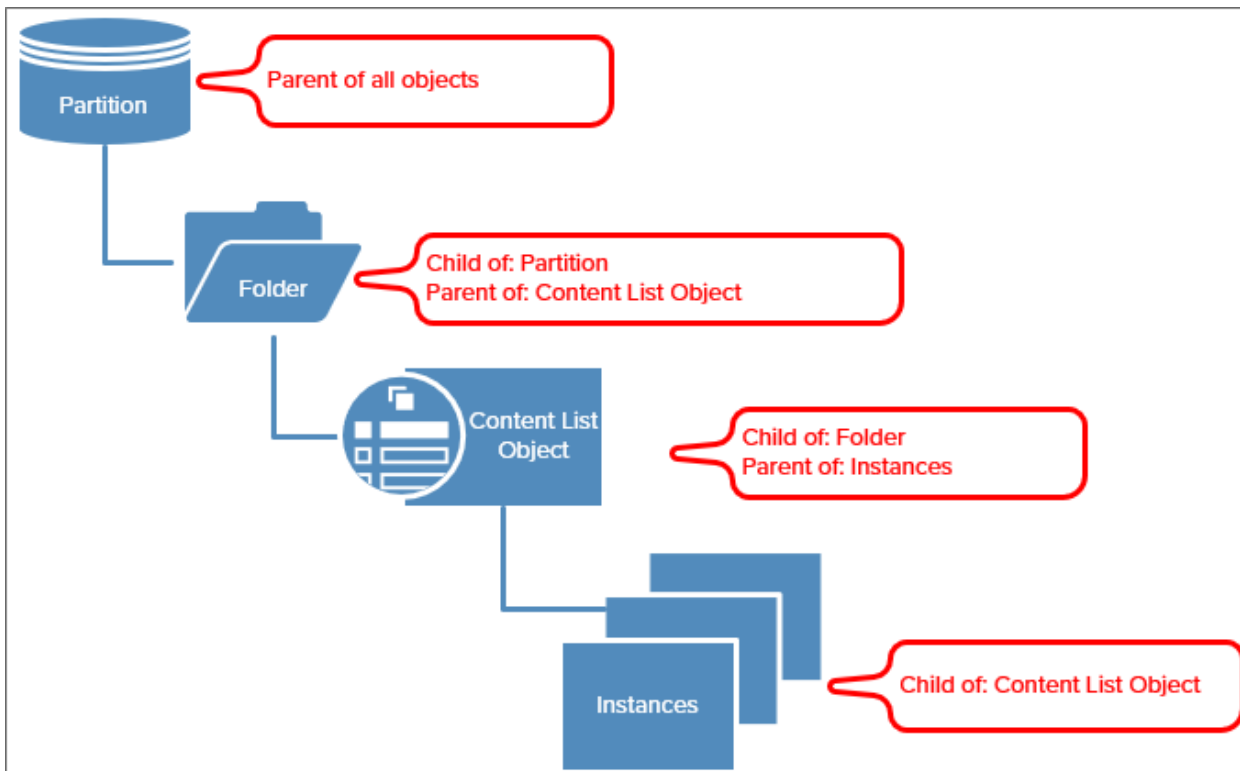
**Object Definitions and Instances:** Each object definition is the parent of its object instances. For example, every time a Form is submitted, a Form instance is created. All of these Form instances are considered children of the Form definition, which is the parent.

**Instances and Attachments:** Object instances are the parent object for their child attachments. If you attach a Word document as an attachment to a Form instance, the Form instance is the parent, and the Word document attached to it is the child.

**Controls:** A number of Form controls, such as the **Section** control, are containers for other controls. In this context, the **Section** control is the parent control, and all controls contained in the **Section** are child controls. Indeed, the Form itself is a container for all of the controls, so the Form is the parent object of the **Section** control, which is the Form's child.

















**Process Timelines (Parent Activities):** The Process Timeline has an Activity Type called the Parent Activity. The Parent Activity is commonly used to create iterating loops in a Process Timeline, but is also used as a container for a set of closely related activities. All of the Process Timeline activities that are subordinate to the Parent Activity are child activities. A Parent Activity can also contain subordinate Parent activities, which are children of the top-level Parent Activity, but are also parents of their subordinate activities.






**Process Timelines (Dependencies):** The Process Timeline organizes activities by dependency, creating finish-to-start dependencies between activities in order to create a critical path dependency tree. We generally refer to higher level objects in the dependency tree as parents, while lower-level objects are referred to as children.




## Content Types

Process Director supports various content types in the database. The term “object” will be used generically throughout this document to refer to one or more of the content types defined below. Each object icon shown is the default, but can be changed during implementation if desired.

ICON	CONTENT TYPE	DESCRIPTION
	Business Rule	Business rules can be defined as simple conditions or sophisticated interrelated rule sets.
	Business Value	A Data Virtualization object that returns a value from an external database, REST Service, or Knowledge View.
	Case Definition	The central object of a Case Application that stores all shared case data and identifies the Dashboard used to display the Case folder.
	Chart	A chart object graphically displays data.
	Dashboard	A user-defined interface object that can display any desired Process Director object in a custom portlet.
	Data Source	A re-usable Datasource that can connect to an external database.
	Document	Any document or file type that has been uploaded to the server. If the file type is known, the icon displayed will appear as it would in Windows Explorer. Examples:  Word Document  Excel File  PDF File  Image file
	Dropdown Object	Content for a dropdown on a Form.
	Form	Electronic form (Form) that can be filled out and submitted online. Completed forms are also stored on the server under the Form definitions with the  icon.
	Folder	Objects on the server, including sub-folders.
	Goal	An object that performs a scheduled evaluation of conditions and sets a global system status and/or automatically starts a process based on the result of the scheduled evaluation.

ICON	CONTENT TYPE	DESCRIPTION
	Knowledge Views	Query window into the database that is able to display objects based on filter criteria.
	Process Timeline	A Process Timeline Definition is a predefined route for the automation of a process in your environment. Running and completed Timelines are also stored on the server under the Timeline definition.
	Report	An report object created with the Advanced Reporting component.
	Script	Custom script for a Form or Process.
	Workflow	A Workflow definition is a predefined route for the automation of a process in your environment. Running and completed Workflows are also stored on the server under the Workflow definition.

 BP Logix recommends the use of the Process Timeline due to its superior capabilities. The Workflow object is a process modeling method that predates the Process Timeline and offers fairly robust functionality. While the Workflow is fully supported in its current state, any development of new process functionality has been limited to the Process Timeline since Process Director v4.5.

The icons displayed above are the standard icons available in Process Director. For each object that you create, the object definition has an **Icon** property, with which you can change the icon for your objects to another one of the many standard icons that come with Process Director, or you can [create your own custom icons](#). When you create custom icons, some Administrative intervention is also required, as outlined in the System Administrator's Guide.

## Implementation Samples

BP Logix Provides a number of implementation samples that illustrate how to perform a variety of tasks in Process Director. The XML file containing the most recent samples collection can be found on the Downloads page of the [BP Logix Support web site](#). Simply download this XML file, then import it to the root folder of the desired Partition of your **Content List** in Process Director to create a new [Samples] folder.

All of the currently available samples are described below. The [Samples] folder organizes the sample files in a directory structure.

## [Sample Datasources] Folder

This folder contains the Datasource objects that are used by all of the sample projects included in the Samples package.

## Case Management Folder

### Case Management

A sample auto insurance claims sample that illustrates the Case Management functionality of Process Director.

## Charting Folder

### *Charting*

A sample chart that retrieves data from an external Datasource, and displays it as a Bar chart.

## Data Folder

The samples in this folder demonstrate how to use data from external sources in process Director.

### *Excel Import*

This sample demonstrates how to import data from a Microsoft Excel spreadsheet for use in Process Director. The Excel data is imported into the Process Director internal database, and is automatically re-imported every time the Excel file is checked in after editing.

### *REST Web Services*

This sample demonstrates how to use a REST web service. In this sample, the REST service takes a Zip Code or Postal Code and returns the correct city, state/province, and country, which it uses to populate the appropriate Form Fields.

## Forms Folder

This folder contains samples that demonstrate various Form operations.

### *Calculations*

This sample demonstrates the use of calculations by inserting a control or system variable to calculate a specified formula. These calculations can be used inside an array to calculate rows and sum columns, as well as outside an array to calculate form field values.

### *Copy Array*

This sample copies the values in an array from an original form to a destination form.

### *DB Connectors*

This sample demonstrates two methods of filling dropdowns and fields with data from an external database. The first—and preferred—method is to use Business Values. The second method is to use the Fill From DB Custom Tasks.

### *Form Element Styles*

An example of how to conditionally apply built-in styles to form elements.

### ***Email Template***

This sample demonstrates custom email templates that can be used for process notifications. This template can be used to customize email notifications.

### ***Fill 2 Dropdowns in an Array from External Database***

The sample demonstrates how to fill dropdowns within an array from data in an external database.

### ***Fill Selected Items***

This sample demonstrates how to populate a List box from a Business Value, then mark ListBox items as selected (checked) based on a second Business Value.

### ***Form Attachments Handling***

This sample demonstrates how to attach documents, Forms, or copy buffer items (items from your clipboard). Attachments can be displayed on the form or as a downloadable link.

### ***Form Conditions***

This sample contains Forms conditions that allow you to Show/Hide elements on a form and to Enable/Disable form fields.

### ***Form Tabs***

This sample shows the use of tabs in a Form user interface.

### ***Repeating Lists (Arrays)***

The Form builder supports the creation of repeatable and dynamic input fields known as an array, typically used for a list of items. An “array” tag surrounds a group of information or input fields on the Form, allowing for the dynamic display of fields when the Form is displayed to the user. The number of rows in an array can be increased or decreased as needed using an array control.

### ***Show Form Sections Conditionally with Business Rules***

This sample demonstrate how to show or hide sections or embedded sections, based on the result of a Business Rule.

### ***Validation***

A sample implementation of the form field validation logic used in Process Director.

### ***Word Export***

This sample demonstrates how to export data from Form fields into a Word template.

## **Knowledge Views Folder**

This folder contains samples that demonstrate the use of Knowledge Views.

### ***Email KView Results as CSV***

This sample shows how to export the contents of a Knowledge View to a CSV file, attach the exported CSV file to a template-based email, then send the email as a notification.

### ***Knowledge View Process Report Samples***

This sample demonstrates how a Knowledge View can be used to report on Process Timeline activities.

### ***View Knowledge View in Form***

This sample demonstrates how to display a Knowledge View in a Form.

## **Mobile Folder**

This folder contains samples that demonstrate how Process Director can be used with mobile devices.

### ***Mobile Device Sample***

This sample shows how to capture a picture and geo-coordinates from a mobile device, and display the location on Google maps.

## **PDF Folder**

### ***Add an Image to a PDF***

This sample takes an image generated through the Signature control and places it into a PDF document.

### ***Export PDF***

This sample demonstrates the use of a Custom Task to export Form fields, attachments and the Form itself to generate to a PDF. This sample will allow you to attach a document to the Form and export it as a PDF in the process, as well as displaying it on the form.

### ***Using PDF Templates***

This sample uses two kinds of PDF generation: it will create a PDF of the Form, or insert Form information into an existing PDF template.

## **Process Folder**

This folder contains samples that demonstrates various process actions.

### ***Timeline Conditions***

The sample demonstrates conditional routing, enabling you to route a process based on a specific condition defined in the Timeline.

## **Scripting Folder**

This folder contains samples of custom scripts.

### ***Attachment Restrictions***

This sample demonstrates a script that looks at the file attachments in a Form, and rejects the file if the file isn't a PDF file, or the file name doesn't meet some naming criteria.

### ***Sample Form Scripting***

This sample will use various APIs from Process Director that demonstrate how to get and set a value of a form field, and the use of the current user API.

### ***Sample Knowledge View Script***

This sample uses a Knowledge View script to query and change the value of a column in the data returned in this Knowledge View.

### ***Using a script in a Form***

A sample designed to instruct the user on how to insert a simple script in a form.

### *Using JavaScript in a Form*

A sample showing how client side JavaScript can be used to format data in form fields (e.g. uppercase text box input).

## Tasks Folder

This folder contains samples of various task assignment methods.

### *Assign a Task From a Business Rule*

This sample demonstrates how to assign a task to a user from a Business Rule.

### *Delegation from Date Range*

This sample demonstrates how to delegate your tasks to a user during a specific date range.

### *Dynamic User Assignment from Form*

This sample demonstrates how to dynamically assign one or more users to a process step. The Form will contain a user picker for the initiator to select a user to be assigned via the process step.

### *Serial Assignment*

You can assign multiple persons to complete a single step in a process. For example, this ability may be useful to run through an approval process in a single step, rather than creating multiple activities in a Process Timeline. This is especially true if the number of people involved in the process can vary.



## Process Director User Interface

For Process Director v6.0.300, BP Logix has implemented a new User Interface for the product. Indeed, UI changes to make the product more easily usable, and with a modern look and feel, is one of the primary drivers of change for Process Director. We recognize that these changes may mean that users of earlier versions of Process Director will see a radically different interface, which uses different interface conventions to perform common tasks. Since that is so, we've updated the documentation to include both the new and legacy interfaces for the product.

We recognize that this increased rate of change to the product's look and feel may cause some confusion, as different organizations may be on versions of Process Director that incorporate different UI conventions. At BP Logix, we'll help alleviate this by reorganizing the documentation as necessary. For some of the product changes, we'll add entirely new topics to the documentation. For other changes, we'll insert the Navigator UI documentation into existing topics. There will, however, be some cases in which the look and feel

Previously, the documentation had a single topic that covered the Main UI and navigation scheme, Content List, and other common UI elements. The release of the Navigator UI for Process Director v3.0.600 necessitates a change to the documentation's organization. Various elements of both the old and new UI have been split into three different topics, as described below.

### Navigator UI

For users of Process Director v6.0.300 and higher, you can refer to the [Process Director Navigator User Interface topic](#), which covers the new UI, product Home Page, navigation scheme, and user profile options in the Navigator UI.


### Legacy UI

For users of Product Director versions **prior to** v6.0.300, the topic that covers the main user interface is covered in the [Process Director User Interface \(Legacy\) topic](#), which covers the use of the tabbed interface and Workspace navigation buttons.

### Other Common UI Elements

Please refer to the [Common UI Elements topic](#) to learn about other common user interface conventions.

## Process Director Navigator User Interface

 This topic refers to an upcoming release of Process Director. The release is currently in Beta with some selected customers, and is subject to change without notice.

**i** If you are using a version prior to v6.0.300, you can view the user interface documentation for your version of the product on the [User Interface \(Legacy\)](#) topic.

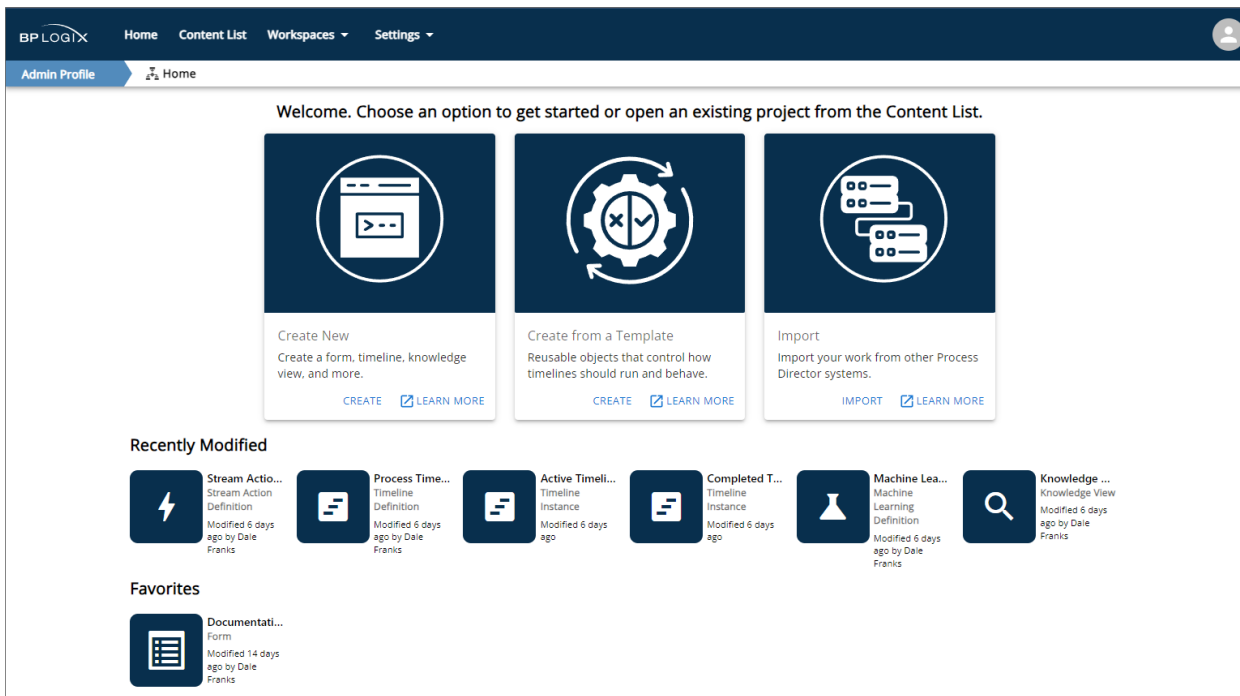
Process Director's user interface for v6.0.300 is very different from previous versions, and represents a fundamental change for administrators and process designers. At BP Logix, we've christened this new user interface "Navigator".

## Overview

The Navigator UI for Process Director consists of a [Home](#) page that opens by default, as well as significant changes to the UI features and navigation scheme.

## Home Page #

By default, a [Home](#) page is displayed that provides access to many features that enables you to begin building Process Director objects quickly. Unlike previous versions of Process Director, the [Content List](#) does not serve as the main [Home](#) page for administrators and designers. It is, instead, accessed via a menu item, as described below. Keep in mind that the [Home](#) page you see on your installation may be slightly different, and may not display all of the elements we discuss in this section of the topic.



At the top of the [Home](#) page, large panels provide **Create** links to enable you to quickly:

1. Create a new Process Director object, such as a Form or Process Timeline,
2. Create an application from an existing template contained in the [\[Template Library\]](#) folder of your installation, or
3. Import a Process Director PDZ file into the installation.

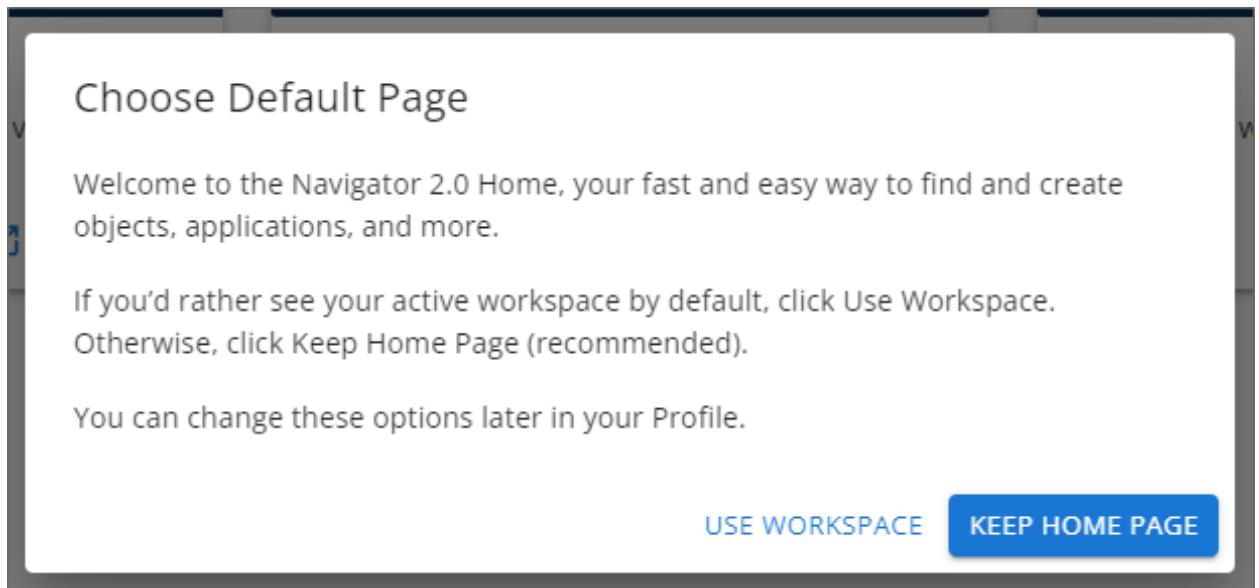
Additional [Learn More](#) links in each panel will, when clicked, open the appropriate documentation page for [creating objects](#), [creating a new template-based application](#), or [importing objects](#), in a new browser window.

Below the large panels, a series of icons in the **Recently Modified** section are displayed that link to the most recently-configured objects in the system. These links will immediately navigate you to these recent items and open them for configuration, without having to navigate through the Content List to find them.

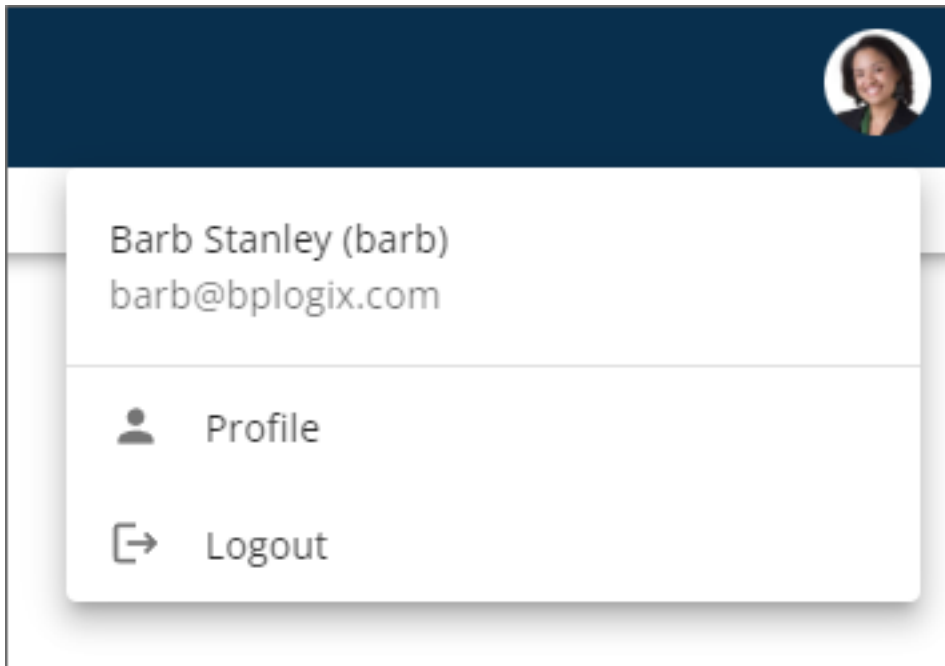
Finally, a **Favorites** section displays a series of icons that link to objects that you've marked as a favorite in the [Content List](#). This section will only appear if you have marked one or more [Content List](#) items as a favorite. Please see the [Marking Objects as Favorites section](#) below for more information.

### Home Page Display Selection

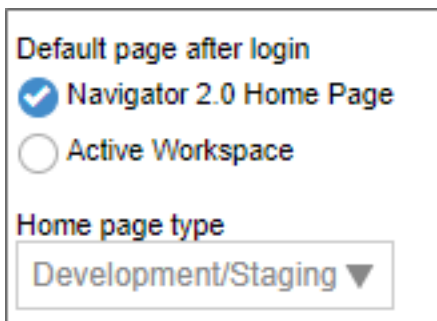
The very first time you log into Process Director as an Administrator, you will be presented with the selection dialog that enables you to choose whether to view the [Home](#) page as the default, or to view the [Content List](#), just as you did in previous versions of the product.



You can click the appropriate button to either [Keep Home Page](#), which is the default, or [Use Workspace](#), which is the legacy behavior. The choice you make here will be saved to your user Profile, which is accessible from the user icon in the top right corner of the screen.



In the user profile, Administrators will see some [Home](#) page settings that they can configure.



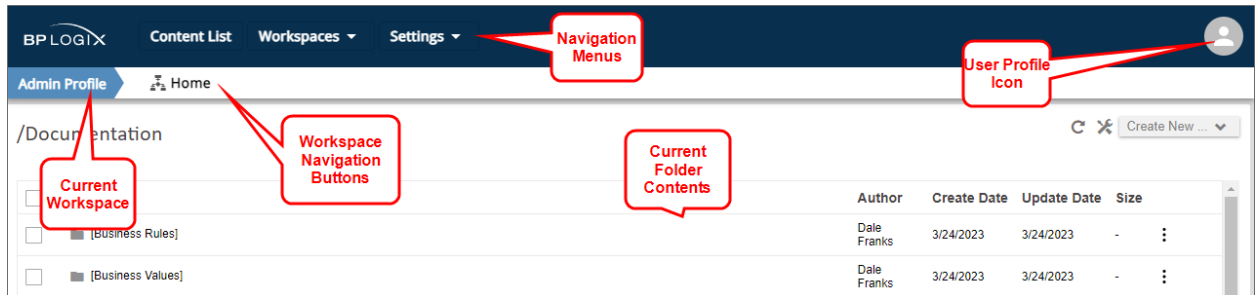
The [Default page after login](#) property will enable you to switch between the [Home](#) page and Workspace as the default display object when you log into the system. In addition, the [Home page type](#) property enables you to alter the visual appearance of the Home page, based on the type of system you're accessing. This setting controls the appearance of the large activity panels at the top of the [Home](#) page. The available options are:

- **Development/Staging:** This setting will show all three of the large panels at the top of the [Home](#) page for creating and importing objects.
- **Production:** This setting will show only the panel for importing objects, since, on a Production system, you should not be creating new objects or applications directly on the Production server, but you should be able to import applications from Dev/Staging.

The [Home page type](#) property is not accessible if the [Default page after login](#) property is set to *Active Workspace*.

## Navigation Scheme #

Unlike the tabbed user interface for previous version of Process Director, v6.0.300 introduces a menu-based navigation scheme to provide quicker and easier access to the various features available in the product. The new menus are arrayed across the top of the page and are only visible to users with the appropriate permissions. To provide the fullest overview of the available menu items, we'll look at what a fully authorized system administrator sees.



At the top of the page, a navigation bar enables you to access the functions available for your user role/permissions. The navigation menus on the left side of the page enable you to view the **Content List**, navigate to different **Workspaces**, or to access the IT Admin **Settings** for Process Director. On the right side of the navigation bar, the **User Profile** icon provides access to your user profile page and, if applicable, other functions.

Immediately below the navigation bar, a Workspace bar shows the name of the currently displayed Workspace on the left side of the page. In the example shown above, we're viewing the **Admin Profile** Workspace. Immediately to the right of the Workspace name, any navigation buttons used in the Workspace will be arrayed from left to right. In this example, we have a single navigation button, which, when clicked, will take you to the Workspace **Home** page.

**!** Workspace navigation bars that are configured to show built-in items such as IT Admin, Logout, etc., that are part of the main menu navigation, will no longer display those specific items in the Workspace navigation bar, since they now appear in the main product UI. This change enables you to use the available space for custom navigation items in the Workspace navigation bar.

Below the Workspace bar, the contents of the current folder are displayed. Clicking on any item in the contents pane will navigate to that item by opening it.

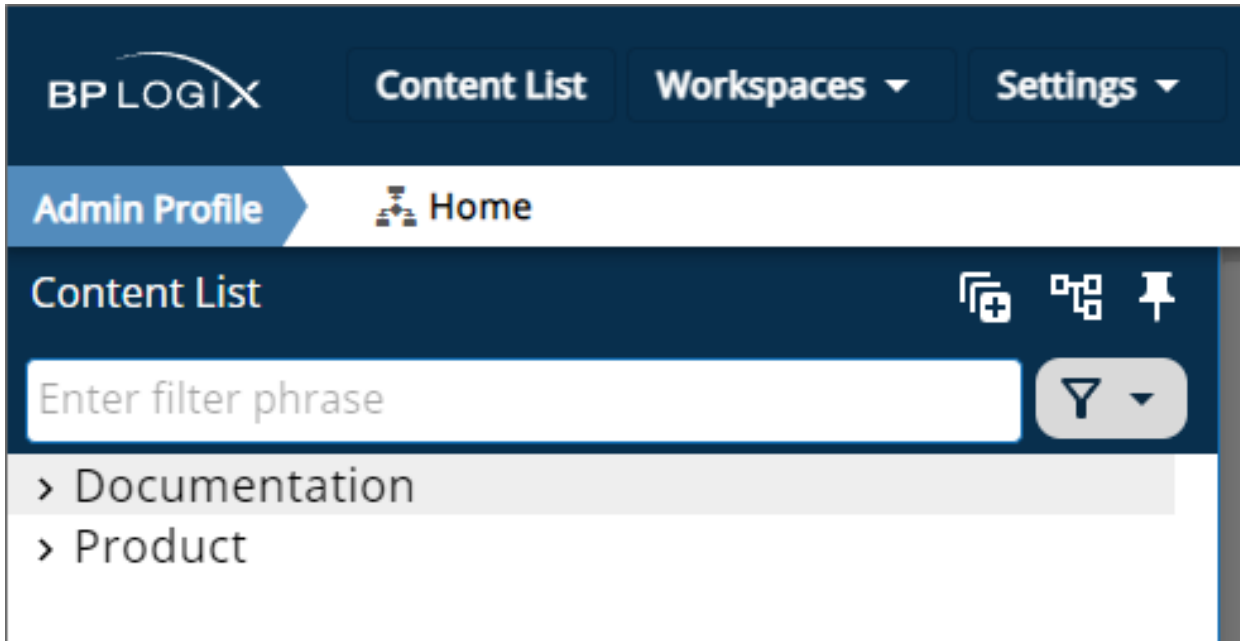
Lets take a closer look at each item in this interface.


## Navigation Menus #

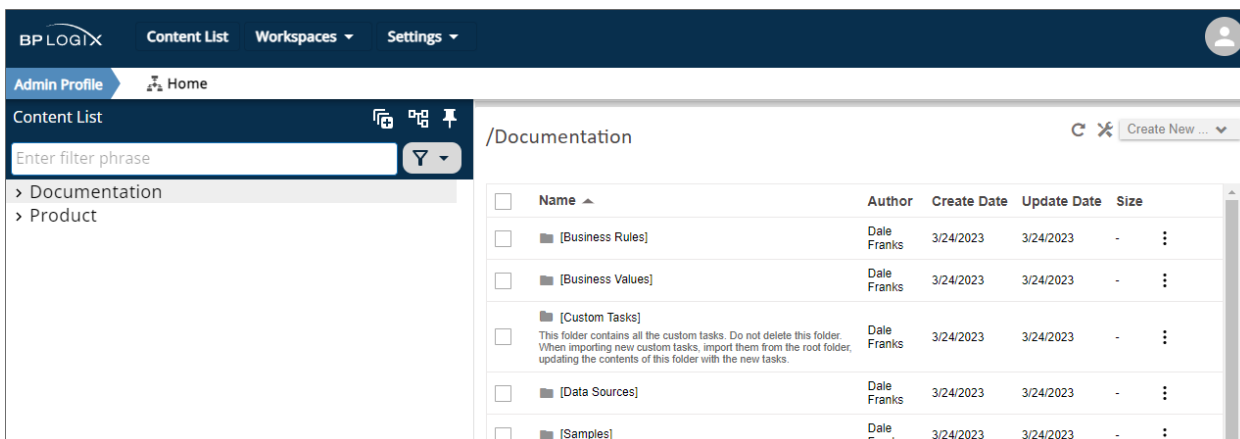
There are four possible navigation menus displayed in the navigation bar, **Content List**, **Workspaces**, **Settings**, and **User Profile**. System Administrators will have access to all of these menus. Normal end users will see only the **Workspaces** menu, if they are assigned to more than one workspace, and the **User Profile** menu.

## Content List Menu

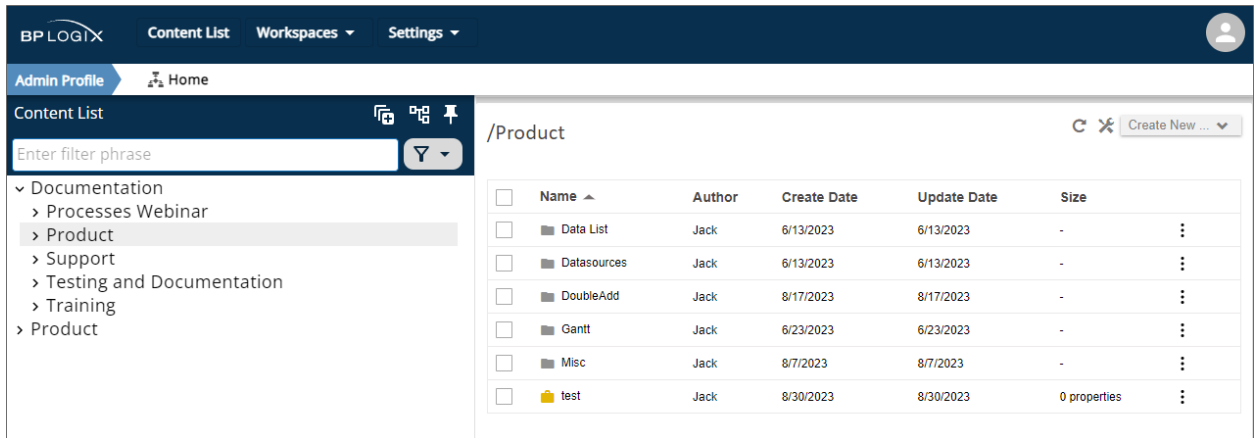
The **Content List** menu is a single, clickable menu item that opens the folder navigation tree for the full Content List.




By default, the **Content List** is dismissible, which is to say that if you click outside of it, the tree will disappear. A **Pin** icon (  ) will when clicked, make the Content List persistent, keep it permanently displayed. While the tree is dismissible, the folder contents pane of the "Content List" will be grayed out. Clicking on the grayed out area, or other area of the page, will dismiss the tree. But, once you click the **Pin** icon and make the tree persistent, the current contents pane will be accessible on the right side of the page, and the tree will be accessible on the left, similar to the UI convention that was used in previous versions of the product.

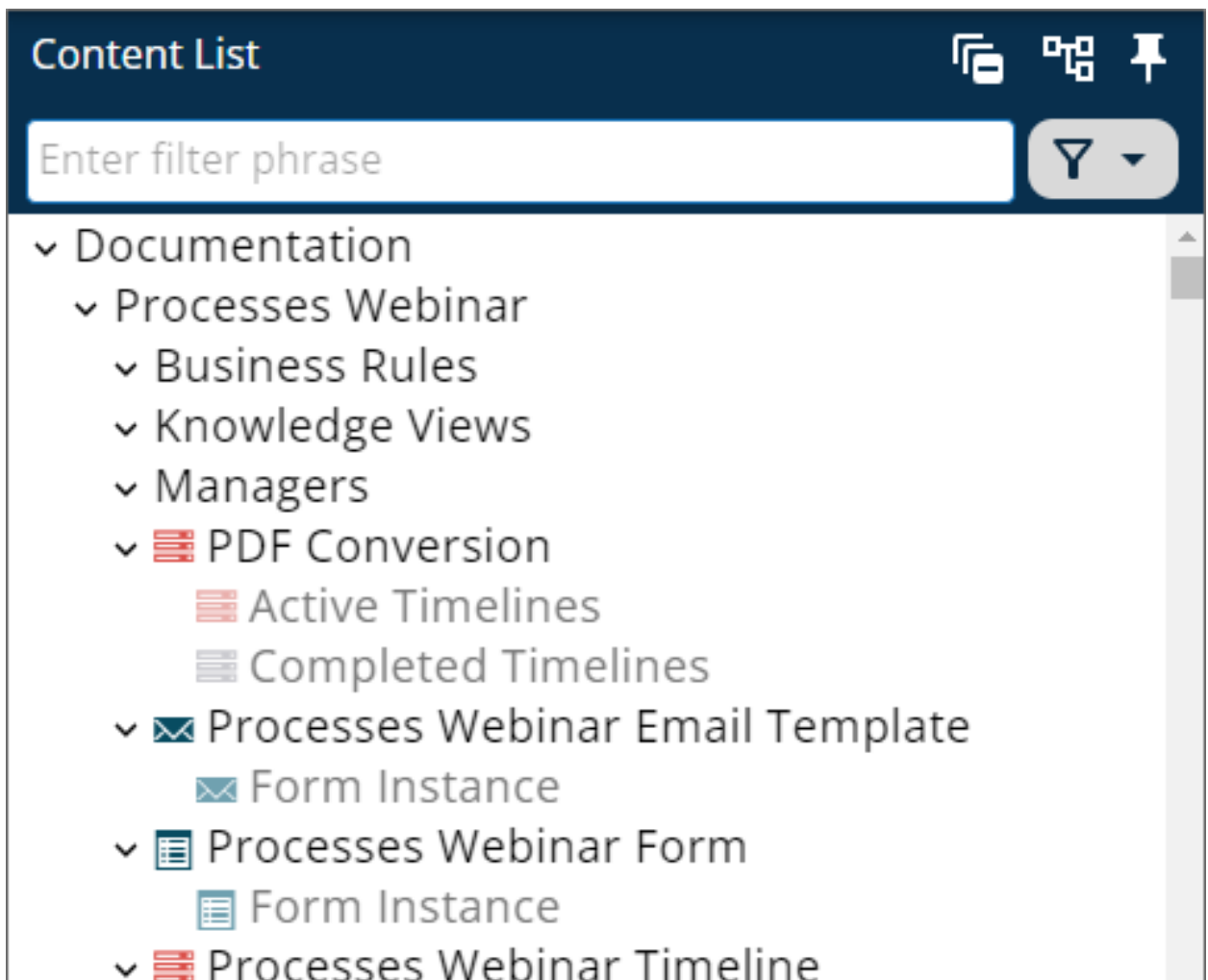



You can navigate through the content using either the folder tree pane on the left, or the contents pane on the right. As you do, both panes will update to display the current content you're viewing.




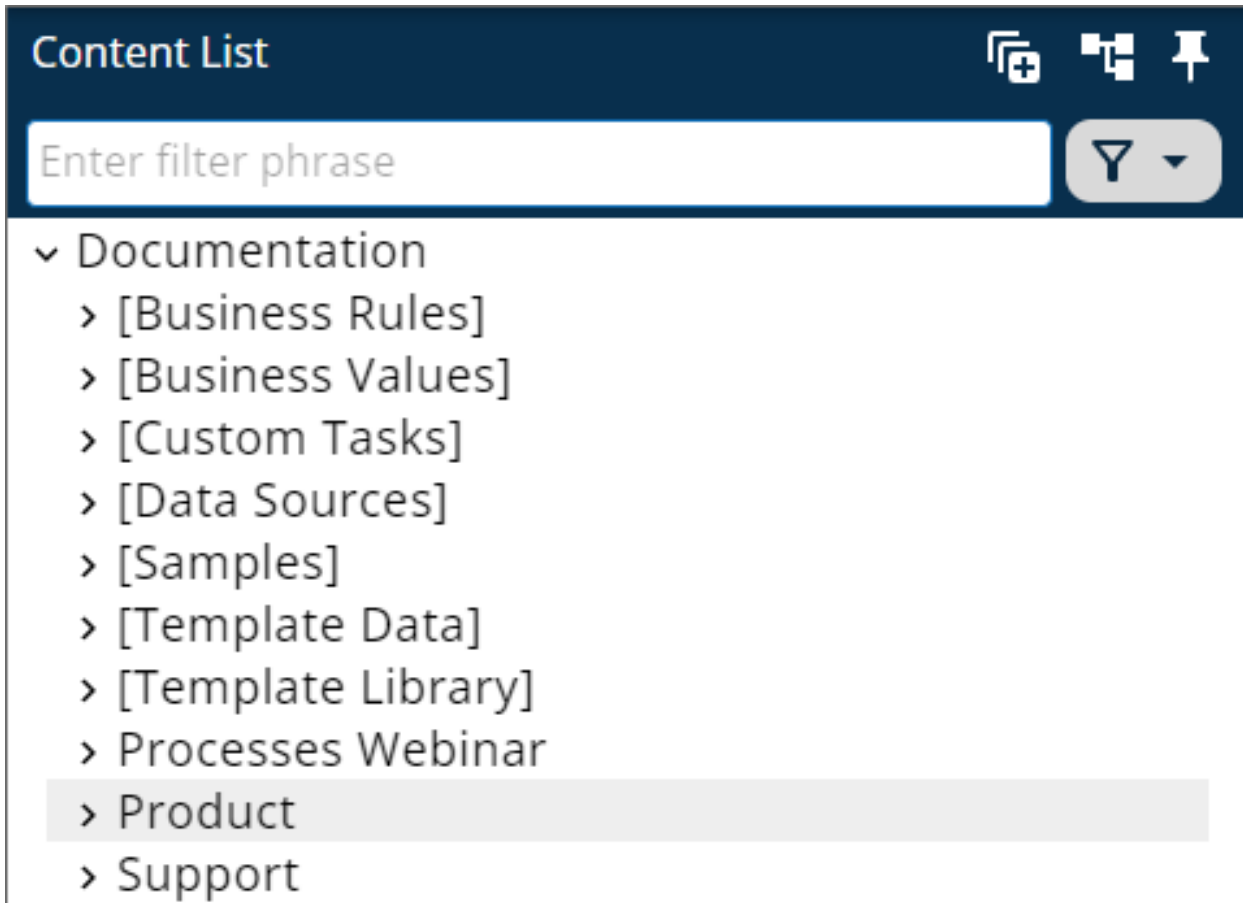
The treeview pane offers several features to make navigating through the [Content List](#) more convenient.


The **Expand All** icon () will, when clicked, open the entire structure of the folder tree.



This same icon will, while the tree is expanded, automatically change to a **Collapse All** icon (  ). Clicking the icon will collapse the tree view to its original state.

By default, system folders such as [\[Custom Tasks\]](#), [\[Business Values\]](#), [\[Data Sources\]](#), or any other folder whose name is placed in square brackets, will be hidden from view. The **Show System Folders** icon (  ) will, when clicked, display these folders in the tree.



Again, this icon automatically changes to the **Hide System Folders** icon (  ) while the system folders are displayed, and clicking it will hide the folders again.

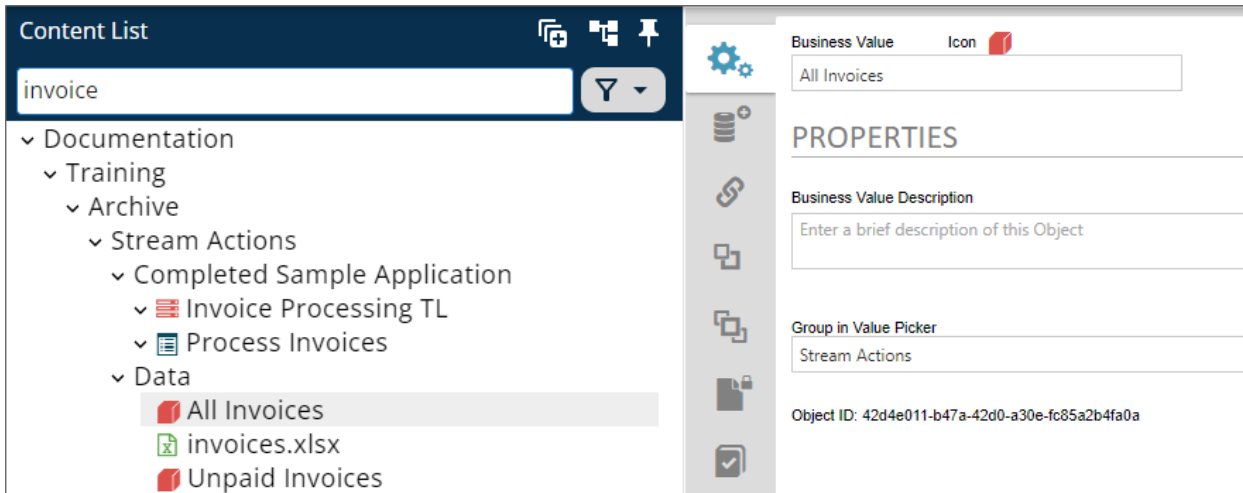
A search bar that's labeled "Enter filter phrase" enables you to enter a search term. When you enter your desired term, any [Content List](#) items whose names match the search term you enter will be displayed in the tree.



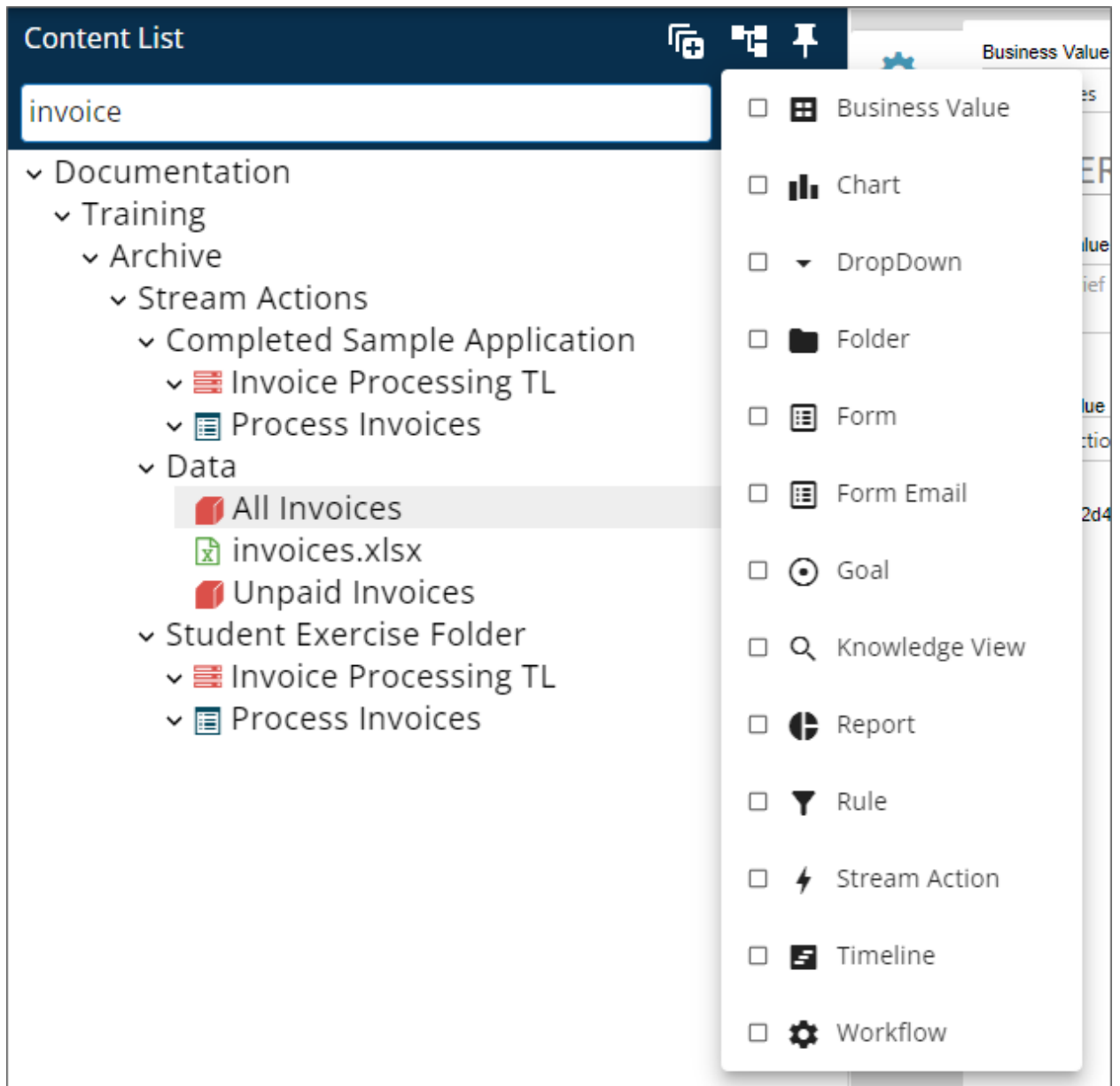
The screenshot shows a 'Content List' interface with a search bar containing the text 'invoice'. Below the search bar is a tree view of content items. The tree structure is as follows:

- Content List
  - Documentation
    - Training
      - Archive
        - Stream Actions
          - Completed Sample Application
            - Invoice Processing TL (with icon)
            - Process Invoices (with icon)
          - Data
            - All Invoices (with icon)
            - invoices.xlsx (with icon)
            - Unpaid Invoices (with icon)
          - Student Exercise Folder
            - Invoice Processing TL (with icon)
            - Process Invoices (with icon)

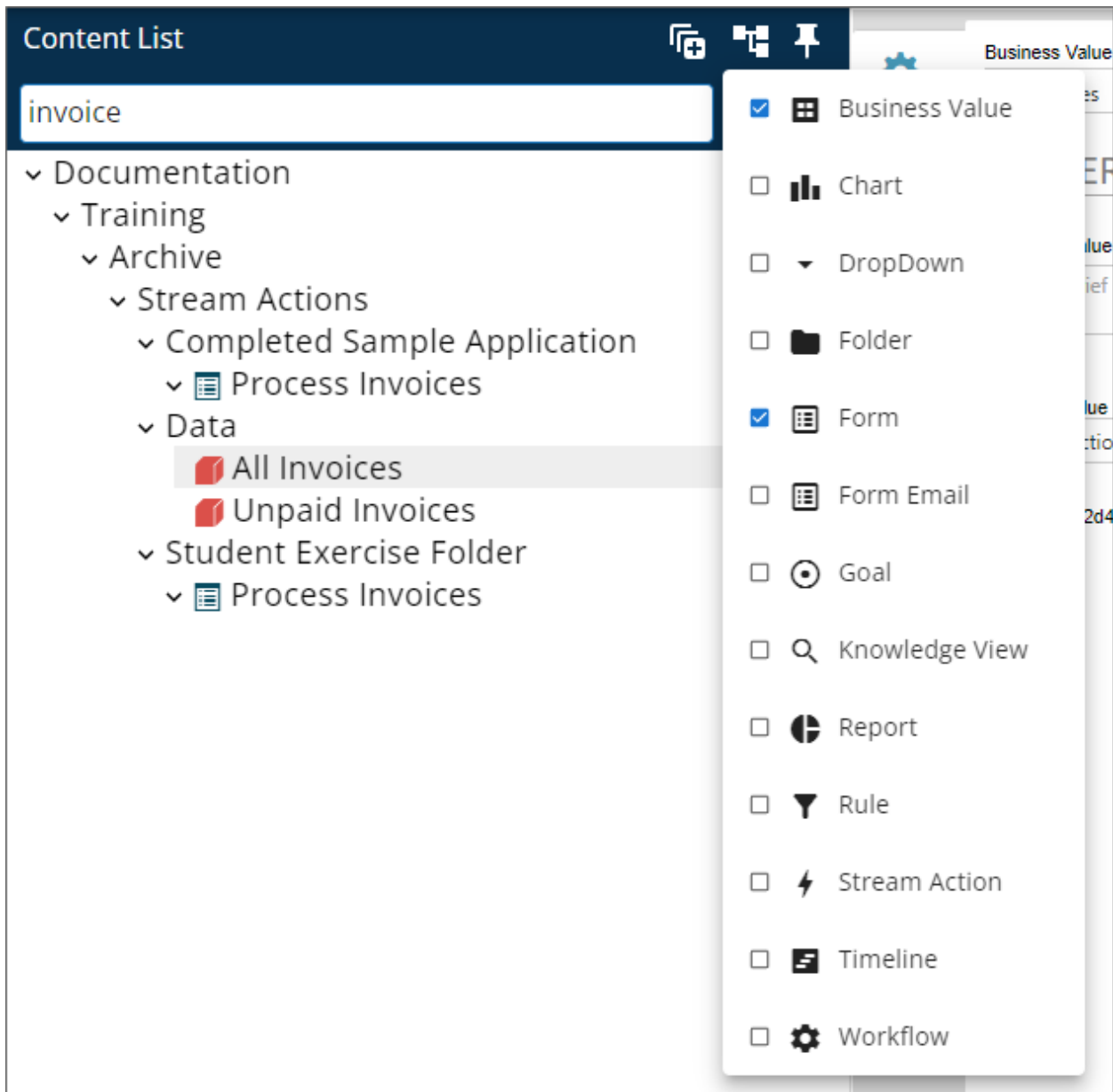
In the example above, using the search term "invoice" returns all Forms, Process Timelines, or other objects whose names contain the term "invoice". Clicking on any item in the tree will open it directly.



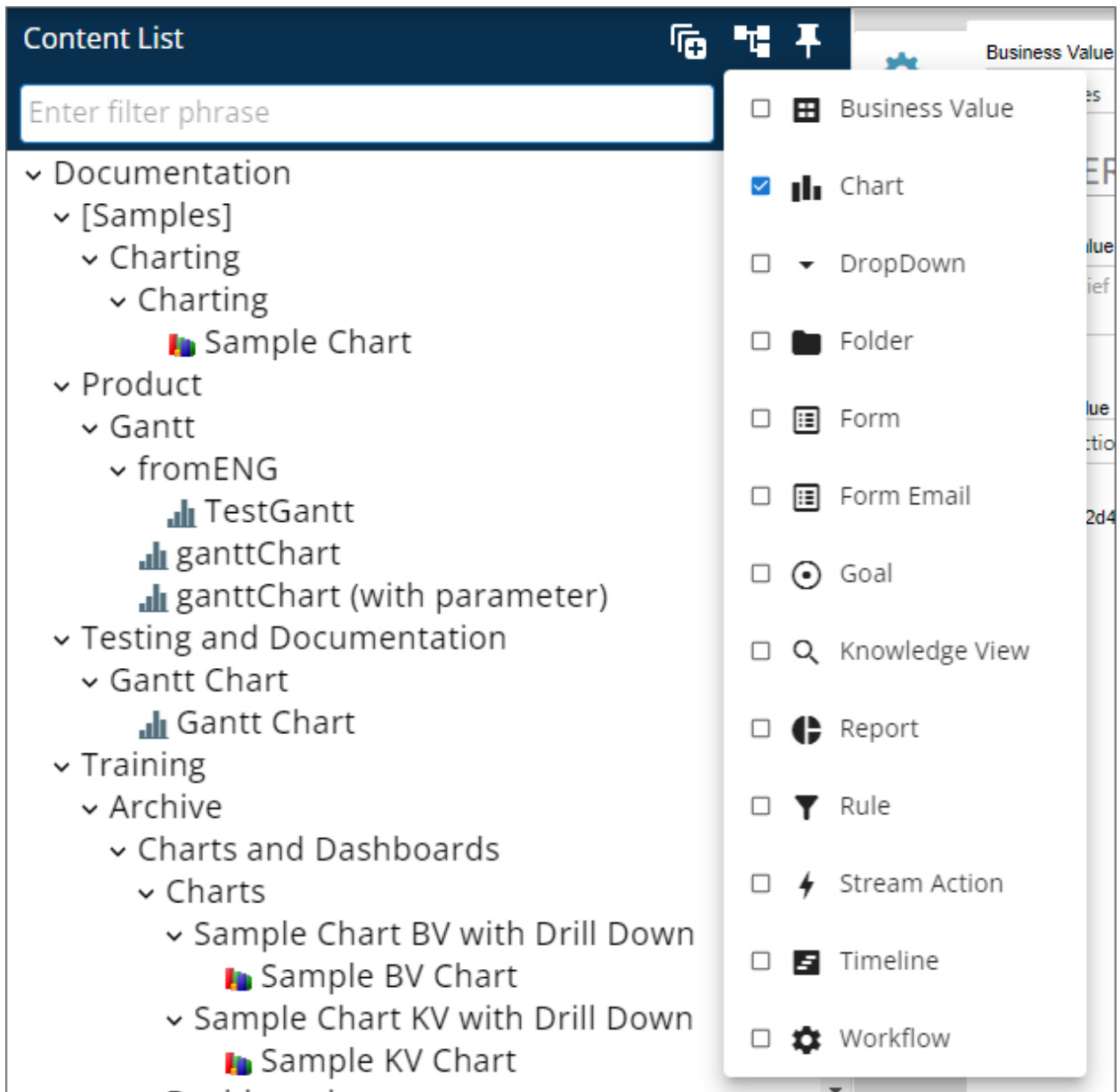
To the right of the search bar, a Filter menu enables you to select objects of a specific type, so that only those object types are displayed in the tree.



Selecting one or more items from the list of object types will filter the tree to show only objects of the selected types.

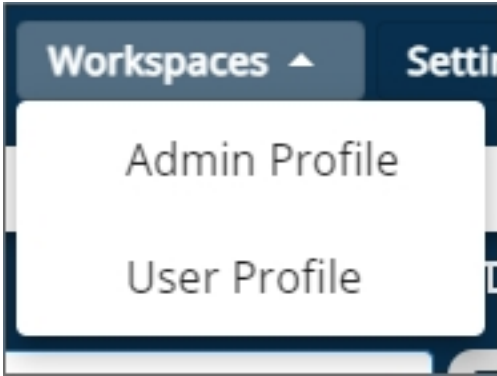


Notice that, in the example above, the filter works in conjunction with the search bar to show only Business Values and Forms whose names contain the search term. You do not, however, have to use the search bar in conjunction with the filter. If no search term is entered, the filter will still change the tree display to show only object types that match the filter you select.



### Workspaces Menu

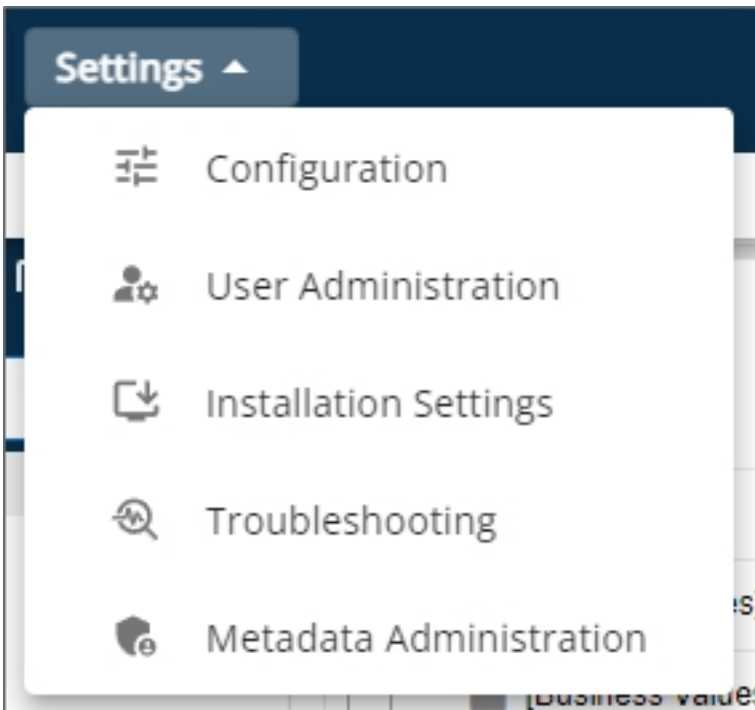
The **Workspaces** Menu provides you with access to all of the workspaces for which you are a member.



Clicking on any **Workspaces** menu item will immediately open that workspace to replace the [Content List](#). User who have access to only a single workspace will not be presented with this menu in the UI.

### Settings Menu

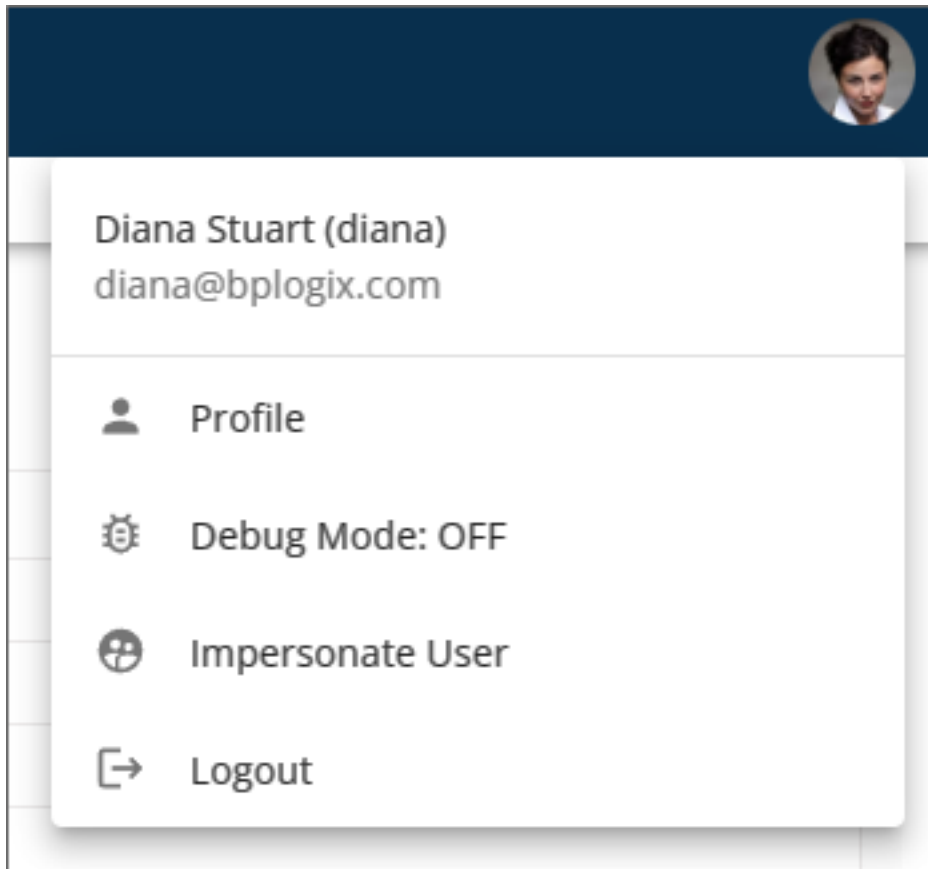
The **Settings** menu appears only for users with administrative access to the [IT Admin](#) or [Metadata Administration](#) areas of the Process Director installation.



Each menu item provides access to a specific section of the administrative portion of the installation. For the specifics of what each section of the IT Admin area contains, please refer to [IT Admin Area topic](#), or to the [Meta Data Administration](#) topic in the Administrator's Guide.

### User Profile Menu

The **User Profile** menu appears at the upper right corner of the screen.



The **User Profile** menu is denoted by one of two visual appearances. For users that do not have a profile image uploaded to their user profile, the menu will display as a generic user icon, as shown above. User with valid profile images will see their profile image displayed as a circular icon of the same size as the generic icon.

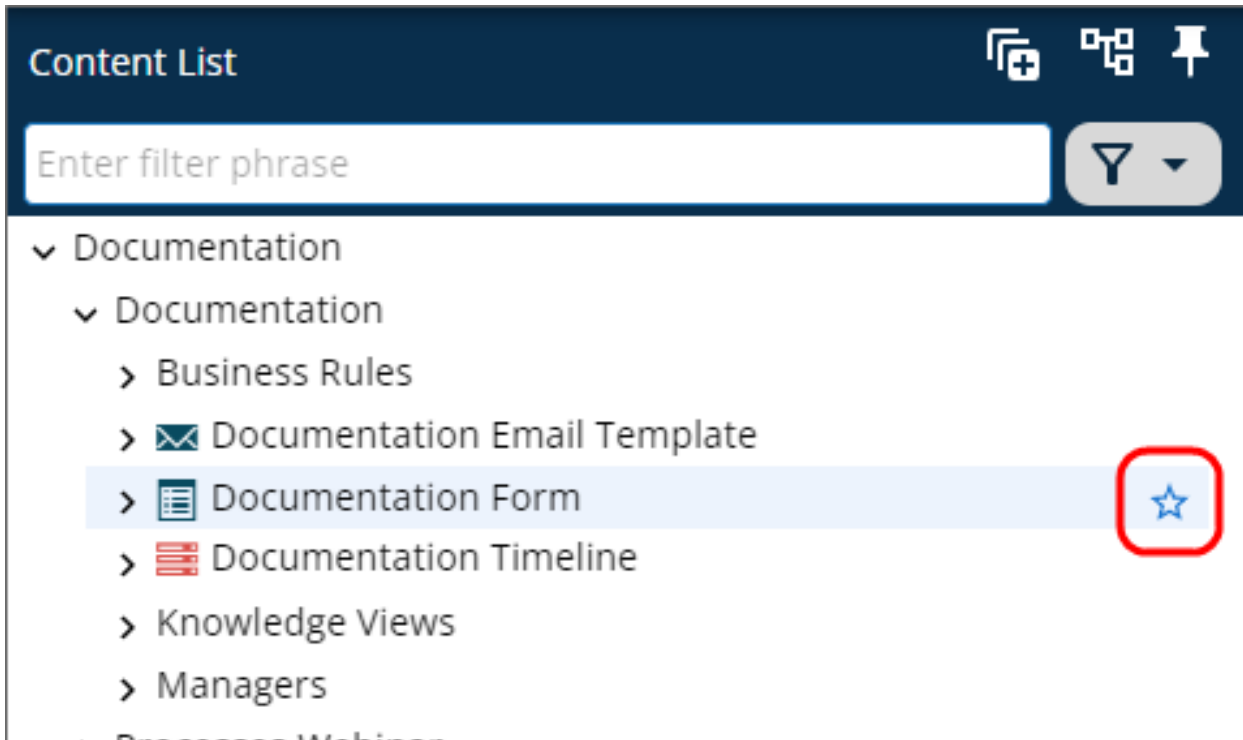
For most **built-in** users, the only two menu items displayed will be the **Profile** menu, which opens their **User Profile** page, and the **Logout** menu, which immediately logs them off of the system. **SAML** or **Windows** users on installations that use SAML/Windows Single Sign-On will *not*, by default, see the **Logout** menu item. A custom variable added in v6.1.0, [UserInfoShowSignOut](#), will enable the Logout menu to appear, if desired.

Users with the permission to view the Content List in [Debug Mode](#) have a menu item labeled **Debug Mode** that includes an indicator to show whether Debug Mode has been turned on or Off. In the example shown above, Debug Mode is off.

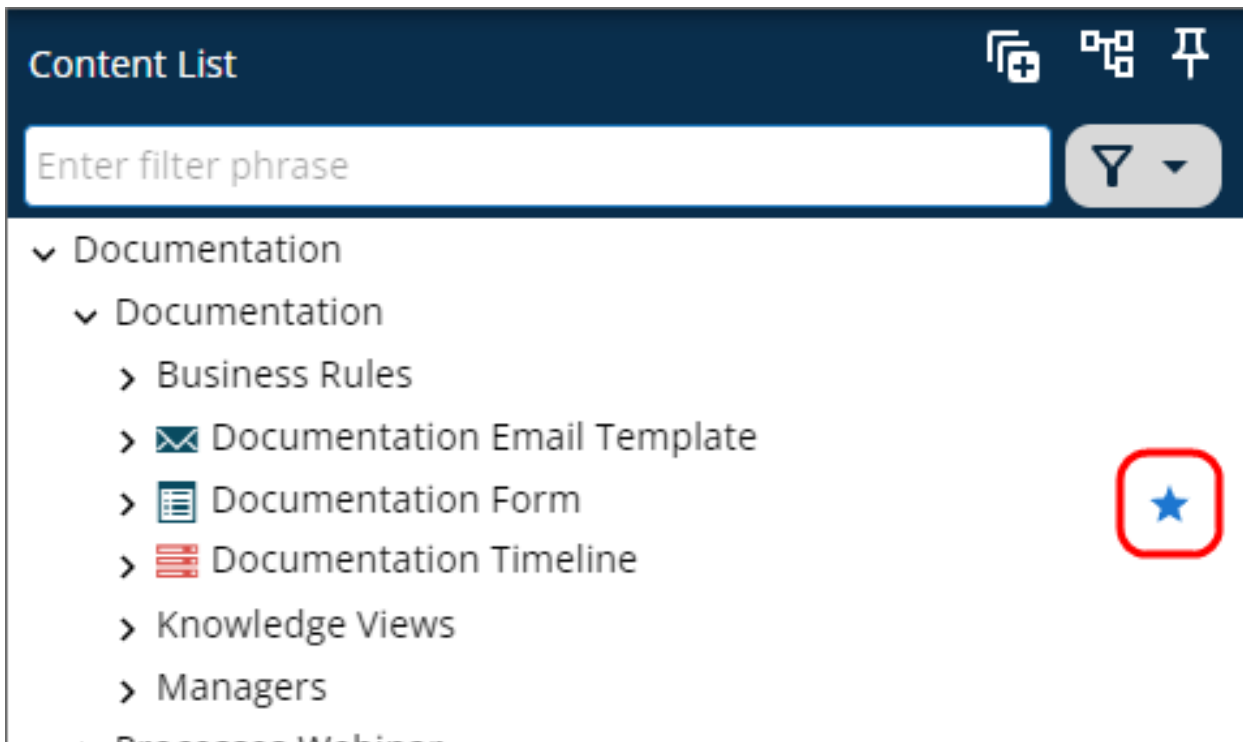
Users who have been granted impersonation ability will also see an **Impersonate User** menu item, which will open the **Impersonation** page to enable them to begin impersonating a different user. Form more information on impersonation, please see the [Impersonation topic](#) of the Administrator's Guide.

## Marking Objects as Favorites <#>

When using the [Content List](#), if you hover your mouse over an object, a small icon of a star outline will appear on the right side of the object.

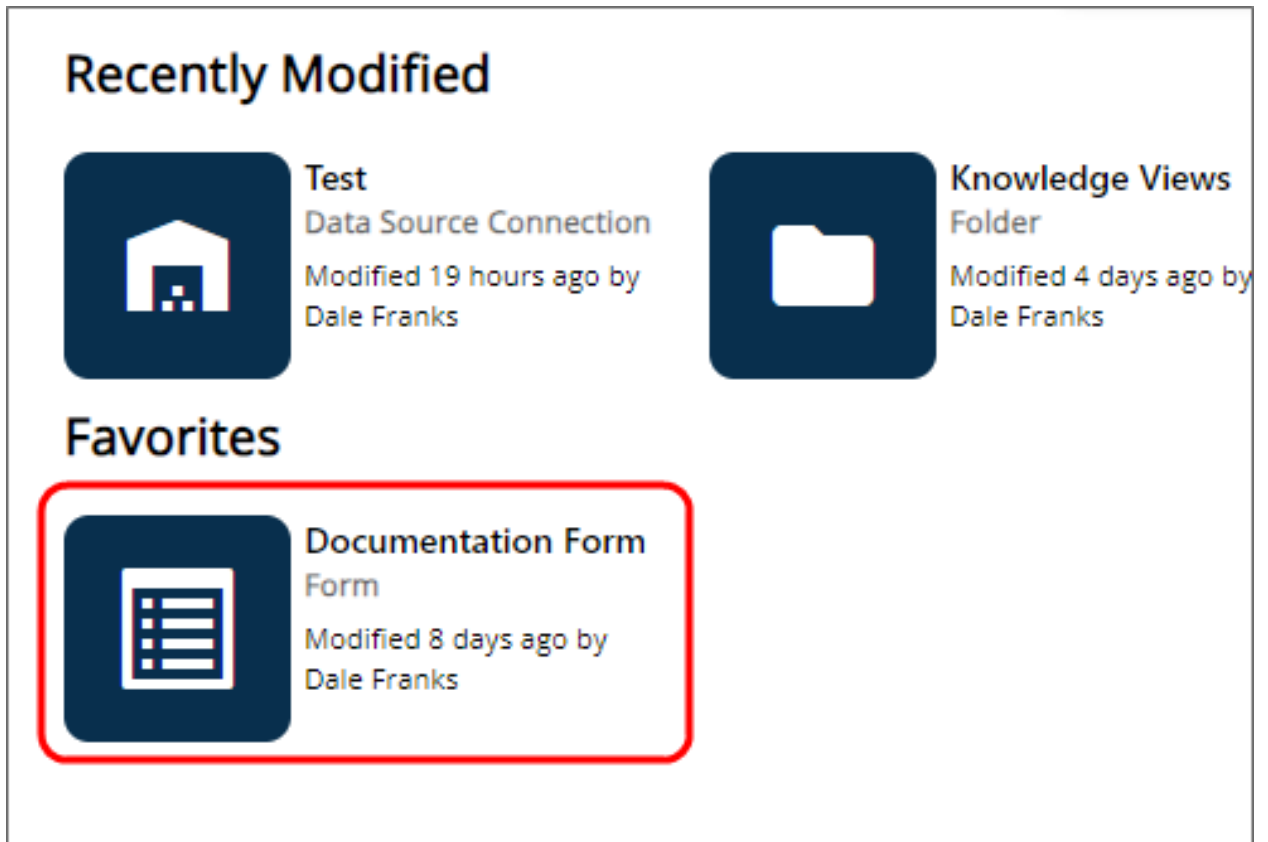


This icon serves as the **Favorite** icon for the object. Clicking this icon will mark the selected object as a favorite. Favorited items alter the user interface in two ways (though it may take a second or two for the changes to show after you click the icon). First, the icon display will change to permanently display a solid star in the **Content List**.





Second, the [Home](#) page will display the object as a favorite item in the [Favorites](#) section.



This feature enables you to mark any [Content List](#) object you'd like as a favorite item, so that you can open it directly from the [Home](#) page, without having to navigate through the [Content List](#) to find it. The UI changes that occur when an object is marked as a favorite will persist until you unmark the item as a favorite.

To unmark an item as a favorite, simply click on the [Favorite](#) icon again in the [Content List](#). The icon will change from solid to an outline, and the favorited item will be removed from the Favorites section of the [Home](#) page.

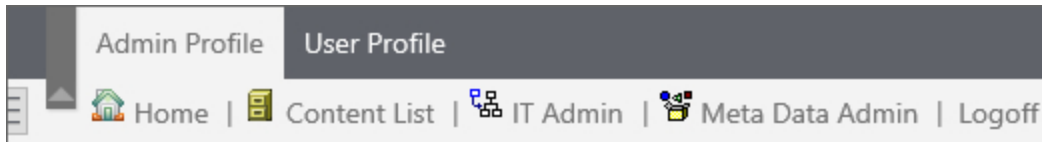
## Other Common UI Elements

Please refer to the [Common UI Elements topic](#) to learn about other common user interface conventions.

Process Director User Interface (Legacy)

## Navigation System #

Process Director's web interface for v6.0.300 and below provides a tabbed navigation system at the top of the browser, containing entries that will take users into various components of Process Director. After the initial installation of Process Director you'll see a very simple navigation scheme displayed.



The top row of the User Interface presents a tab for each Workspace to which you have been assigned. This top row will only appear when you've been given access to more than one Workspace. The second row displays the navigation buttons that have been configured for each Workspace. Selecting a Workspace from the top row of tabs will present different navigation buttons that are specific to that workspace.

The basic navigation scheme will be expanded through creating new workspaces for different users. Throughout this guide, you may see screenshots from various installations that have a number of different workspace navigation schemes. The navigation schemes for your particular installation will be quite different once you start implementing workspaces relevant to your processes.

For Process Director v5.23 and higher, a different user interface is also available. This user interface, called the Desktop Interface, is documented in the [Desktop Interface Workspace](#) topic. This is an optional UI, and must be specifically configured. Otherwise, the traditional Workspace will be displayed to users.

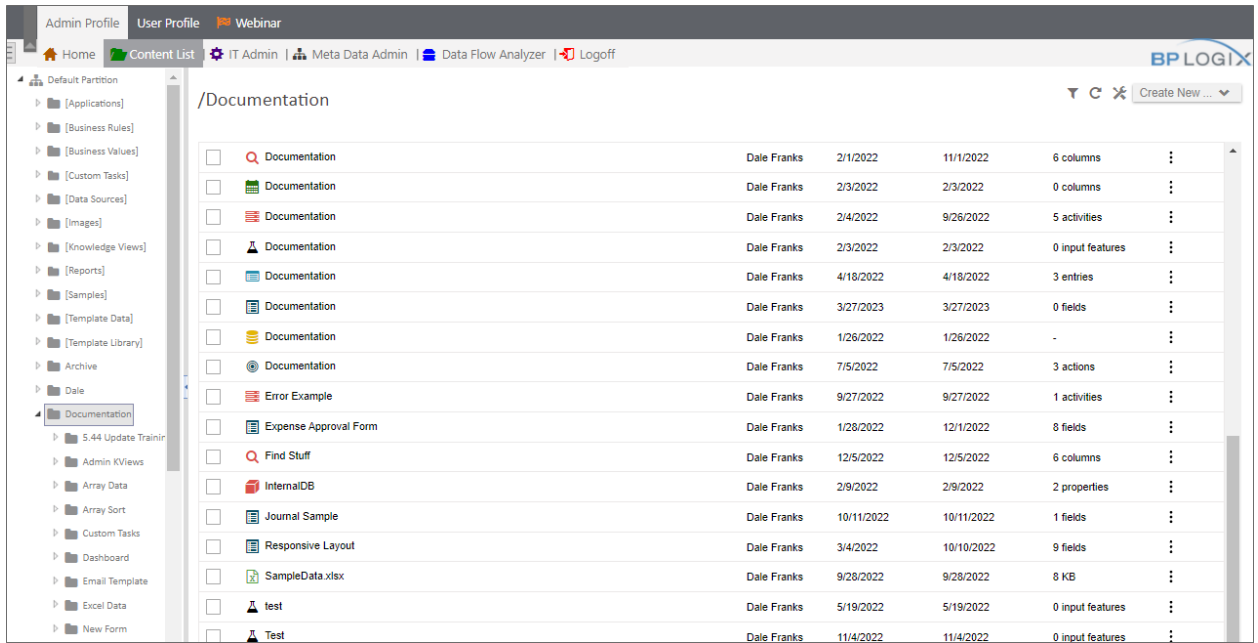
For detailed information about how to create and configure workspaces, please refer to the [System Administrator's Guide](#).

## Other Common UI Elements

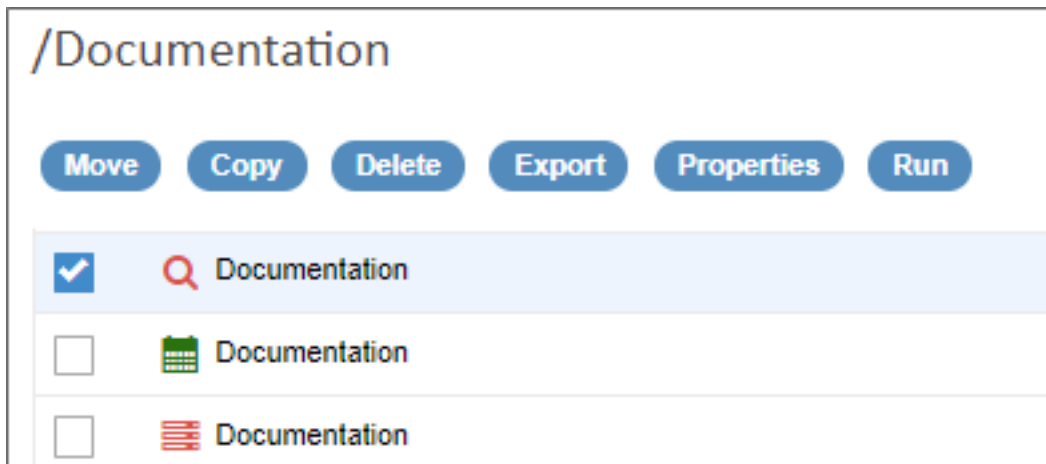
Please refer to the [Common UI Elements topic](#) to learn about other common user interface conventions.

## Common UI Objects

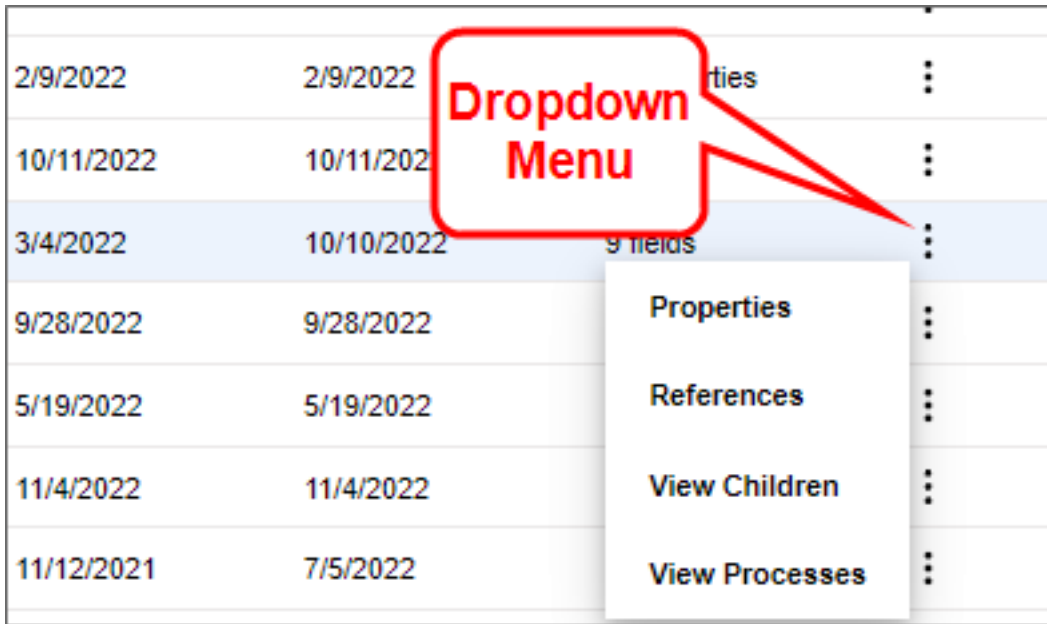
The release of Process Director v6.0 introduced an updated look and feel for the [Content List](#), as well as the display of Knowledge Views. While the functionality is largely unchanged, the UI presents it slightly differently.



When a [Content List](#) object is selected in the list, the action buttons that appear at the top of the Knowledge View are also styled differently, though the available options are the same.



Similarly, the Icons displayed on the right side of each object row have been replaced by a new, context-sensitive menu, from which you can select the operation that was represented by an icon in previous versions of the product.



This new UI appearance is optional, via the configuration of the [Global Knowledge Views](#) on the system.

## Folder Properties

Every folder in the [Content List](#) has folder definition properties that can be accessed by selecting **Properties** from the dropdown menu to display the folder definition.

Folder Name  Icon

### PROPERTIES

Description

Automatically start this process when objects are created in this folder: (optional)

Automatically start this process when objects are modified in this folder: (optional)

Automatically start this process when objects are moved into this folder: (optional)

We'll discuss the various tabs that are arrayed on the left side of the object definition below, but, for now, in the **Properties** tab of a Folder definition, you can rename the folder by changing the **Folder Name**

property. Similarly, you can change the default folder icon by clicking the folder icon displayed in the [Icon](#) property, and selecting a new Icon from the [Icon Chooser](#).

There are three very important properties displayed at the bottom of the Properties tab:

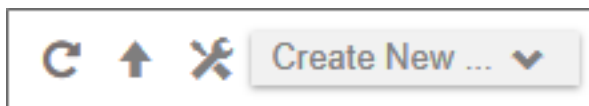
- [Automatically start this process when objects are created in this folder](#)
- [Automatically start this process when objects are modified in this folder](#)
- [Automatically start this process when objects are moved into this folder](#)

Each of these properties enable you to specify a process that you want to start automatically when the listed events occur. For instance, you can start a process automatically when any document is imported into the folder. This would enable you to set up a scheduled file import into the [Content List](#) folder, and, each time a document imports, run a process on that document, which will automatically attach the imported document to the process.

For Process Director v6.1.0 and higher, the [Automatically start...](#) properties enable you to chose a Stream Actions object to invoke. Prior versions only enabled you to choose only a process definition, e.g., a Process Timeline object.

## Standard Icons #

At the top right corner of the [Content List](#), a series of standard icons and a [Create New](#) menu appears.



From left to right, the standard icons you see will be:

- **Refresh:** This icon will refresh the current screen when clicked.
- **Navigate to Parent:** For Process Director v6.1.0 and higher, this icon appears inside folders that are located below the root level of the Partition. Clicking this icon will automatically navigate you to the parent folder of your current location.
- **Properties:** This icon will, when clicked, open the [Properties](#) screen of the current [Content List](#) folder.

The [Create New](#) menu is used to create new Process Director objects inside the current folder. For more information on the operation of this menu, please refer to the topic for [Creating Content List Objects](#).

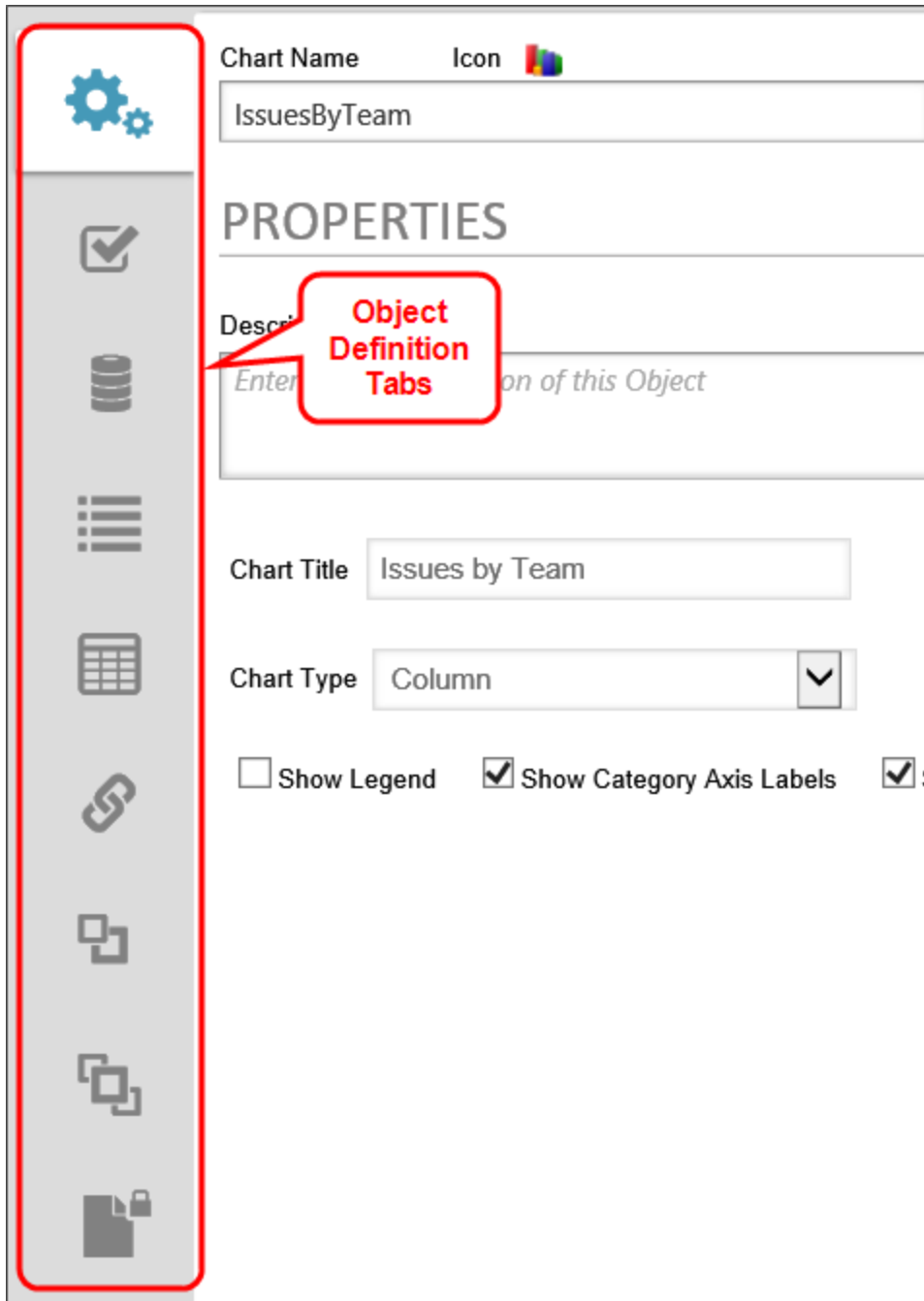
## Action Buttons #

Objects in the [Content List](#) can be selected by checking the box that appears at the left side of each row in the [Content List](#). When you click the check box, to select an item, several Action Buttons will be displayed, to enable you to move, copy, export, etc., the selected item.



## Object Definition Interface #

The definition for each Process Director definition object is presented in a tabbed interface.



Each object has a set of tabs that are specific to the object type of the definition. Additionally, each process Director Object has a set of common tabs that are the same for every object.

### Properties Tab #

Every Process Director object has a **Properties** tab, and the contents of this tab will vary widely. For instance, the **Properties** tab for a definition object usually contains the basic configuration properties for the object, while the **Properties** tab of an object instance will display, but not generally allow you to configure, some properties of that instance. Let's take the Form object as an example. The Form definition's **Properties** tab provides several configuration settings that you can edit.

The screenshot shows the 'Properties' configuration window for a form named 'Purchase Request (Online)'. The window has a sidebar with various icons and a main content area. The main content area is titled 'PROPERTIES' and contains several sections:

- Form Name:** Purchase Request (Online)
- Description:** Purchase Request form built with the Online Form Designer
- Instantiated Form Name:** {FORM\_DEF\_NAME} Submitted On {CREATE\_DATE}
- Form Fields:** Form Fields are Enabled by Default (dropdown menu)
- Entire form is read-only when:** Click to create condition ...
- Options:**
  - Automatically start this process after Form is submitted (optional): PR/PO Process (Online)
  - Workflows Associated with Form (optional):
  - Timelines Associated with Form (optional):
  - Default Groupname for Form Instances:
  - Form Script (optional):

On the other hand, the Form instance's **Properties** tab displays the field contents of that form instance, and, for Process Director v5.1 and higher, enables you to refresh the data, or to export the data to a CSV file by using the icons that appear in the upper right corner of the information grid.



Control Name	Control Type	Value
Accounts	ListBox	Equipment, Miscellaneous
ArrayItems	Array	1
AvailableAccounts [1]	DropDown	Conferences/Meetings

Again, the types of information you'll see on the **Properties** tab is widely divergent, so the detailed explanation for configuring the **Properties** tab will be given in the documentation for each individual object.

## Object Links Tab #

The **Object Links** tab shows all Process Director objects which have a link to the object definition, or to which the definition has a link.

Used In	Object Type	Folder Path	Referenced In	Link Type
Webinar	Workspace			Workspace Window
Training	Knowledge View	/Documentation/Video/Samples		Link
sample 2	Knowledge View	/Documentation/Video/Samples		Link
PR/PO Process (Online)	Timeline Definition	/Training/Specialist/Processes		Link
Greater Than 10k	Business Rule	/Training/Core/Business Rules		Link
test	Business Rule	/Training/Specialist/Resources		Link
More than 10k	Business Rule	/Training/Specialist/Business Rules		Link
Generate PO (Online)	Workflow Definition	/Training/Specialist/Processes	Workflow Step : Copy Form Data	Form Process Task
PR/PO Process (Online)	Timeline Definition	/Training/Specialist/Processes	Timeline Activity : Set PR Number	Form Process Task
Training	Knowledge View	/Documentation/Video/Samples	KView Column : Submitted On	KView Column
Training	Knowledge View	/Documentation/Video/Samples	KView Filter	Condition
Training	Knowledge View	/Documentation/Video/Samples	KView Column : Submitter	KView Column
Training	Knowledge View	/Documentation/Video/Samples	KView Filter	Condition
Training	Knowledge View	/Documentation/Video/Samples	KView Column : Vendor	KView Column

## Object Links

Examples of linked objects would include:

- The Process Timeline.
- Forms that use the Business Value.

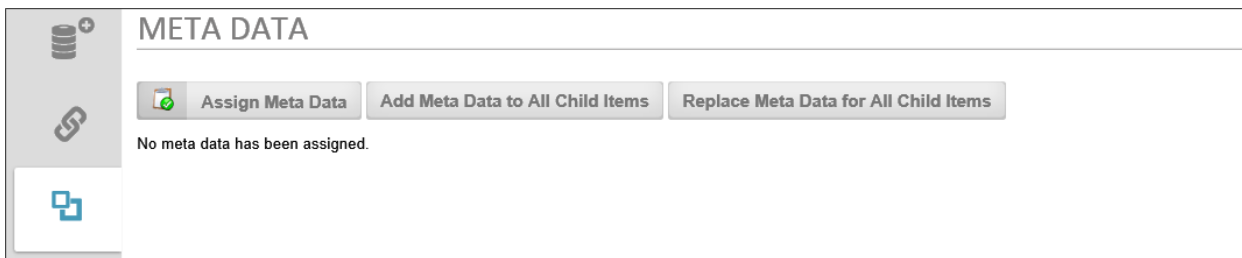
- Dashboards, Goals and Business Values linked to the definition.
- System Variable references to the object

Check boxes enable you to show System Variable or Custom Task references to the object, if desired:

- Search for System Variable strings referencing something in this definition
- Search Custom Task configurations that may be referencing something in this definition (This may require searching for SysVar strings also)

It is important to remember that these two check boxes are interdependent. For example, clicking on the [System Variable strings...](#) check box won't return system variable strings used in a Custom Task unless the [Custom Task configurations...](#) check box is also checked. Similarly, checking the [Custom Task configurations...](#) check box won't return results for references by System Variable in a Custom Task unless the [System Variable strings...](#) check box is also checked.

## Meta Data Tab #



Meta Data categories and attributes can be assigned to each Definition.

### Meta Data Buttons

When you click the [Assign Meta Data](#) button, it opens a dialog box from which you can choose the Meta Data categories you wish to assign.

When you click the [Add Meta Data to All Child Items](#) button, all of the Meta Data assigned to the definition will be copied to all of the instances *in addition to* any Meta Data that is already assigned to the instances.

When you click the [Replace Meta Data for All Child Items](#) button, all of the Meta Data assigned to the definition will be copied to all of the instances *and replace* any Meta Data that is already assigned to the instances.

For more information about Meta Data and its configuration, please refer to the [Meta Data section](#) of this guide.

## Versions Tab #

Form Name 
Icon 
OK

---

### VERSIONS

Version snapshots of this Form are listed below. Most recently applied version: None.  
 Applied: n/a  
 Form last saved: 2/5/2019 9:00:55 AM

Label:

Object Version Description:

[Create New Version](#)

Label	Saved By	Version Number	Version Date			
Original <i>First version save</i>	Dale Franks	1	2/5/2019 9:00:19 AM	<a href="#">Download</a>	<a href="#">Apply This Version</a>	<a href="#">Delete</a>

Process Director implements versioning for all objects, and each version consists of a complete snapshot of the object. Each version can be downloaded, deleted, or applied to the object to replace the current version.

To create a new version of the object, first fill out the version properties. The **Label** is the name you wish to give the version, and the **Description** is a brief explanation about the version that can provide additional information. Once you've finished setting the properties, click the **Create New Version** button to create the version.

Each version will display in a table that specifies the Label, the person who created the version, the version number, and the version creation date. Additional columns provide links to download an XML export of the version, to replace the current version with the selected version, or to delete the version.

While any object can be versioned, BP Logix recommends creating versions at the application folder level to create a snapshot of the entire application, rather than versioning each object individually. Please refer to the topic on [importing and exporting content](#) for recommendations on how to organize your Content List into application folders.

### Label

The name of the version object.

### Object Version Description

A description of the versions purpose, changes made from previous versions, or other relevant information.

Once you've filled out the properties, click the **Create New Version** button. Process Director will record a snapshot of the version, and a new version row will appear at the bottom of the screen.

Links on each version row give you the option to apply the version as the current version in use, or delete it. If you choose to apply a version, Process Director will display a status screen to notify you of the success or failure of the attempt to apply it.

From the list of versions, you may

- Download an exported XML file containing the object by clicking the Download link.
- Revert to a desired version of the object by clicking the [Apply this Version](#) link for the version to which you wish to revert.
- Delete version by clicking the [Delete](#) link.

When a folder is selected for versioning, the version will contain all of the [Content List](#) objects contained in the folder. Reverting a folder to a previous version, therefore, will revert all of the objects in the Content List to the reverted version.

### Form Instance Versions

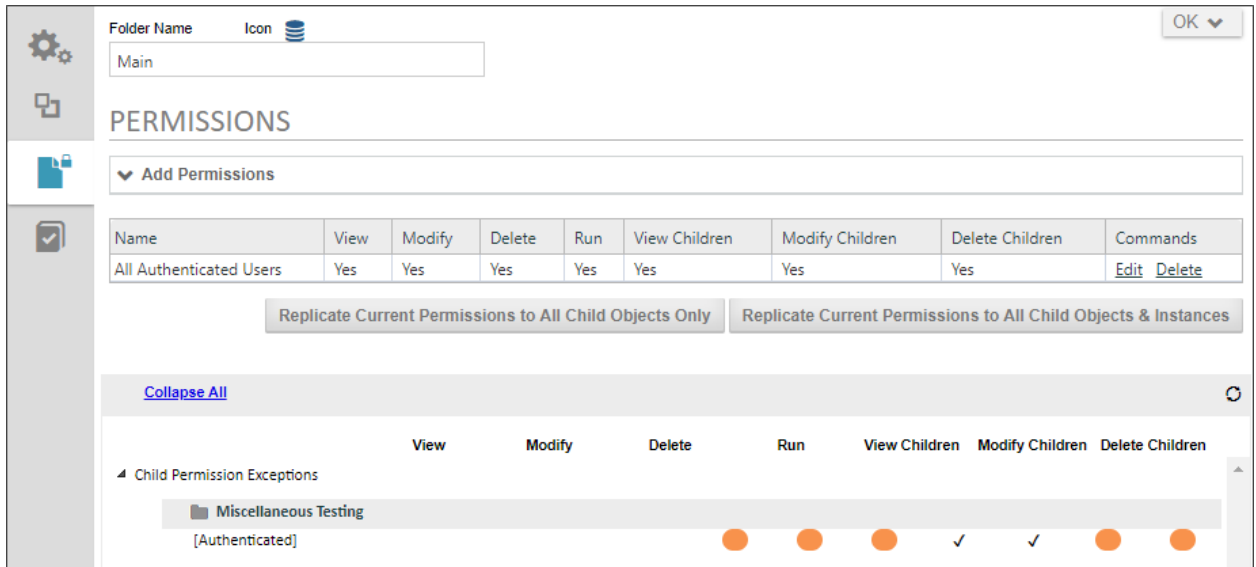
Just as definition objects have a version, Form instances also have a versioning scheme that is automatically applied by Process Director. When a Form instance is opened for the first time by a user, the Version number is "0". Each time the Form instance is edited and saved, the version number increases by an increment of 1. Unlike definition objects, however, Form instances can also have negative version numbers.

A negative version number for a Form instance has a special meaning. It means that the Form instance has been created but *not* submitted. For example, a Form with a version number of -1 is a form attached to a Timeline instance but which has never been viewed by a user. A version number of -2 means that the user opened a new form and did a "save for later", which did not submit the Form, even though the data was saved.

By default, Forms with a negative version number won't be displayed in a Knowledge View because they aren't yet submitted forms, but are only partially complete. The negative version number also tells the Process Director that any configured form field "default values" still apply to the Form. Thus, when a user does open a form with a version number of -1, the default form field values will be applied. You can increment the version from a negative to a positive value, for example, by using custom scripting, but this will prevent default values from being applied.

### Permissions Tab <#>

Process Director implements fine-grained permissions for all objects. Though we recommend that permissions be set at the folder level, each Process Director object may have unique permissions set on the object, using this tab.



For Process Director v5.1 and higher, a Permissions Exceptions section will appear at the bottom of the page that shows all child objects whose permissions don't match that of their immediate parent. However, if a folder's permissions match its parent but it has children that don't match its own permissions, then that folder will be shown in order to show the children whose permissions don't match their parent.

For detailed information on how to set permissions, please see the [Permissions](#) topic.

## References Tab #

Several Process Director object definitions include a **References** tab that displays a list of all objects or system variables that the object references. The purpose of the References tab is to attach a reference object, such as a file or document, to an object definition that will, when an instance is created, attach the reference object to the instance. For example, perhaps there is a document you'd like to upload, or a [Content List](#) document that you'd like to add, as a reference to the object definition. On the references tab, there is a **Create New...** actions menu in the upper right corner of the screen that enables you to choose:

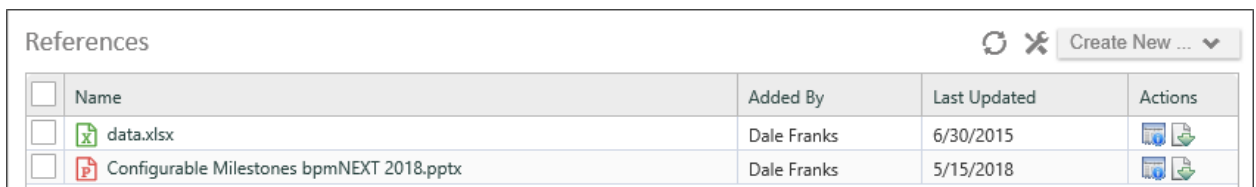
### Document/File

If you choose this option, the upload dialog box will appear to enable you to upload a document or file as a reference to the definition object.

### Reference to Item

If you choose this option, an Object Picker will appear to enable you to select an item from the [Content List](#).

Selected reference items will appear in the list on the **References** tab.



You can select the check box next to each reference item to view its properties, delete it, or set a **Group Name** for the object.

Once an instance of the definition object is created, the reference objects will automatically be attached to the instance. You can view the attachments in the Instance properties of the object in the [Content List](#). For forms, you can view the attached reference objects via a **ShowAttach** control on the form. For example, when a new form instance is created using the reference objects above, you can view the attached reference objects on the form instance.

**Upload Document**

Configurable Milestones bpmNEXT 2018.pptx	<a href="#">View</a>	<a href="#">Download</a>	5/15/2018 2:37 PM	Dale Franks
data.xlsx	<a href="#">View</a>	<a href="#">Download</a>	6/30/2015 12:18 PM	Dale Franks

**OK**    **Cancel Changes**

## Import History Tab <#>

Process Director v4.55 and higher maintains a log of all import events, Active Directory Synchronizations, and Evaluations. These logs are displayed for each object in the object definition's **Import History** tab.

## IMPORT HISTORY

Events in Which This Object was Imported:

Import Started At	Object Name
1/4/2018 10:08:16 AM	/webinar/201711

This import history can be disabled using the [fEnableDatabaseLogs](#) custom variable. By default, the logs are maintained for 30 days, but you can change the number of days to maintain the logs in the database by using the [nImportLogDays](#) custom variable.

## Milestones Tab #

The screenshot displays the 'MILESTONES' configuration interface. At the top, there is a 'Form Name' field containing 'DocumentFeedback' and an 'Icon' field. Below this is the 'MILESTONES' title and an 'Add Row' button. A table with the following columns is visible: Trigger, Milestone Group, Milestone Name, Icon, Journal Entry, and Condition. The first row contains 'Choose a Trigger' in the Trigger column, an empty field in Milestone Group, an empty field in Milestone Name, a red 'X' icon in the Icon column, an empty field in Journal Entry, and 'Click to create condition' in the Condition column. A sidebar on the left contains various icons for navigation and actions.

Users of Process Director v4.6 and higher have a **Milestones** tab available in event-driven objects such as Forms or Process Timeline definitions. Milestones are also accessible via the [Milestone System Variables](#).

When creating a process application in Process Director, there are a number of Process Timeline and form events you can use to drive some decisions. But application events, while helpful, often don't provide the business-level, broad semantic context that is sometimes needed. Broad semantic context is where Milestones live. For instance you might want business-level events that provide information to drives business decisions such as:

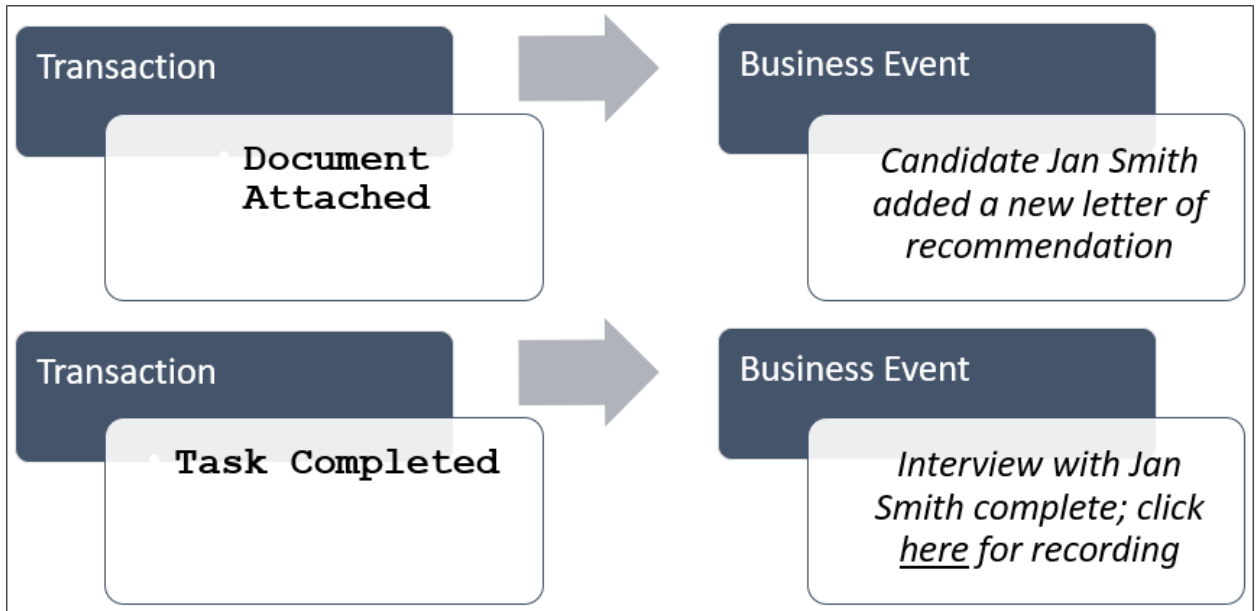
- “Application for admission complete”
- “Additional information needed to pursue claim”

Process application events generally occur sequentially, but Milestone achievement may rely on disparate, asynchronous events and decisions. Similarly, while the events at the application level are very good at letting you know where you are in a specific process, they don't provide high-level, business-related information. That high-level information is often necessary to drive lower level transactional and semantic decisions, e.g.:

- “Begin review when application for submission complete”
- “Notify insured when certain documents remain outstanding”

Process Director enables you to generate business events from the lower-level transactional data, then combine business events to produce Milestones. You can use both events and milestones to drive application behavior and collaboration.





Milestone logging enables you to create special logs for business events and display those logs on a Form, using the [Journal](#) control.

You can add as many Milestones as you desire by clicking the **Add Row** button at the top of the tab.

For every event that you'd like to log, you have the following options.

### Trigger

The following triggers are generally available, though they don't include the Timeline Activity events, such as Activity Started, etc.:

EVENT	DESCRIPTION
Process Started	Invoked whenever a Process Timeline is started.
Activity Started	Invoked whenever a Timeline Activity is started.
Activity Completed	Invoked whenever a Timeline Activity is completed.
User Task Complete	Invoked whenever a user completes an assigned task.
Document Added to Process	Invoked whenever a new document is attached to a process.
Document Updated in Process	Invoked whenever an existing document attachment is edited.
Form Added to Process	Invoked whenever a new Form is added to a process.
Condition Met	Invoked any time that a condition is met while milestone processing is occurring.

### Milestone Group

The **Milestone Group** is a text value that is used to organize the activities. When you assign a log event to Display in the Activity Log control, all of the available log items will be grouped into the groups you

create with this setting. So, you may have a "Documents" group, into which you might place all of the events that are invoked when document attachments are added or edited during the process. In the Object Picker that you use to display events in the [Journal](#), all of those events will be organized into a Treeview Node, with the Group Names you create at the highest level of the Treeview.



### Milestone Name

The text name you wish to use for the Milestone.

### Icon

You can choose an icon for the event from the icon chooser.



Milestones can't use the older, raster-based icons displayed in the Older Built-in Icons or Custom Icons tabs of the Icon Chooser.

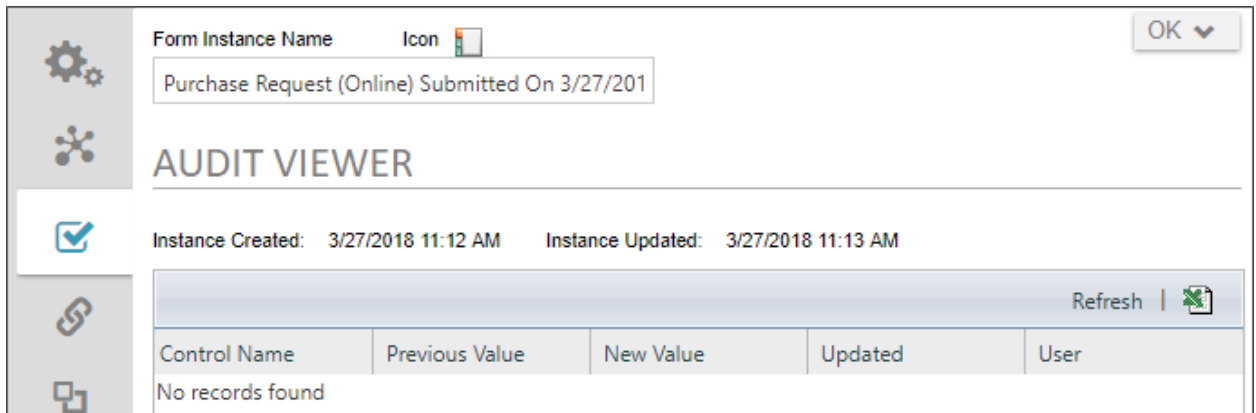
### Journal Entry

This is the logging message you wish to appear in the [Journal](#) control.

### Condition

You can use the [Condition Builder](#) to create a condition set that must apply before the event is logged. Only events that match the condition will be logged.

## Audit Viewer Tab #

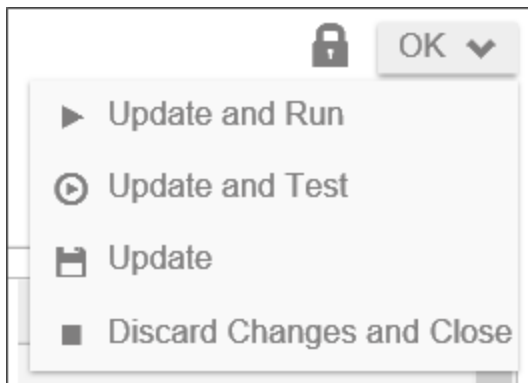


Audit logs for Form and Case instances can be viewed on the **Audit Viewer** tab for the instance. Fields changes and other audit information will be displayed on this tab for instances that have the [Audit Field Changes property](#) checked in the object definition's properties.

The view can be refreshed by clicking the **Refresh** link, or exported to a CSV file by clicking on the Excel document icon, both of which are in the upper right corner of the information grid.

## Object Control Menu #

Overall control of saving or running an object definition is embedded in the dropdown menu at the upper right of the definition. While not every object has all of the menu options described below, in general, object definitions give you the following control options.



Clicking the **Lock** icon will lock the definition for editing and prevent other users from editing the definition. Locked definitions are identified by the **Release Lock** icon (🔒) replacing the **Lock** icon.

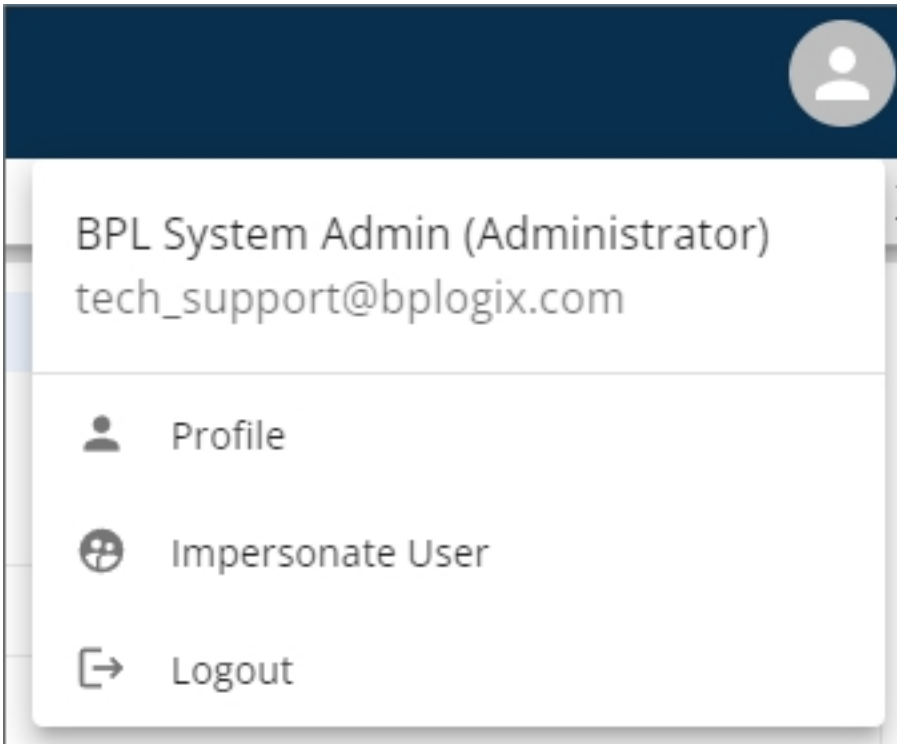
The **OK** button will save and close the definition automatically. When you put your mouse over the **OK** button, additional menu option will open that you can select.

- The **Update and Run** menu item will save the object definition and open a new instance of the object.
- The **Update and Test** menu item will save and Form definition and open test version of the Form. The Test version shows you the look and feel of the Form, but won't instantiate a process. Instead, the Form will, when submitted, show a list of the values that would be saved with a live Form instance.

- The **Update** menu item will save the definition in its current state.
- The **Discard and Change** menu item will close the definition without saving any changes you may have made.

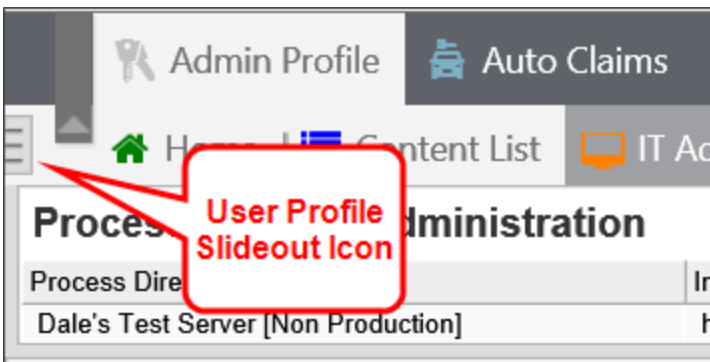
## User Profile #

For Process Director v6.0.300 and higher, a user Icon in the upper right corner of the screen enables access to the User Profile page. Clicking this icon opens the **User** menu.

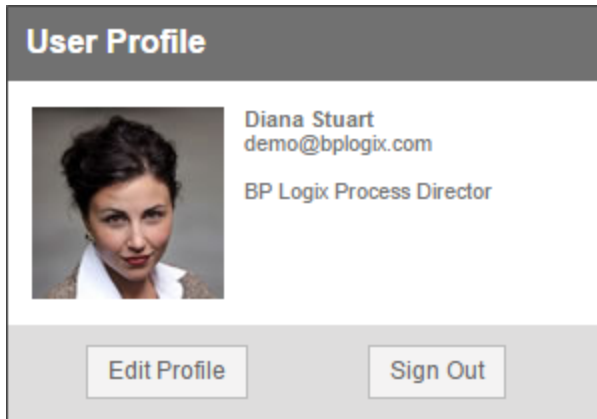


From the **User** menu, you can select **Profile** to display the **User Profile** page. Please see the

For users of earlier version of Process Director, the **User Profile Slideout** is a control that appears when you click the upper left corner of the Process Director screen. For users of Process Director 5.23 and higher, the Desktop interface for Workspaces presents the User Profile differently. Please refer to the [Desktop Workspace](#) topic for more information.



Clicking the Icon opens the Slideout to display the basic user information.



You can access and edit the [User Profile](#) page by clicking the [Edit Profile](#) button to open the user's profile page.

### User Profile Page

From the user profile page, the user can edit their time zone/localization settings, perform delegation (if allowed by the System Administrator), and add personal/signature images to their profile.

User Settings, Built-in

User ID  
diana

Email Address  
demo@bplogix.com

Display Name  
Diana Stuart

Change Password


Delegate all of my tasks to (User):  
Search for user Delegate User

Share my tasks with these users:  
Search for users

Preferred Language / Locale  
English

Users Time Zone  
Default Time Zone Auto DST

Disable All Your Emails

Picture Image  
  
Browse Upload Remove

Signature Image  
Diana Stuart  
Browse Upload Remove

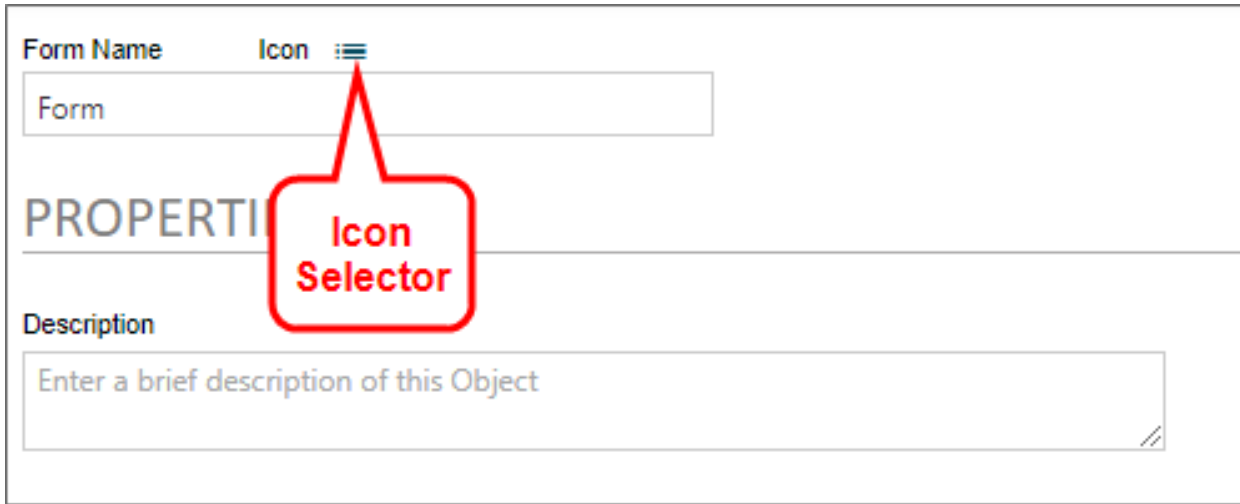
## Icon Chooser

Every object definition and instance is assigned an icon in Process Director.

For Process Director v6.0.300, a new suite of default icons for each object type has been implemented.

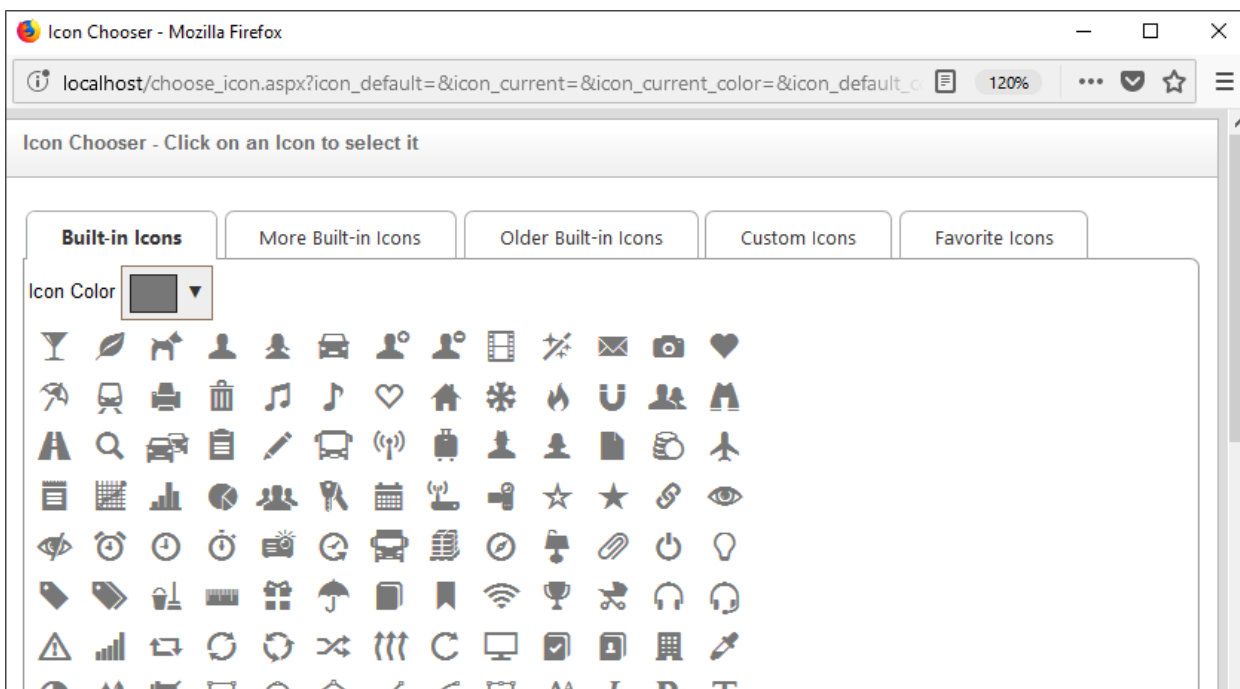
/Documentation/New Objects	
<input type="checkbox"/>	<b>Name</b> ▲
<input type="checkbox"/>	▼ Business Rule
<input type="checkbox"/>	📊 Business Value
<input type="checkbox"/>	👛 Case Definition
<input type="checkbox"/>	📊 Chart
<input type="checkbox"/>	📊 Dashboard
<input type="checkbox"/>	🏠 Datasource
<input type="checkbox"/>	▼ Dropdown Object
<input type="checkbox"/>	📄 Form
<input type="checkbox"/>	🎯 Goal
<input type="checkbox"/>	🔍 Knowledge View
<input type="checkbox"/>	👤 Machine Learning Definition
<input type="checkbox"/>	📅 Process Timeline
<input type="checkbox"/>	📊 Report
<input type="checkbox"/>	⚡ Stream Actions Object

Anywhere in the settings for any object where you see an icon displayed, you can click on the icon selector to open the [Icon Chooser](#) and select an icon for your object. So, for any object, you can replace the default icon with a different icon of your choice.



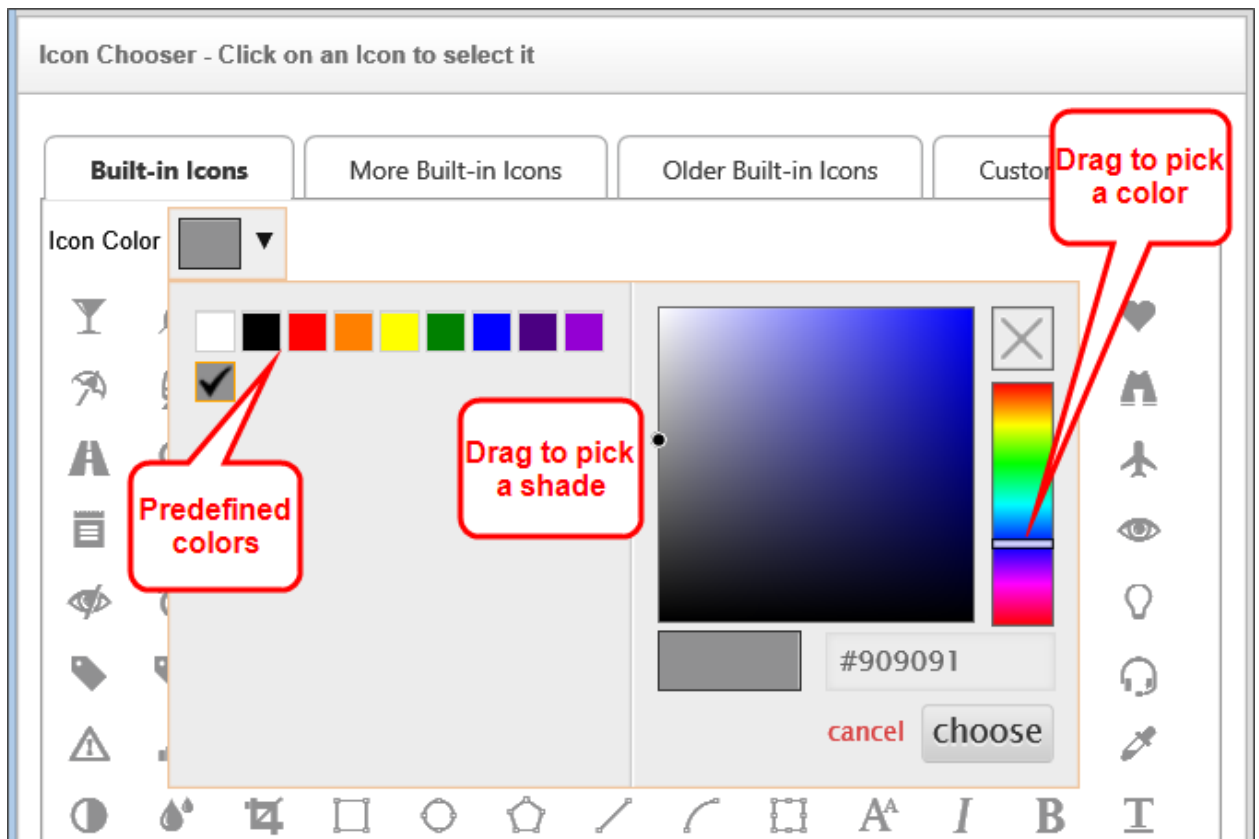
The [Icon Chooser](#) is a tabbed form. The first two tabs contain the standard Built-In icons available in process Director. The third tab contains the legacy icons that were used in prior versions of process Director, and remain in the product for backwards compatibility with earlier versions. The fourth tab displays an custom icons that exist in your installation.

**i** System administrators may also add custom icons, and instructions for doing so can be found in the [Creating Custom Icons](#) topic of the System Administrator's Guide.





To select a built-in icon, first choose a color, if desired, to use a different color than the default gray color. The **Select Color** dialog box enables you to select any desired color. legacy icons and custom Icons are image files, so colors can't be selected for those icon types.

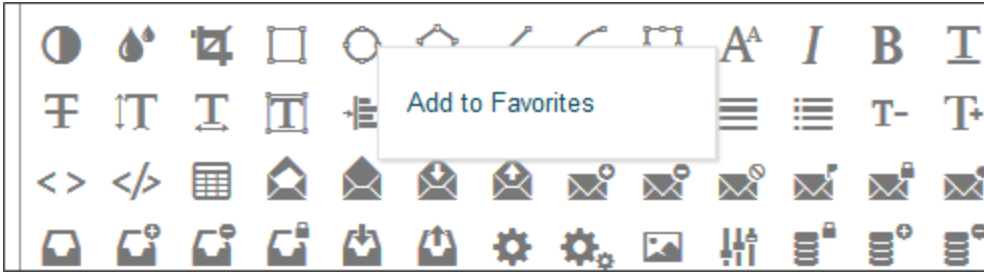


You can choose one of the pre-defined colors on the left side of the **Select Color** dialog box, or you can use the color palettes on the right to select any custom color you desire. If you select a predefined color, the **Select Color** dialog box will automatically close when you select the color. If you use the custom palettes, click the Choose button when you've configured the desired color, and the **Select Color** dialog box will close. Whichever method you choose to select the color, all of the built-in icons shown in the current tab of the **Icon Chooser** will be displayed in the selected color when the **Select Color** dialog box closes .

Find the icon you'd like to use and click on it. The **Icon Chooser** will close, and your selected icon will be applied to the object.

To change an icon, simply click on the object's icon selector again. When the **Icon Chooser** opens, your selected icon will be highlighted. You can then repeat the process above to select a different icon.

Finally, you can store your most-used icons on the **Favorite Icons** tab of the chooser. Simply find the icon you like, right-click the icon, and select **Add to Favorites** from the pop-up menu that appears.



Once you do so, that icon will now appear on the **Favorite Icons** tab.

**i** For Process Director v5.43 and higher, icons used on controls will no longer be grayed out when the control is disabled. This change was made to avoid confusion with grayscale icons that were part of the page design. All icons for disabled fields will now display in their configured colors.

## Condition Builder

**Conditions** are true or false statements that are evaluated in order to determine an appropriate response. Process Director includes a **Condition Builder** that enables you to create conditions to test nearly any value you desire. You'll see the **Condition Builder** in many different places, such as when to make a form field enabled or hidden, or what value to return from a Business Rule. The **Condition Builder** is usually invoked by an action link that is labeled **Click to create condition...** Anywhere you see this link, you can open the **Condition Builder** by clicking on it.



Adding a condition to a specific option makes it easy to set a value or perform an action based on your preference.

### Adding a Condition #

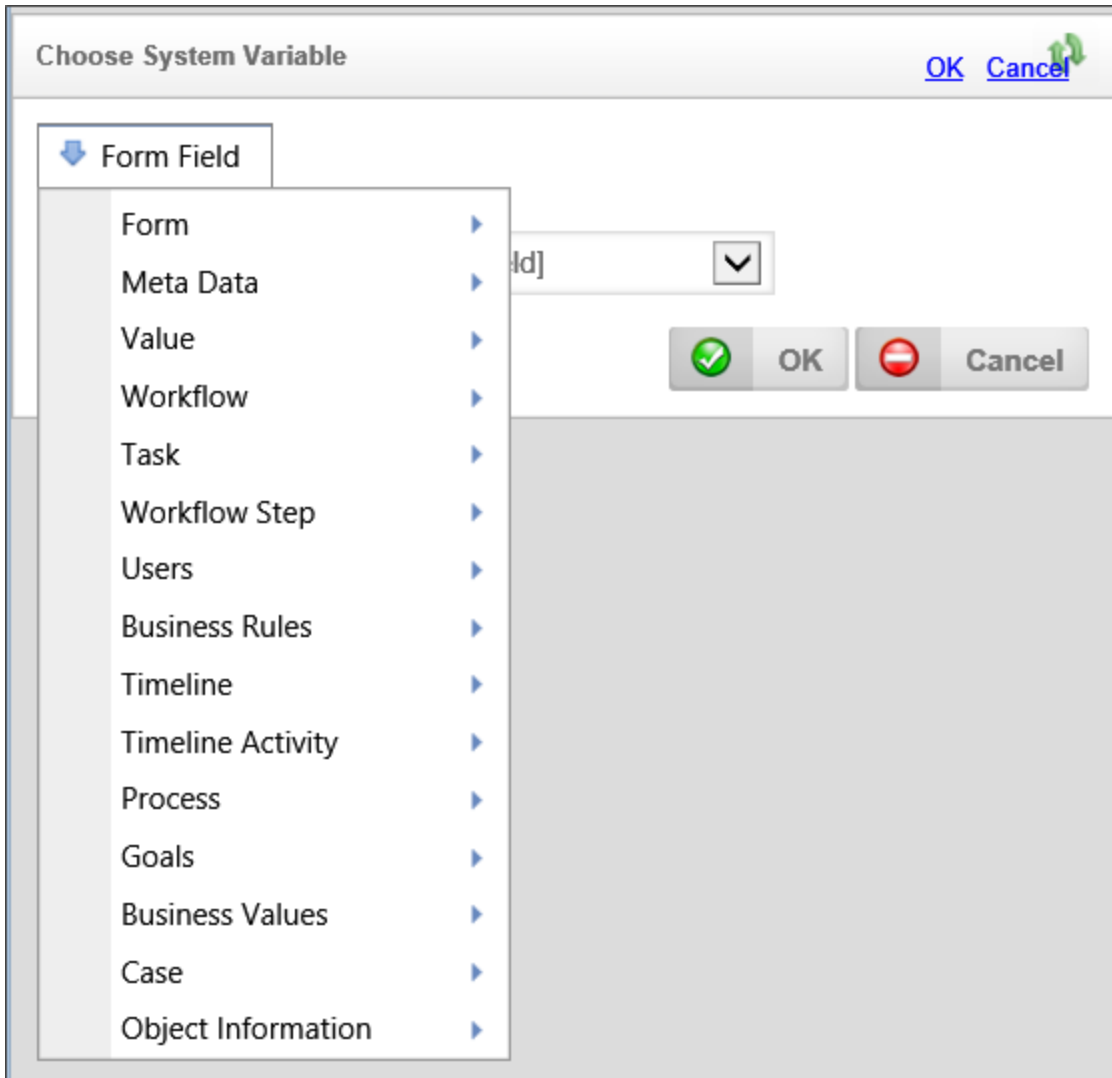
When the **Condition Builder** has been invoked, you may add a condition by selecting the **Add Condition** button .

Once you click the **Add Condition** button, the first conditions row will appear, and the first control in the row (we refer to this as the “left-hand side” of the condition) is a **Picker** control that will display the text, “Form Field” in most contexts. You'll use this **Picker** control to set the variable that you'll evaluate for the condition. The middle control is a **Dropdown** control that enables you to pick the method for comparing the system variable to a desired value, which is called the **operator**. Finally, there is, for most variable types, a third field, which is the desired value to which you wish to compare the system variable. We refer to this as the "right-hand side" of the condition.

Not all system variables will have a desired, or right hand side value. For instance, boolean variables only return true or false. As such, they aren't compared to a desired value, merely evaluated to determine if they are true or false.

### System Variable #

Click the Picker Control's **Build** button to reveal the **Choose System Variable** dialog box. This dialog box contains all of the system variables that are available for evaluation.



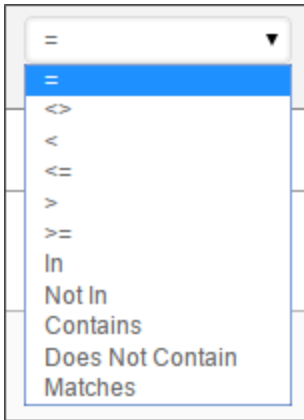
All of the system variables are organized in a dropdown menu that divides the system variables into a number of categories, as shown above. Use the dropdown menu to select the appropriate system variable. Once you've selected the system variable, click the **OK** button to close the dialog box and set the system variable in the condition row. Please see the [System Variables Reference Guide](#) for a description of all the possible system variables that may appear in the dropdown. This dropdown menu is context-sensitive, so, depending on where you are using the [Condition Builder](#), different options will appear that are relevant to the selection or filtering operation you are conducting.

**i** For process Director v5.32 and higher, the Condition Builder won't include fields that have no value, like Embedded Sections, or labels, when evaluating form fields

## Operator #

Once the variable has been selected, you must choose an operator for the variable, i.e., the method we will use to compare the variable to the desired value. A number of operators are available, and the type of

available operators will depend on the type of system variable you select. A Boolean system variable, for instance, will only allow you to select whether the variable is true or false, while a string variable will provide a much wider array of choices.



In general, the following operators are available:

### = (Equals)

This operator requires that the system variable and the desired value match *exactly*. You should only use the = operator when you're using fixed, predictable values. For example, you shouldn't use this operator when comparing dates, unless you're sure that both the date and time values will always match, to the second. Otherwise the data comparison will always return `faLse` for the date evaluation.

### <> (Does Not Equal)

This operator requires that the system variable and the desired value don't match.

A search for fields that are NOT empty can be performed using this operator with the syntax :

```
Value <> ""
```

### < (Less Than)

This operator requires that the value of the system variable be less than that of the desired value.

### <= (Less Than or Equal To)

This operator requires that the value of the system variable be less than or equal to that of the desired value.

### > (Greater Than)

This operator requires that the value of the system variable be greater than that of the desired value.

### >= (Greater Than or Equal To)

This operator requires that the value of the system variable be greater than or equal to that of the desired value.

### In

This option is primarily designed to match a value to a list of values. For instance, if the left side of the condition contains a user, and the right side of the condition contains user group, this operator will

determine if the user is contained in the group. Similarly, if the value on the left is a string and the value on the right is a comma-separated list of values, the In operator means “in that list”. If the value is determined to be in the list of values, this operator will return "true"

### Not In

This option is also primarily designed to match a value to a list of values, though, in this case, it returns "true" if the value isn't in the list.



A User filter on a Knowledge View will now support "user IN/NOT (group or user list)". If it is a "user IN group list", the filter will return True if the user is in ANY of the groups. If you have a "user NOT IN group list" filter, then the filter will return true if the user is NOT IN ANY of the groups.

### Contains

The string value of the System Variable is contained in the string value of the desired value. For instance, a desired value of "Rejected" would contain a system variable with the string value of "Reject".

### Does Not Contain

The string value of the System Variable isn't contained in the string value of the desired value.

### Matches

This operator works similarly to the Contains operator, but supports the ability to use AND and OR operators in the filter string. The filter string can also contain double quotes to force the search on that entire character string. If there is no operator between strings, or a ",", these will be treated as "AND". Additionally, the Matches operator supports a Regular Expression (REGEX) value for the desired value.

### No Earlier Than

This operator is applicable only to date/time values. It enables you to specify the lower bound of a date range.

### No Later Than

This operator is applicable only to date/time values. It enables you to specify the upper bound of a date range.

## Full-Text Search Operators <#>

For Process Director v5.21 and higher, full text search operators enable you to use nonspecific or "fuzzy" search terms to return a larger set of potential matches for a search term. Please remember that full-text searching can be resource-intensive, may take a long time, and may return an unexpectedly large set of search results, depending on the precision of your search terms. There is a custom variable, [fCONTAINSUseValueSearchOnly](#), that can speed up full-text searching, though at the cost of some search accuracy.

To implement full-text searching, you must enable it on your system by performing the following process:

- While in [debug mode](#), click the command in the troubleshooting page to create the index and catalog. There is another link to see status, and another to turn off full-text searching. Turning on this feature may take a while to complete on existing databases.

Process Director Administration					
Process Director Name	Interface URL	Server Version	Install Path	License Description	Server Address
Training Server [Non Production]		5.25	C:\Program Files\BP Logix\Process Director\website\	Process Director - Tier 3 - Non Production	::1

Process Director Control Pages: Configuration, User Administration, Installation Settings, Troubleshooting

Server Control: System Information, Impersonate, Audit Logs, Logs, Run Email Tests, Help

Process Director diagnostic and control links.

<a href="#">Reload the Process Director Settings</a>	This will reload the configuration settings and clear all caches.
<a href="#">Write Diagnostics To Logs</a>	This will dump diagnostic information to the Process Director log files.
<a href="#">Clean cache statistics</a>	This will reset all cache statistics. This will not clear the actual caches.
<a href="#">Run Delete Cleanups</a>	This will force the clean up routines to run immediatly.
<a href="#">Run Global Cleanups</a>	This will force the .NET clean up routines to run immediatly.
<a href="#">Run Timers</a>	This will force all timer routines to be run immediatly.
<a href="#">Clean Disk Cache</a>	This will remove all temporary disk cache files
<a href="#">Turn Diagnostic Logging ON</a>	This will turn on a higher level of debug logging. This may have a slight performance impact when turned on.
<a href="#">Turn Diagnostic Logging OFF</a>	Turn off the debug logging.
<a href="#">Run Basic HTML Tests</a>	This will ensure HTML can process.
<a href="#">Run Basic ASPX Tests</a>	This will ensure ASPX functions can run.
<a href="#">Run Database Diagnostic</a>	This will run the database diagnostic routines.
<a href="#">Create internal VIEWS</a>	This will ensure all internal database VIEWS are created and current.
<a href="#">Fix Old DB Column Formats</a>	Fix Old DB Column Formats
<a href="#">Set the Search Column in tblFormData</a>	For upgrades to v5.20 this will set the new sValueSearch column to 256 bytes of the sValue
<a href="#">Add FULL TEXT INDEX</a>	This will create a catalog and add a FTS index on tblFormData sValue and sDisplayString.
<a href="#">Remove FULL TEXT INDEX</a>	This will remove the catalog and FTS index from tblFormData sValue and sDisplayString.
<a href="#">FULL TEXT INDEX Status</a>	Display the population status of the full text index catalog on tblFormData
<a href="#">Recalc Step Due Dates</a>	Recalculate due dates on running steps using the step definition

There are some restrictions that you must keep in mind when using full-text search operations.

- Partial word searches must start with the beginning of the word.
- Some words may not be found if they contain special characters that the database considers to be a "stop word". For instance a word that contains a number may be ignored, even though it otherwise matches the search term. What constitutes a "stop word" depends on the system settings of your database, so Oracle and SQL Server may operate differently.

**i** You can turn the stoplist off on SQL Server by using the following SQL Command in the SQL Server Management Console:

```
ALTER FULLTEXT INDEX ON tblFormData SET STOPLIST = OFF
```

To turn the stoplist ON with the default of "system" use this SQL command:

```
ALTER FULLTEXT INDEX ON tblFormData SET STOPLIST = SYSTEM
```

Once full-text searching is enabled, the following operators will be come available when appropriate.

### Starts With

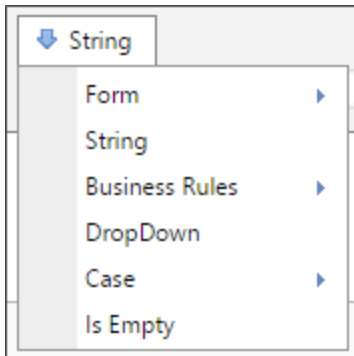
This full text search operator enables you to find text strings that begin with the specified search term. For instance, a Starts With search for "ham" will return "hammer", "ham-fisted", "Hamlet", "Hamilton", hamburger, etc.

### Contains (Lexical)

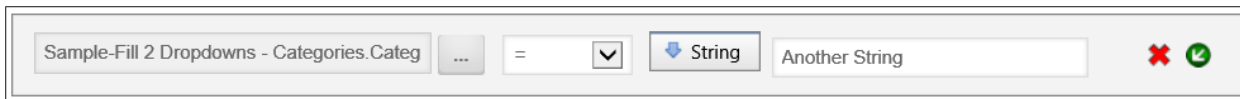
This full text search operator enables you to perform a search of text that returns terms that are lexically similar terms to the search term. For instance, a search for "check" will return items that contain the lexical matches such as "Uncheck", "Checked", "Checking", "Checker", etc.

## Desired or Comparison Value #

Once the operator has been selected, you must provide the desired value on the right hand side of the Condition Row, to which the System Variable will be compared. The desired value can be derived from Process Director, or manually entered by the user. A dropdown menu is available to select the type of desired value. Different data types will present different items in the dropdown menu, each of which is appropriate to the data type you are comparing.



Depending on which item you select from the dropdown menu, the appropriate selection or input method, if necessary, will be displayed in the [Condition Builder](#) to enable you to enter the appropriate comparison value. For instance, if you select a "String" as the desired comparison value, a text box will be provided into which you can type the string value you desire.



In the example above, the condition requires that the value of the [Sample-Fill 2 Dropdowns](#) form field equal the desired string value, "Another String".

## Managing Conditions

Once a condition has been created, you may remove a condition by selecting the red "x" icon next to the condition to be removed.



Conditions for: Default eForm

Form conditions.

Description

User Information.Name (None) = String Some String

Delete Icon

Add Condition

OK Update Cancel

You may also select the green arrow icon to insert a new condition to be included in that condition.

Conditions for: Default eForm

Form conditions.

Description

User Information.Name (None) = String Some String

[ AND ]

User Information.Email (None) = String Some Other String

Icon to add an [AND] condition

Add Condition

OK Update Cancel

You may also select the **Add Condition** button to insert another condition but to use it as an OR condition.

Once you've set your conditions for your option you can save it by clicking the **OK** or the **Update** buttons.

## Using Arrays in Form Validation Conditions #

Validation filter conditions that use Array fields on a Form can be complex, and may not work in the way you expect, because various filter conditions operate differently, depending on whether you are trying to filter based on a single (scalar) value or an array (column) value, and whether the value is used on the left or right side of the operator in the [Condition Builder](#). The various condition scenarios are described in the table below.

LEFT SIDE	RIGHT SIDE	COMPARISON BEHAVIOR	EXAMPLE						
Scalar	Scalar	<p>Simple comparison of the two values.</p> <p>For instance, if you wanted to search a Form Field named favoriteColor for the color Blue, the comparison would be:</p> <table border="1"> <thead> <tr> <th>Form Field</th> <th>Operator</th> <th>Value (String)</th> </tr> </thead> <tbody> <tr> <td>favoriteColor</td> <td>=</td> <td>"Blue"</td> </tr> </tbody> </table> <p>If the favoriteColor field contains the value "Blue" the condition will return "true".</p>	Form Field	Operator	Value (String)	favoriteColor	=	"Blue"	Field1="Value"
Form Field	Operator	Value (String)							
favoriteColor	=	"Blue"							
Column	Scalar	<p>The column value on the left side of the operator is returned as a comma-separated list, and the condition will evaluate to see if the scalar value on the right side is contained in the list. This condition will return "true" if the scalar value matches the condition for any row in the array column.</p>	Column1=Field1 Column1<=Field1						

LEFT SIDE	RIGHT SIDE	COMPARISON BEHAVIOR	EXAMPLE																		
		<p>For instance, let's assume we have an array column named primaryColors, containing the following values:</p> <p>Row [1]: red Row [2]: blue Row [3]: yellow</p> <p>Any of the following comparisons will return "true":</p> <table border="0"> <thead> <tr> <th>Form Field (Array Column)</th> <th>Operator</th> <th>Value (String)</th> </tr> </thead> <tbody> <tr> <td>primaryColors</td> <td>=</td> <td>"red"</td> </tr> <tr> <td colspan="3"><b>OR</b></td> </tr> <tr> <td>primaryColors</td> <td>=</td> <td>"blue"</td> </tr> <tr> <td colspan="3"><b>OR</b></td> </tr> <tr> <td>primaryColors</td> <td>=</td> <td>"yellow"</td> </tr> </tbody> </table>	Form Field (Array Column)	Operator	Value (String)	primaryColors	=	"red"	<b>OR</b>			primaryColors	=	"blue"	<b>OR</b>			primaryColors	=	"yellow"	
Form Field (Array Column)	Operator	Value (String)																			
primaryColors	=	"red"																			
<b>OR</b>																					
primaryColors	=	"blue"																			
<b>OR</b>																					
primaryColors	=	"yellow"																			
Scalar	Column	<p>The column value on the right side of the operator is returned as a comma-separated list, and the condition will evaluate to see if the scalar value on the left side is contained in the list. This condition will return "true" if a matching value is found.</p> <p>For instance, let's assume we have an array column named primaryColors, containing the following values:</p> <p>Row [1]: red Row [2]: blue Row [3]: yellow</p> <p>Any of the following comparisons will return "true":</p> <table border="0"> <thead> <tr> <th>Form Field (Array Column)</th> <th>Operator</th> <th>Value (String)</th> </tr> </thead> <tbody> <tr> <td>"red"</td> <td>IN</td> <td>primaryColors</td> </tr> <tr> <td colspan="3"><b>OR</b></td> </tr> <tr> <td>"blue"</td> <td>IN</td> <td>primaryColors</td> </tr> <tr> <td colspan="3"><b>OR</b></td> </tr> <tr> <td>"yellow"</td> <td>IN</td> <td>primaryColors</td> </tr> </tbody> </table>	Form Field (Array Column)	Operator	Value (String)	"red"	IN	primaryColors	<b>OR</b>			"blue"	IN	primaryColors	<b>OR</b>			"yellow"	IN	primaryColors	Field1 IN Column1
Form Field (Array Column)	Operator	Value (String)																			
"red"	IN	primaryColors																			
<b>OR</b>																					
"blue"	IN	primaryColors																			
<b>OR</b>																					
"yellow"	IN	primaryColors																			
Column	Column	<p>Each row in the column is compared with the corresponding column in the same row. This condition will return "true" if any row comparison matches the operator.</p> <p>For instance, let's assume we have two array columns, primaryColors and berryColors, containing the following values:</p>	Column1=Column2																		

LEFT SIDE	RIGHT SIDE	COMPARISON BEHAVIOR	EXAMPLE						
		<p>Array column “primaryColors”</p> <p>Row [1]: red</p> <p>Row [2]: blue</p> <p>Row [3]: yellow</p> <p>Array column “berryColors”</p> <p>Row [1]: red</p> <p>Row [2]: black</p> <p>Row [3]: blue</p> <p>The comparison condition would be:</p> <table border="0"> <tr> <td>Form Field (Array Column)</td> <td>Operator</td> <td>Form Field (Array Column)</td> </tr> <tr> <td>primaryColors</td> <td>=</td> <td>berryColors</td> </tr> </table> <p>Since Row[1] matches the condition, the condition will return true.</p>	Form Field (Array Column)	Operator	Form Field (Array Column)	primaryColors	=	berryColors	
Form Field (Array Column)	Operator	Form Field (Array Column)							
primaryColors	=	berryColors							

## Client Validation #

In Process Director v5.1 and higher, a client side validation function allows conditional processing on form fields to occur in the client browser without having to make the form fields into event fields, and reposting the form when the fields change.

Conditions for: Sectionembedded4

If the following conditions evaluate to true, then this form control will be DISABLED or ENABLED based on the form field properties.

Description

Do not allow evaluation of Condition on the Client-side (default)
  Allow evaluation of Condition on the Client-side

chkDisableSection Checked ▼
✕
↻

Add Condition

OK
Update
Cancel

These conditions are used to control the display/hidden “when”, enabled/disabled “when”, and required/not required “when” settings in the properties displayed from the **Form Controls** tab of the form definition.

When this client side condition flag is enabled, system variables will only be evaluated once on the server when the page is displayed or an event occurs. Form fields, however, will always use to most current form data on the visible form when evaluating the condition, because the evaluation will take place in the client browser, and will **not** require the form to be reposted to the server to evaluate the form fields.

By default, the traditional server-side evaluation method will be the default evaluation method for conditions, unless you specifically choose to use client validation.

As of Process Director v5.1, only the following control types can be used for client-side conditional processing, both for Showing/Hiding and for the querying of live data:

- Input/Text
- Checkbox
- Date
- Dropdown
- Buttons
- Sections

## Date Comparison Issues #

First of all, using the "=" operator is usually a bad idea for date comparisons. Date fields—even those where you don't explicitly set a time—**always** have a time value. The database default time for DateTime fields is always precisely 12AM, e.g., `1 Jan 2023 00:00:00`. Obviously, if you use a **DateTime Picker** control, the time element will be whatever is explicitly set. In either case, the date will always have a time value as part of the data. Since the "=" operator always requires that two values match **exactly**, if the two time elements are even one second off when the evaluation occurs, a date comparison that uses the "=" operator will always return `false`. The "=" operator can only return `true` at a single second in time. Date comparisons should use either the "<" or ">" operators.

Next, while comparing two known dates is relatively simple, date comparisons in conditions can use a variety of date calculations such as "Days Ago", "Weeks Ago", etc., which also enable you to specify the number of days or weeks. Making these types of date comparisons in conditions are slightly more complicated than one might think.

Let's say you wish to create a condition to see if the start date of a Process Timeline activity was more than 2 days ago. To properly set this condition, your condition would look like:

```
(Activity Start Date [ActivityName] < 2 Days Ago)
```

Note that this condition uses the Less Than (<) operator and **not** the Greater Than (>) operator. Why is this, if I'm looking to see if the Activity start date is "greater" than 2 days ago? Well, it's because the `2 Days Ago` element doesn't take the Activity Start Date and calculate to see if it's now more than two days later. Instead, the `2 Days Ago` element independently returns whatever the datetime value was exactly two days before the evaluation.

You're not really asking if the Activity Start Date was more than 2 days ago. What you're actually asking is whether the Activity Start date is **earlier** than whatever date is returned by the `2 Days Ago` element.

For example, let's say the Activity Start Date was `1 Jan 2023 09:00:00`. If you were to evaluate the `2 Days Ago` element at exactly the same time the Activity started, it would return `29 Dec 2022 09:00:00` as the date.

`1 Jan 2023 09:00:00` is **later** (>) than `29 Dec 2022 09:00:00`. Since that is so, the condition will evaluate as `false`, which is what we want.

On the other hand, if the `2 Days Ago` evaluation is performed at 9:01AM on 3 Jan 2023, the `2 Days Ago` element will return `1 Jan 2023 09:01:00`. The Activity Start Date of `1 Jan 2023 09:00:00` is now one

minute earlier (<) than 1 Jan 2023 09:01:00. Our condition will evaluate as true, therefore, on at any time after 9AM on 3 Jan 2023, as the Activity Start Date will now always be earlier (<) than the date returned by the 2 Days Ago element of the condition.

## Debug Mode

Process Director can function in a troubleshooting mode, called Debug Mode, that is useful for both administrators and implementers. In Debug Mode, Process Director's user interface shows additional information that is useful for troubleshooting issues. For example, the standard view of the [Content List](#) in Process Director looks like the example below:

/Default Partition						
<input type="checkbox"/>	Name ▲	Updated By	Create Date	Update Date	Size	Actions
<input type="checkbox"/>	[Business Rules]	Dale Franks	9/29/2016	9/29/2016	-	
<input type="checkbox"/>	[Custom Tasks] This folder contains all the custom tasks. Do not delete this folder. When importing new custom tasks, import them from the root folder, updating the contents of this folder with the new tasks.	Dale Franks	11/3/2014	12/14/2016	-	
<input type="checkbox"/>	[DataSources]	Dale Franks	7/30/2015	7/30/2015	-	
<input type="checkbox"/>	[Samples]	Dale Franks	4/4/2016	4/4/2016	-	
<input type="checkbox"/>	[Templates] This folder contains product templates for getting started.	Dale Franks	4/13/2016	8/16/2016	-	
<input type="checkbox"/>	Customer Tech Support Tickets Area for creating test projects for Support Tickets.	Dale Franks	10/31/2014	6/17/2016	-	
<input type="checkbox"/>	Documentation Area for creating objects for documentation/screenshots.	Dale Franks	10/31/2014	2/3/2015	-	
<input type="checkbox"/>	Miscellaneous Testing Folder for testing issues that do not arise from Tech Support Tickets	Dale Franks	6/17/2016	6/17/2016	-	
<input type="checkbox"/>	New Projects Area for creating new projects for demos, etc.	Dale Franks	10/31/2014	10/31/2014	-	
<input type="checkbox"/>	Training	Dale Franks	2/23/2015	2/23/2015	-	
<input type="checkbox"/>	webinar	Dale Franks	11/2/2016	11/2/2016	-	

In Debug Mode, however, the [Content List](#) shows additional information:

/Default Partition							
<input type="checkbox"/>	Link	Name ▲	Updated By	Create Date	Update Date	Size	Actions
<input type="checkbox"/>		[Business Rules]	Dale Franks (UserID:dale, UID: ddfcdd5e-2a82-4786-a26f-7b010fa8b886, ADID: 4c851547-f4af-4aaa-b0ac-51c502088863)	2016-09-29 11:58:52.640	2016-09-29 11:58:52.640	-	
<input type="checkbox"/>		[Custom Tasks] This folder contains all the custom tasks. Do not delete this folder. When importing new custom tasks, import them from the root folder, updating the contents of this folder with the new tasks.	Dale Franks (UserID:dale, UID: ddfcdd5e-2a82-4786-a26f-7b010fa8b886, ADID: 4c851547-f4af-4aaa-b0ac-51c502088863)	2014-11-03 17:09:01.827	2016-12-14 11:35:22.147	-	
<input type="checkbox"/>		[DataSources]	Dale Franks (UserID:dale, UID: ddfcdd5e-2a82-4786-a26f-7b010fa8b886, ADID: 4c851547-f4af-4aaa-b0ac-51c502088863)	2015-07-30 13:57:42.587	2015-07-30 13:57:42.587	-	
<input type="checkbox"/>		[Samples]	Dale Franks (UserID:dale, UID: ddfcdd5e-2a82-4786-a26f-7b010fa8b886, ADID: 4c851547-f4af-4aaa-b0ac-51c502088863)	2016-04-04 16:42:50.767	2016-04-04 16:42:50.767	-	
<input type="checkbox"/>		[Templates] This folder contains product templates for getting started.	Dale Franks (UserID:dale, UID: ddfcdd5e-2a82-4786-a26f-7b010fa8b886, ADID: 4c851547-f4af-4aaa-b0ac-51c502088863)	2016-04-13 11:46:38.183	2016-08-16 11:46:38.183	-	

This enhanced view appears throughout the interface, including Forms, showing additional information about every object that is often useful in finding errors or other problems. For example, the image below is a portion of a running Form viewed in Debug Mode.

The screenshot shows a portion of a form in Debug Mode. At the top, there is a field labeled "Number:" followed by a blank input field. Below it is a field labeled "Submitted By:" with the text "Dale Franks (UserID:dale, UID: ddfcdd5e-2a82-4786-a26f-7b010fa8b886, ADID: 4c851547-f)".

Below the "Submitted By:" field, there is a red callout box with the text "Debug Information for the User" pointing to the user information in the "Submitted By:" field.

Further down, there is a form section with several fields: "Vendor Contact:", "Address:", "City:", "State:", and "Zip". The "Address:" field is highlighted with a yellow tooltip that says "Enter the name of the Vendor" and contains the text "VendorName". Below the "Address:" field, there is a field labeled "FCID:" with the text "FCID: 2d2f4d32-a37e-4b51-95b8-b8f35aebc50c".

Below the "FCID:" field, there is a red callout box with the text "Field Debug Information" pointing to the "FCID:" field.

The "State:" field is a dropdown menu with "Alabama" selected. The "Zip" field is a text input field.

Note that even the disabled field at the top that contains a user's name shows debug information like the UserID and UID for the user. Additionally, placing your mouse over the Vendor name field in this example displays debug information in the field's tooltip, such as the actual field name and FCID. (The UID and FCID are GUIDs generated by the Process Director database to uniquely identify the objects.)

Debug mode can also provide extra options that aren't available in normal mode. For instance, an option exists in Debug Mode to restart all Process Timelines in an error state. This is done by entering Debug Mode in a Knowledge View that returns Process Timeline instances, and clicking on the checkbox to select all processes. A new action item will appear that says "Restart Error Workflows". Additionally, when viewing properties for Form or case instances in Debug Mode, you can edit the data directly in the instance properties, although no validation or error-checking will be performed on data edited in this way, so use caution when doing so.

Using Debug Mode, therefore, should be one of your first steps in troubleshooting an issue.

Turning on Debug Mode is simple. Depending on which version of Process Director you're using, you can use one of two methods:

- **Process Director v 6.1 and higher:** In the [User Profile menu](#), click the option labeled "Debug Mode".
- **All older versions:** Place the mouse pointer in the workspace tab area at the top of the page, and click the right mouse button. This will place the page in Debug mode, and a dialog box will appear notifying you that the mode has been changed. Click the OK button to close the dialog box.

To turn Debug Mode off, place the mouse pointer in the workspace tab and right-click again. A dialog box will once again appear, notifying you that Debug Mode has been turned off.

## Desktop Interface Workspace

Users of Process Director v5.23 and higher have an optional user interface for the traditional Workspace used in all prior versions. This new user interface, which is called the Desktop Interface, provides a more customizable interface. Unlike the traditional Workspace, which can't be customized at the user level, the Desktop Interface offers end users the ability to customize the available interface objects in a number of ways. The Desktop Interface is an enhancement for the traditional Workspace. Using the Desktop interface is optional for each Workspace, and both traditional and Desktop Workspaces can be used in the same installation.

Let's compare the traditional Workspace, with its array of tabs and specified workspace portlets, with the Desktop Interface. The traditional Workspace appears similar to the Workspace below.

The screenshot displays the BP Logix Desktop Interface workspace. On the left, there is a navigation pane with a search bar and a list of portlets including 'Alert Icon Issue', 'Application Events Sample', 'Array', 'Array Form', 'Array Sample Form', 'Array Validation', 'ASP Controls', 'Auto Claim', 'InTest', 'Child Form', 'Complete', and 'Contract Review Request'. Below this is a 'Task List (3 tasks)' table:

Name	Task	Priority	Assigned On	Due Date
Invoice Processing Submitted On 7/10/2019 11:03 AM	Process Invoice	1	7/10/2019	7/11/2019
Invoice Processing Submitted On 7/10/2019 11:03 AM	Process Invoice	1	7/10/2019	7/11/2019
Invoice Processing Submitted On 7/10/2019 11:03 AM	Process Invoice	1	7/10/2019	7/11/2019

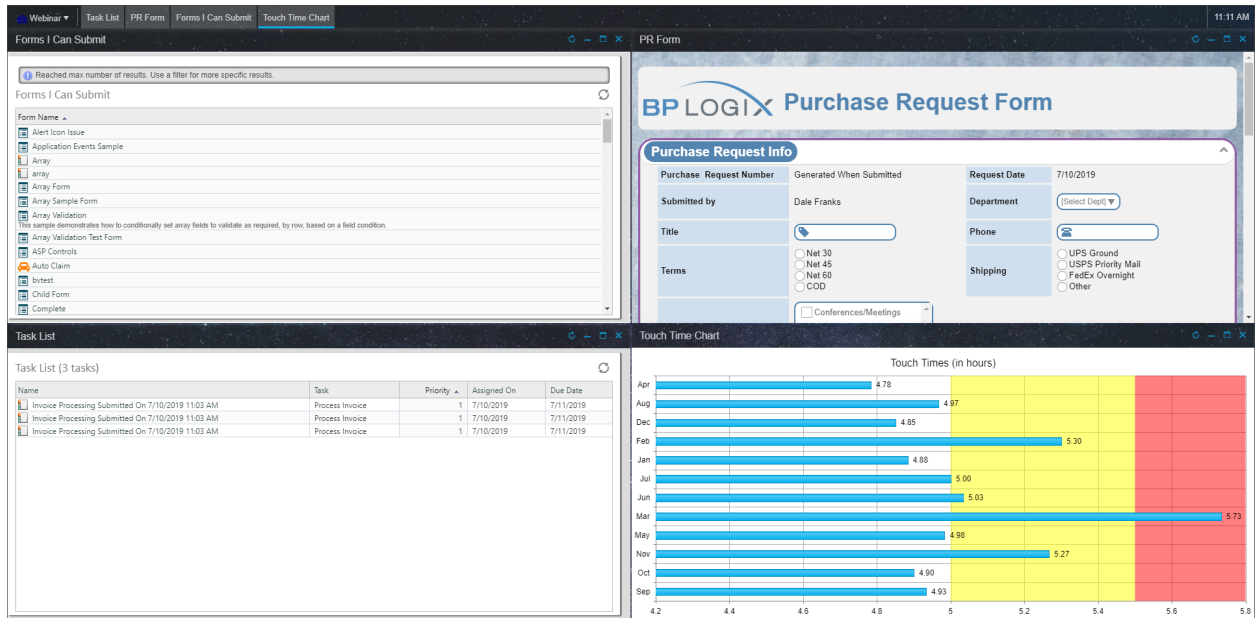
The main workspace area features a 'Purchase Request Form' with the following fields and options:

- Purchase Request Number:** Generated When Submitted
- Request Date:** 7/10/2019
- Submitted by:** Dale Franks
- Department:** (Select Dept) [v]
- Title:** [Text Input]
- Phone:** [Text Input]
- Terms:** Radio buttons for Net 30, Net 45, Net 60, and COD.
- Shipping:** Radio buttons for UPS Ground, USPS Priority Mail, FedEx Overnight, and Other.
- Additional options:** Checkboxes for Conferences/Meetings and Equipment.

At the bottom right, there is a bar chart titled 'Issues by Team' showing data for various teams (0.405 to 0.523) with two series, A and B.

User can temporarily change the size of the individual portlets, but these changes aren't permanent. Other than that, there are no user-level customization options. The Desktop Interface, on the other hand, has a different visual appearance, and customizations are cached as user preferences, and can, in fact, be saved permanently as the default configuration. With that in mind, the same Workspace used in the example above will display using the Desktop Interface as shown below.

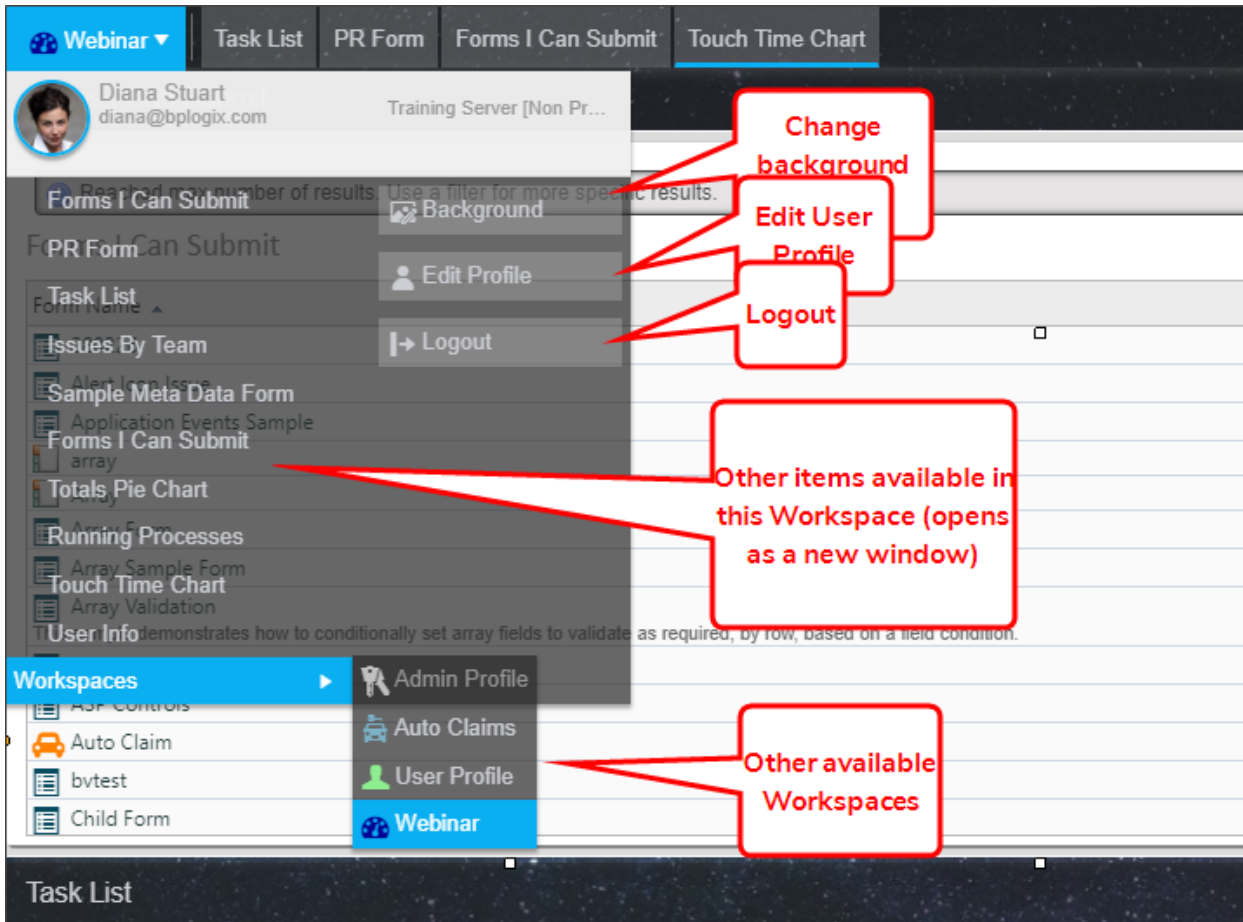




A few immediate differences are apparent. The portlets themselves are no longer displayed in sizable frames, but rather as independent windows, each of which can be resized, minimized, maximized, and moved about on the screen. The window positioning and size is automatically saved whenever the user makes a change, so any changes the end user makes to the window displays are persistent.

At the top of the screen, instead of a row of tabs for different workspaces, a row of tabs listing the visible windows is displayed, which can be used to select the desired window, as well as manipulate the window, if necessary. For instance, if you minimize a window, you can restore it by clicking its tab at the top of the screen. The Workspace tabs have been replaced in the Desktop by a dropdown navigation menu in the upper left corner of the screen.

The navigation dropdown menu offers a number of options.

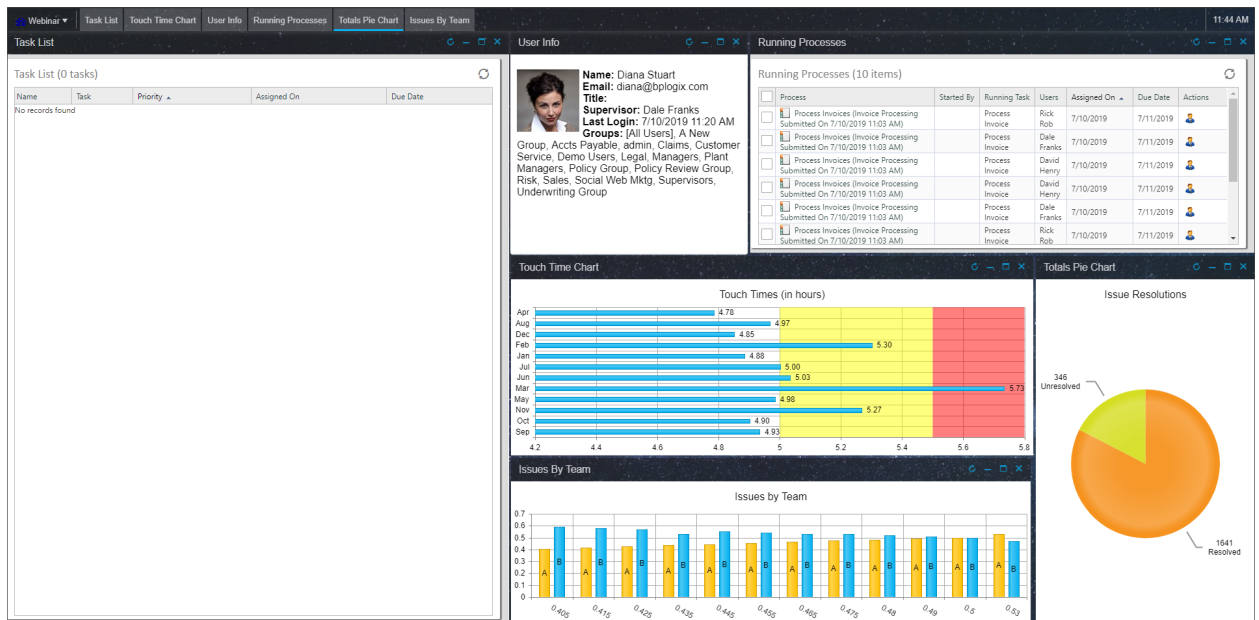


The **Background** button can be used to change the background image that was initially configured for this Workspace in the Workspace configuration that was implemented by the administrator.

The **User Info** slideout of the traditional Workspace has been replaced by an **Edit Profile** button that will open the user profile as a new window.

The **Logout** button now appears automatically, and no longer needs to be specifically configured in the Desktop Interface, as it did in the traditional Workspace.

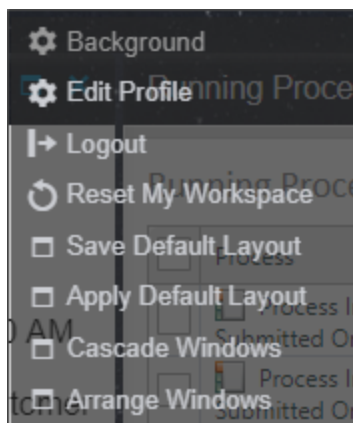
Navigating to another workspace item will open the item as a new Window in the Desktop Interface, and, when the window opens, will display a new tab for the new window at the top of the screen. The new window can be sized just like any other window, and, in fact, placed persistently in the workspace. Every portlet or navigation button configured for the Workspace will show up in the navigation menu, and added to the workspace display by the user. Using this sample Workspace, the end user can customize this Workspace to appear like the example below.



Users don't have access to any item that hasn't been configured as part of the Workspace, but they can customize how they wish to display the items that have been made available to them. This enables users to, within the limits defined by administrators, create a Workspace UI that they feel enhances their user experience.

Finally, at the bottom of the navigation dropdown, users can navigate to other workspaces.

An additional popup menu appears with you right-click in the tab bar at the top of the screen.



In addition to the **Background**, **Edit Profile**, and **Logout** buttons, additional configuration options are available.

The **Reset My Workspace** button will remove any recent changes the user has made to the workspace, and revert the Workspace configuration to the last saved appearance.

The **Save Default Layout** button is available only to administrators, and will save the current layout configuration as the default configuration for the Workspace. Initially, the saved default is the configuration that was specified when the workspace was created. Saving a layout will replace the initial default with the new, saved configuration.

The **Apply Default Layout** button is available only to administrators, and will revert to the default layout.

The **Cascade Windows** button will arrange all of the visible windows in a standard Window cascade.

The **Arrange Windows** button will rearrange the existing windows in equal-sized windows.

## Process Director Content List Objects

Every application you build in Process Director contains one or more [Content List](#) objects, also called Process Director objects, that define how the application will work. Each object performs a specific function in the application. A [Form](#), for example, is the object that users access to enter information in electronic format, which the application will then process in some way. The [Process Timeline](#) models the process and defines the route the Form will take during each iteration of the application's life, from Form submission until the application has reached the end of the process. In general, a functional application needs at least one Form and one [Process Timeline](#) to perform a useful task. The Form stores the information needed by the application, and the Process Timeline routes the information to users who need to see and make decisions based on the information.

Most applications, of course, are more complex than this, and require other objects to function properly. An application may need to use a [Business Value](#) to extract information from an external data source, or a [Business Rule](#) to implement some specialized business logic. Applications that model business processes must often implement complex business logic or sophisticated information processing and evaluation. Thus, they may require many objects to function as desired. Each Process Director object encapsulates a specific function or feature, and enables you to create modular applications—often quite complex ones—without requiring you to write customized programming code. As an implementer, you add and configure the objects to your application to provide the desired features.

This section of the documentation lists, in alphabetical order, all of the [Content List](#) objects that are available to you. You can navigate to the documentation topic for each object using the Table of Contents displayed on the upper right section of this page, or by using one of the links below:

[Business Rule](#): Enables you to incorporate business logic to return a true/false value, or other custom values, as the result of evaluating a condition.

[Business Value](#): Enables you to use virtualized data anywhere in Process Director.

[Case Definition](#): Enables you to implement Case Management applications.

[Chart](#): Enables you to display infographics in your application.

[Dashboard](#): Enables you to display Process Director objects in a unified screen, to organize what the end users see.

[Datasource](#): Enables you to create connections to external databases, for extracting data, which is usually presented to end users via a Business Value.

[Dropdown List Object](#): A list object that stores options to present in a [Dropdown](#) control on a Form. This object is re-usable and can be referenced by any [Dropdown](#) control on any form to provide the selectable options that will be visible to the end user.

[Form](#): The primary data collection and storage object for an application.

[Goal](#): An object that evaluates a condition, sets a system state, and optionally starts a specified process on a scheduled basis.

[Knowledge View](#): The primary object for extracting and reporting on data in Process Director.

**Machine Learning Object:** A specialized data evaluation object for using complex statistical algorithms to evaluate large data sets and return a predictive result. Effective use of this object requires some academic grounding or experience in data science to use correctly.

**Meta Data:** A taxonomy system of categories and attributes that can be applied to most Process Director objects.

**Process Timeline:** The primary method of modeling the process an application will implement. The Process Timeline specifies the activities that will occur, their order, participants, and results.

**Report:** Using a sophisticated report creation component, this object enables you to create complex reports that can't be effectively presented through a Knowledge View.

**Stream Action:** This object can retrieve a dataset/recordset, and begin a process for each item/row of the recordset on a scheduled basis.

**Workflow (Legacy):** The original process model used by Process Director. This process model has been deprecated by the Process Timeline, and remains in the product **solely** for backwards compatibility. BP Logix recommends that you use the Process Timeline for all applications. The Workflow does **not** receive any functionality updates beyond necessary bug fixes, if any.

## Business Rules

Process Director incorporates a Business Rules engine that empowers users to rapidly implement complex business processes. Business Rules are reusable objects that can be embedded within your Process Timeline and/or Form to conditionally control how they should run and behave in various conditions. Business Rules can be defined as simple conditions, or as sophisticated, interrelated rule sets.


Business Rules can be used to return the result of a single condition or a complex condition set which enables you to incorporate sophisticated business logic into the operation of Form or process definitions. Business Rules can also return a list of users, groups, string values, a single value, or run a Custom Task, based on conditions that you configure.

A common use of Business Rules is to evaluate data from a Form field, then return a value based on the Form field value. When used in this way, Business Rules will use the current form data to make dynamic form decisions based on the current value of the field, rather than the value stored in the database for the Form Instance.

## Business Rule Configuration

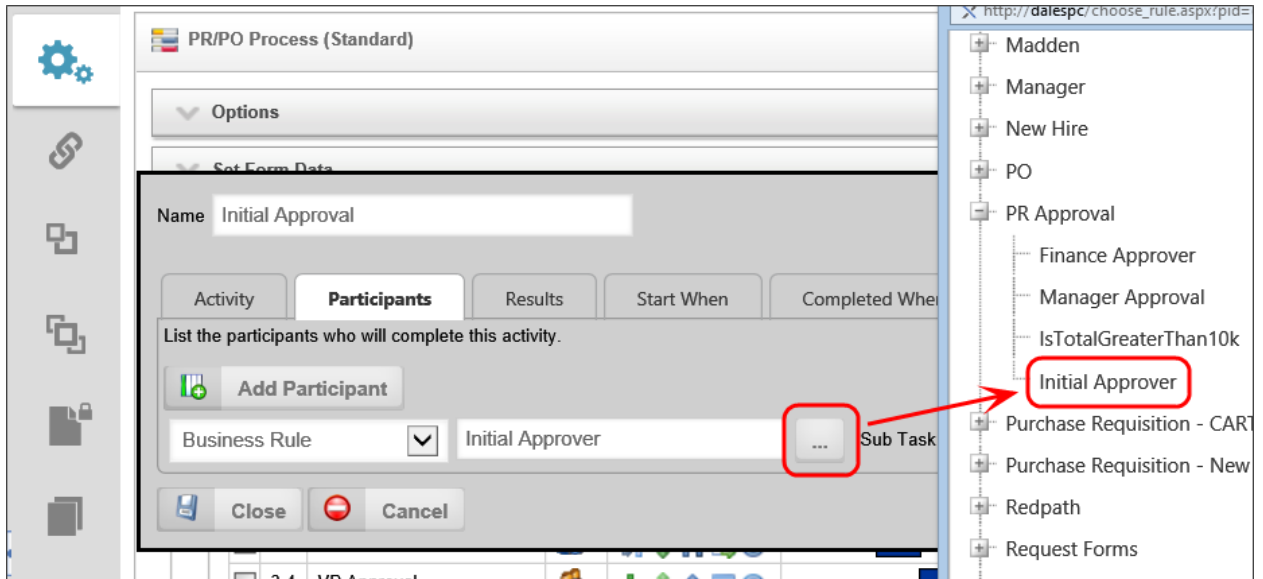
The following properties are available for configuring Business Rules.

### Business Rule Name and Description

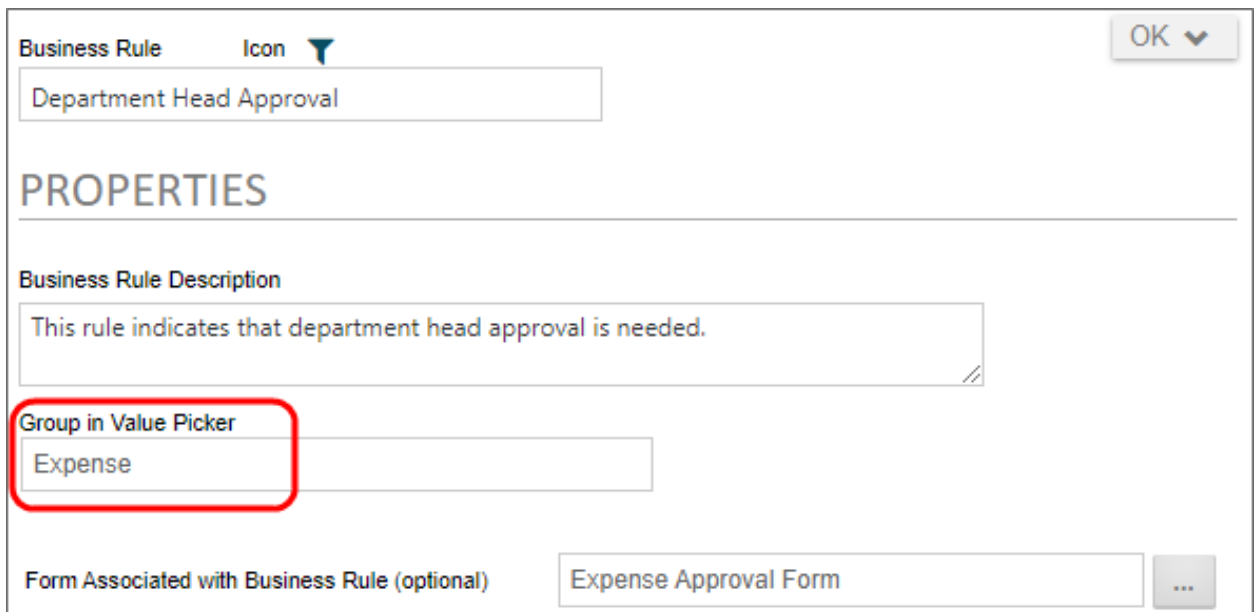
The Business Rule name and description identify and describe the object. A custom icon can also be configured for this Business Rule. To change the icon used for this Business Rule, click on the  icon.

### Group in Value Picker

Specify a group to organize Business Rule names. When selecting a Business Rule, it'll be categorized under the groups named in this field. This becomes useful when there are many Business Rules to choose from. Business Rules will be displayed alphabetically in all pickers, showing the Groups first, then all those Business Rules without Group names.



For example, the **Group** in the **Value Picker** for the **Dept Head Approval** Activity is named “Expense” is displayed under the selection category “Expense” whenever a Business Rule option is presented.



### Forms Associated with Business Rule

You can optionally link a Business Rule to a specific Form definition in the **Content List**. When this is configured you can choose a form field from this Form from a dropdown list instead of selecting the Form in every condition. Essentially, this makes the selected Form the default Form for the condition selections.

This is convenient if you have only one Form that needs to be associated with the Business Rule.

### Workflows Associated with Business Rule

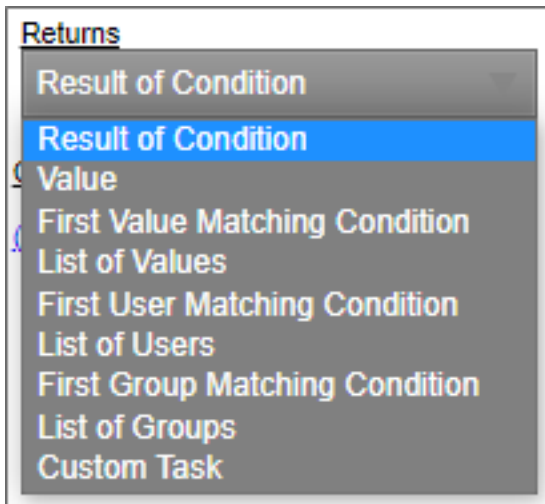
You can optionally link a Business Rule to a specific Workflow definition in the [Content List](#). When this is configured you can choose a step name from a dropdown list instead of typing in the step name when configuring the return result of a Business Rule.

### Process Timeline Associated with Business Rule

You can optionally link a Business Rule to a specific Process Timeline Definition in the [Content List](#). When this is configured you can choose an Activity from a dropdown list instead of typing in the Activity when configuring the return result of a Business Rule.

### Returns

When a Business Rule is evaluated a result is returned. There are many types of returns that can be used in your Business Rule.



The following is a list of return types and their description.

#### ***Result of Condition***

This return type will return the result of a condition. If selected, you can create a specific condition using the [Condition Builder](#) and it will return the result of that condition. For example: If a specific checkbox is checked the result would be true.

#### ***Value***

This return type will return a value. If selected, you can select from the [Select the Value to Return](#) dropdown. This allows you to select a pre-defined value in the system or you can enter your own value. Optionally you can create a condition to only use that value if the condition is met.



**Returns**

Value ▼

**Select the Value to Return** **Condition**

String  (BR State = CA)

### First Value Matching Condition

This return type will return a single value, based on the evaluation of conditions in the Business Rule. If selected, you can select from the **Select the Value to Return** dropdown. This allows you to select a pre-defined value in the system or you can enter your own value. Optionally you can create a condition to only use that value if the condition is met.

**Returns**

First Value Matching Condition ▼

This returns the value for the first matching condition only. If no conditions are met, an empty value is returned.

Values	Condition
West ...	(BR State = CA) ↑ ↓ ✖
South ...	(BR State = TX) ↑ ↓ ✖

### List of Values

This return type will return a list of string values in a comma separated list. If selected, you can add an item to the list by clicking on the **Add Value** button. This will add a value to the list. Optionally you can conditionally add that value by adding a condition to each value added to the list.

**Returns**

List of Values ▼

This returns the list of all values where the optional condition is met. The values are returned as a comma-separated list.

Values	Condition
Financial ...	(User List In Finance, Financial Aid) ↑ ↓ ✖
Information Technology ...	(User List In IT Approvers, admin) ↑ ↓ ✖

### First User Matching Condition

The Business Value will return a single user, based on the evaluation of conditions in the Business Rule. If more than one user meets the specified conditions, Process Director will return only the first user that meets the condition. The user can be manually identified in the Business Rule, or, from a field on a Form.

Prior to v3.75, Process Director required that the user field in the Form be a User Picker control. With v3.75 and higher, users can be identified from a text field that contains the UID for the user.

### List of Users

This return type will return a list of users in a comma separated list. If selected, you can add a user to the list by clicking on the **Add Users** button. This will add a user to the list. Optionally you can conditionally add that user by adding a condition to each user(s) added to the list.

Returns  
List of Users ▼

This returns the list of all Users where the optional condition is met. The Users are returned as a comma-separated list.

<u>Users</u>		<u>Condition</u>	
Diana Stuart	...	<a href="#">(Single User In admin)</a>	↑ ↓ ✖
Barb Stanley	...	<a href="#">(Single User In Finance)</a>	↑ ↓ ✖

### First Group Matching Condition

The Business Value will return a single user group, based on the evaluation of conditions in the Business Rule. If more than one user group meets the specified conditions, Process Director will return only the first user group that meets the condition.

### List of Groups

This return type will return a list of groups in a comma separated list. If selected, you can add a group to the list by clicking on the **Add Groups** button. This will add a group to the list. Optionally you can conditionally add that group by adding a condition to each group(s) added to the list.

Returns  
List of Groups ▼

This returns the list of all Groups where the optional condition is met. The Groups are returned as a comma-separated list.

<u>Groups</u>		<u>Condition</u>	
Finance	...	<a href="#">(Current User In admin)</a>	↑ ↓ ✖
Express Finance	...	<a href="#">(Current User In Clerks)</a>	↑ ↓ ✖

### Custom Task

This return type will return the result of a Custom Task. If selected, you can add a Custom Task by selecting one from the pick list and then clicking on the **Add Custom Task** button. Once you've added the Custom Task you can configure the Custom Task by clicking on the **Configure** button. Optionally you can conditionally return the result by setting a condition.

The screenshot shows a configuration panel for a Custom Task. At the top, there is a 'Returns' dropdown menu currently set to 'Custom Task'. Below it, a 'returning' dropdown menu is set to 'String'. Underneath, there are three sections: 'Custom Form Task' with the text 'Rule Query On DB', 'Options' with a 'Configure' button (which has a red 'X' next to it), and 'Condition' with the text '(BR State = CA)'.

### Custom Business Rule Parameters #

The screenshot shows the 'Parameters' section of a Business Rule definition. It has a header 'Parameters (1 Parameter)' with an expand/collapse arrow. Below the header is a table with one column labeled 'Name'. The table contains one row with the value 'Value'. To the right of the 'Value' entry is a red 'X' icon. At the bottom of the section is an 'Add Parameter' button with a green plus icon.

The Parameters section of the Business Rule definition enables you to create one or more custom parameters. When a custom parameter is configured, its value can be passed to the Business Rule via a System Variable.


**i** Business Rule Parameters can ONLY be passed via System Variable. There is no method in the UI to pass a custom parameter to a Business Rule, as you would for a Business Value.


You can add the parameter names to be used in the system variable by clicking the **Add Parameter** button in the **Parameters** section, then supplying the appropriate **Name** for each Parameter.

Parameters, once created, can be used in the conditions evaluated by the Business Rule, to return the appropriate values.

Parameters (1 Parameter)

Name













Value 


 Add Parameter

Returns

First Value Matching Condition ▼

This returns the value for the first matching condition only. If no conditions are met, an empty value is returned.

Values	Condition
Hello 	<a href="#">.({\$PARAM:Value} = hi)</a>   
Goodbye 	<a href="#">.({\$PARAM:Value} = bye)</a>   
	<a href="#">.({PARAM:Value} = )</a>   

 Add Value

In a Business Rule System Variable, you can use the following syntax to both call the Business Rule and simultaneously pass the Parameter(s):

```
{RULE:BusinessRuleName,$ParameterName1=ParameterValue1,$ParameterName2= ParameterValue2}
```

The "\$" signifier is required to specify the name of the Parameter(s).

Business Rules can be directly associated with a Process Director object, and parameters are not generally necessary for accessing data from the associated object. For example, if a Business Rule is associated with a Form definition, all the Form fields from the associated Form can be accessed from the Business Rule directly, without needing to use a custom parameter. The drawback to this is, of course, that the Business Rule can only be used in the context of a running instance of the associated Form.

On the other hand, if you do not want the Business Rule to be associated with a specific object, and want it to be globally accessible, you could create a custom parameter that could be passed to the Business Rule by any object. The custom parameter value could then be used in all of the Business Rule conditions, as the example screenshot above demonstrates. As long as the object can pass the appropriate parameter value to the Business Rule, any object in the system can call the Business rule to return the correct value via System Variable.

## Global vs Local Business Rules

Business Rules are somewhat unique in that, unlike other objects, they can be used both locally and globally within Process Director. A local Business Rule can only be used in the context of a specific Form or

Process Timeline. They cannot be called or return a useful value outside the context of a specific running application. For instance, a Business Rule that returns a value based on an evaluation of a specific Form control is almost always a local Business Rule. Indeed, in that case, the **Forms Associated with Business Rule** property will be set to create an explicit association between the Business Rule and a Specific Form definition.

On the other hand, a global Business Rule can be called by any running application. None of the **Xxxxx Associated with Business Rule** properties on the **Properties** tab would be set, and the Business Rule would return a valid value no matter what object called it for an evaluation. For example, a Business Rule that returns a User or Group for task assignment in a Process Timeline activity would always return the same User or Group, no matter which Process Timeline prompts it.

So, when deciding where to create and store a Business Rule, one should always consider whether the Business Rule will be used locally or globally. If locally, the Business Rule should reside somewhere in the top-level folder for the application that uses it, such as a **Business Rules** subfolder. A global Business Rule should be stored in a central location, such as a **[Business Rules]** folder at the root level of the partition. (The square brackets in the name are intentional.) Just remember that if you create a global Business Rule on a development system, and use it in an application you're building, you'll need to export the Business Rule into the same location on your production server *prior to* importing the application itself.

## Business Rule Samples

Your installation of Process Director should come with sample Business Rules and accompanying Forms and processes in the **[Samples]** folder. If you don't have this folder, you can contact [BP Logix technical support](#).

## Business Values

Process Director enables implementers to retrieve and manipulate data from external sources, such as HR, ERP, or CRM systems, without requiring implementers "to know technical details about the data, such as how it is formatted or where it is physically located", a technique known as Data Virtualization<sup>\*1</sup>. One such method of Data Virtualization, for users of Process Director v3.49 and higher, is through the use of Business Values. Using Business Values, Process Director can take data from external systems of record and make it available to implementers for use in managing or directing processes. Business Values, therefore, enable you to drive your applications with real-time data. They are updated whenever they are used; that is, each time a form or process references a specific Business Value, Process Director retrieves the associated external value.

At any given moment there may be hundreds, thousands, even millions of processes underway in your organization, each of them depending on business data. For those processes—many of which operate in the form of custom business applications—to succeed, the information they access must be up-to-date and accurate. Such information may be generated:

---

<sup>1</sup>Data virtualization, 2015, July 6, Wikipedia, The Free Encyclopedia. Retrieved August 17, 2015

- In the course of the process itself, like data a customer enters on a Form, or
- From the organization's **Systems of Record** (such as an ERP or CRM system). The data stored in the Systems of Record thus become a vital component of the applications that facilitate all of the organization's business activities.

Applications that use data from these Systems of Record to interface directly with processes are often referred to as **Systems of Engagement**, because they enable employees, customers, and others to engage directly in the business at hand.

It's clear, then, that Systems of Engagement, such as business applications, need a way to exchange information with Systems of Record, to make data from both systems accessible to process participants. And yet, if your business applications are being built by citizen developers, it may not be easy for those individuals to access and manipulate data from your Systems of Record. After all, Systems of Record, as well as other databases and external data sources like web services, are complex. Extracting data from them often requires detailed technical knowledge of query mechanisms, APIs, or programming languages.

Process Director addresses these issues by separating the details of accessing systems of record from using the resulting data within a business application. As a result, technical experts can configure the appropriate access mechanism once, and implementers can reuse the information provided anywhere at all, within any application, without having to know or understand how the information was obtained. An implementer may also, on occasion, create a Business Value that references only internal data from Process Director, such as a calculation based on a Knowledge View. In either case, once configured, a Business Value is universally available to implementers.

A fundamental advantage of using a Business Value is that any changes to how the Business Value is derived are transparent—and largely irrelevant—to the implementer. If the organization changes to a different CRM or HR system the Business Value can be reconfigured to use data from the new system without implementers having to make changes to existing processes.

A Business Value can extract a single value, or a list of values such as those in a database table. In the context of a Form, for example, a Business Value can be used to fill out a form Array with a recordset from an external database.



As a default, Process Director will return a maximum of 200 rows in a recordset. This default value can be changed by setting the `nMaxBusinessValueRows` Custom Variable in the customization file.

## Related Topics

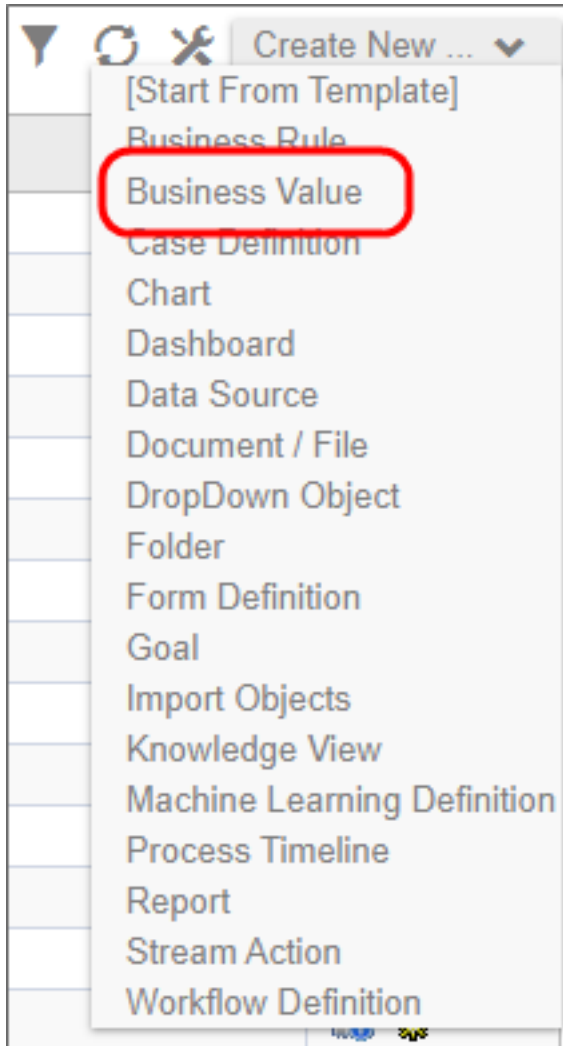
[Creating Business Values](#): Instructions on Business Value configuration.

[Business Value Operations](#): Additional capabilities of Business Values.

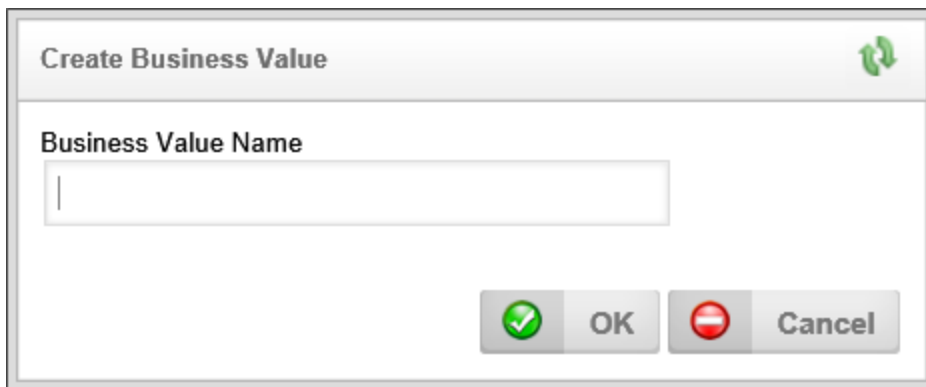
## Creating Business Values

As a best practice Business Value should be stored in a centralized location in your Process Director installation. Business values are global objects that can be called by any application, so BP Logix recommends that they be created and stored in a [\[Business Values\]](#) folder at the root level of the Partition.

To create a new Business Value, navigate to the folder into which you want to create the value, then select **Business Value** from the **Create New** dropdown in the upper right corner of the Process Director screen.

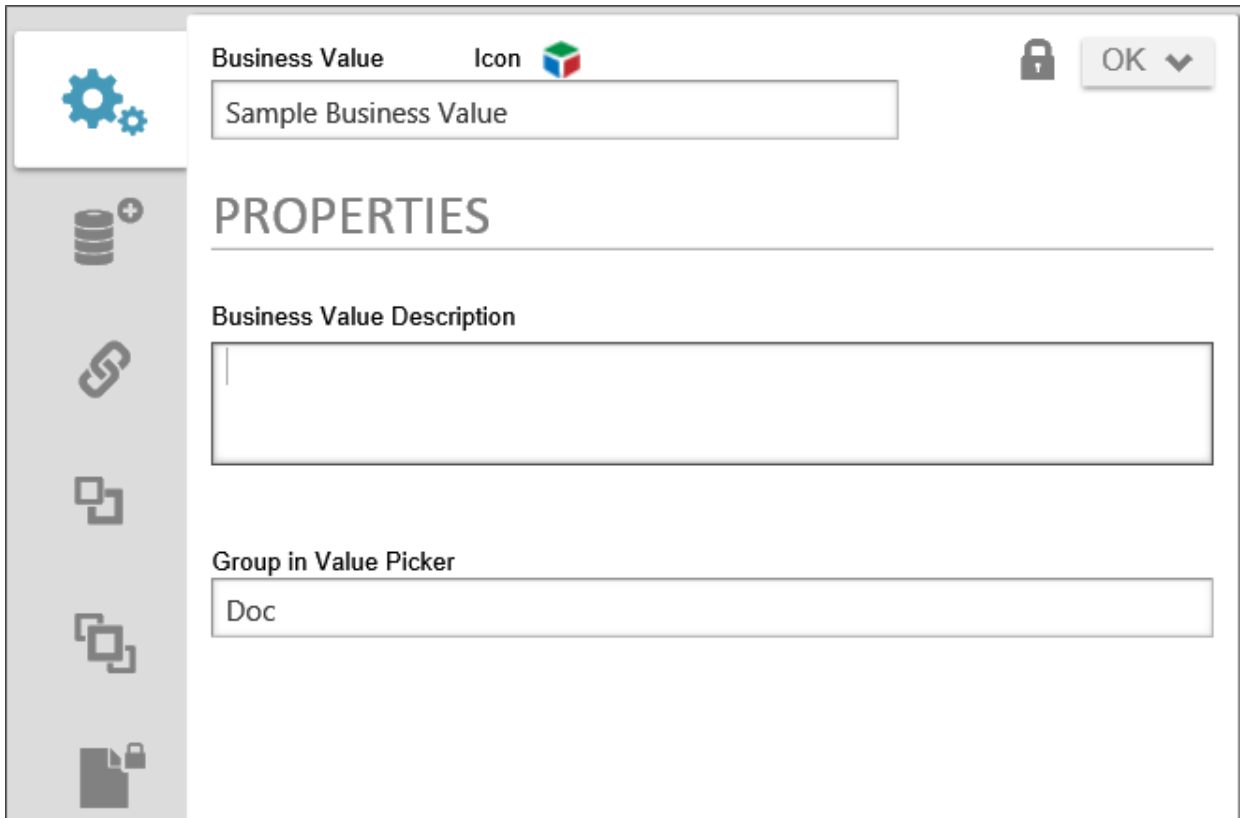


The **Create Business Value** screen will appear. Enter a name in the **Business Value Name** field, then click the **OK** button to save the new Business Value and open the configuration screen.



There are two main configuration tabs in the **Business Value** definition that are used to configure it, The **Properties** and **Configure** tabs.

## Properties Tab #



The screenshot shows a configuration window titled "Business Value" with a sub-label "Icon" and a small 3D cube icon. The window has a sidebar on the left with icons for settings, database, link, copy, and document. The main area contains a text field with "Sample Business Value", a "PROPERTIES" section header, a "Business Value Description" text area, and a "Group in Value Picker" text field containing "Doc". There is a lock icon and an "OK" button with a dropdown arrow in the top right corner.

### Business Value Description

The detailed description of the Business Value's purpose, data sources, etc.

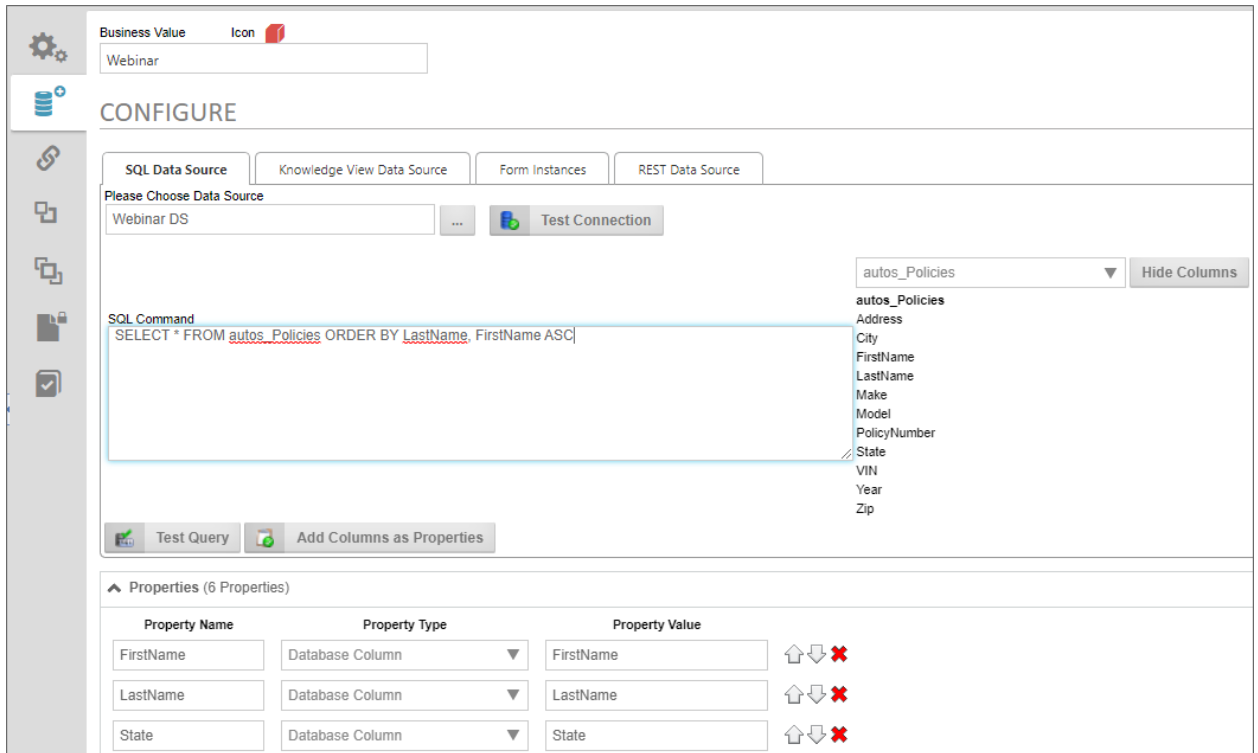
### Group In Value picker

The name of the group under which the Business Value's name will be displayed in dropdown menus. This property enables you to organize Business Values into logical groups that will display as submenus in the properties dropdown menus that display in the **Condition Builder** and elsewhere. This feature operates in the same way as it does for Business Rules.

## Configure Tab #

The **Configure** Tab shows the general properties that apply to the Business Value.

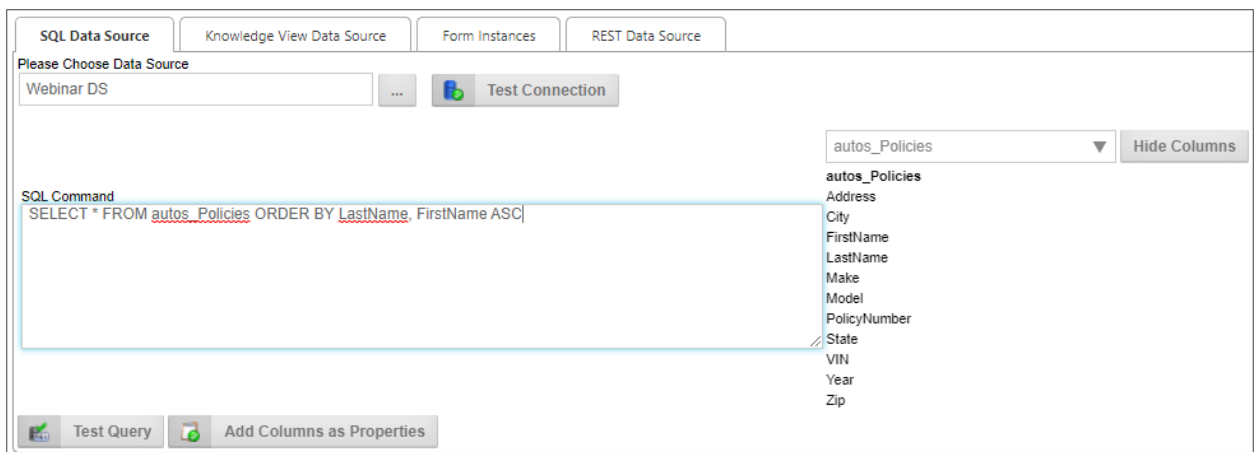




You can select a data source from a SQL database, a Knowledge View, a set of Form Instances, or a REST web service to provide the Business Value. In most cases, you'll choose a single data source, but you can have multiple values extracted from both a SQL and a Knowledge View data source.

### SQL Data Source Tab

The **SQL Data Source** tab enables you to select a SQL Data Source, and provide a SQL Command for returning data from a SQL database. The Business Value supports most SQL data types (Char, Varchar, Nvarchar, Integer, Double, Short, Byte), and converts the value to a string for the remaining data types.



### Please Choose Data Source

A **Content Picker** control that enables you to pick the Datasource from which the data will be extracted.

### Test Connection Button

Clicking this button will attempt to connect to the selected Datasource to confirm that the connection to the Datasource works.

## Database Table

Once a valid Datasource has been selected, a list of database tables associated with the Datasource will fill the dropdown. If you select a table from the dropdown, a list of table columns will appear on the right side of the **SQL Command** text box, in the **Database Column** field.

## SQL Command

Enter the SQL command that will select the data to be set as the Business Value.



While the system will accept any valid SQL command, BP Logix *very strongly recommends* that you limit the use of Business Values to SELECT commands. Updating, deleting, and inserting data should be done separately, through the appropriate Custom Task/Datasource.

## Database Column

Displays a list of database columns in the table selected from the **Database Table** dropdown.

## Show/Hide Columns Button

Clicking this button will show or hide the columns in the **Database Column** property.

## Test Query Button

Clicking this button will run the SQL query in the **SQL Command** property against the database to confirm that the query works properly.

For Process Director v5.44.703 and higher, the **Test Query** button will display up to 100 rows of sample data from a SQL Datasource in the Business Value definition. The data will be displayed in a **SQL Test Results** section on the page. This section is hidden until the **Test Query** button is clicked.

PolicyNumber	FirstName	LastName	Address	City
A458709718	Eduardo	Bowman	801 Linden Street	Basking Ridge
A458786894	Luther	Bush	4884 Olen Thomas Drive	Wichita Falls
A458825482	Roger	Carson	1457 Locust Court	Long Beach
A458748306	Minnie	Daniels	1323 Sand Fork Road	Plymouth
A458651836	Terrell	Ford	986 Clinton Street	Ossining
A458671130	Carl	Fuller	234 Fawn Lane	Greenwood
A458815835	Myrtle	Garrett	1090 Coleman Avenue	San Diego

## Add Columns as Properties Button

For Process Director v5.44.703 and higher, this button will automatically add every column returned by the SQL statement as a Business Value property in the **Properties** section of the page. This feature eliminates the need to manually create each property.

## Knowledge View Data Source Tab

The **Knowledge View Data Source** tab enables you to select a Knowledge View and apply a filter for returning data from a Knowledge View.

**i** All Knowledge View columns are available to the Business Value, including those that are marked not to display in the Knowledge View Definition. Also, Knowledge Views that contain sorting criteria will be presented with the same sort order in the Business Value. For v5.32 and higher, a Knowledge View that uses folder navigation to filter the results can now be used as the data source for a Business Value to return all data, and ignore the folder navigation filter.

## Pick Kview

A **Content Picker** control that enables you to pick the Knowledge View from which the data will be extracted.

## Knowledge View Filter Data

If the Business Value (and underlying Knowledge View) uses a filter Parameter, the Parameter has to be passed to the Knowledge View as a filter string, using the same format you'd use to pass filter strings from a Knowledge View control on a Form, e.g.:

```
KViewFilter1={PARAMETER:Parameter1Name}.
```

If you are passing more than one filter parameter, then each parameter passed should be on its own line in the **Knowledge View Filter Data** text box, e.g.:

```
KViewFilter1=                                     {PARAMETER:Parameter1Name}
KViewFilter2={PARAMETER:Parameter2Name}
```

## Execute Knowledge View under this user context

A **User Picker** control that enables you to pick the user under whose permissions the Knowledge View will be run. Since regular users may not have permissions to view the Knowledge View, you must pick a user who has appropriate permissions to run the Knowledge View. This will enable users with a lower permissions level to access the Knowledge View data for the Business Value.

## Test Knowledge View Button

Clicking this button will run the Knowledge View to confirm that it works properly.

## Form Instances Tab

For users of Process Director v5.13 and higher, Business Values may also use Form Instances as data sources, to retrieve a list of form instances, or form field values. This tab won't appear in earlier versions of the product. For users of Process 5.26 and higher, the Form Instances data source won't return canceled or incomplete form instances, but it will return form instances that were saved for later, or attached to a process.



The screenshot shows a configuration window with four tabs: "SQL Data Source", "Knowledge View Data Source", "Form Instances" (which is selected), and "REST Data Source". Below the tabs, there is a "Form Definition" section with a text input field and a button with three dots. Below that, it says "Include Form Instance when: [Click to create condition ...](#)". At the bottom left, there is a button with a green checkmark icon and the text "Test Form Conditions".

## Form Definition

An **Object Picker** control that enables you to select the form definition whose instances will be retrieved.

## Include Form Instance when

This property will use the **Condition Builder** to enable you to filter the Business Value to return only form instances that match the specified condition.

## Test Form Conditions

This button will, when form conditions have been configured, display the number of form instances that match the configured condition.

## REST Data Source Tab

The **Rest Data Source** tab enables you to type a URL for a REST data source into the **REST URL** text box.

### Parse REST Response As

This property enables you to choose how Process Director will parse the REST Response. By default, Process Director will parse both JSON and XML responses as XML, enabling you to use XPath to parse the response. Some JSON, however, can't be converted to XML. In that case, parsing using JSONPath will be required. For Process Director v5.25 and higher, Business Values that use JSONPath to target data from a REST call can use expressions that return a JSON node rather than a string or scalar value.

### Please Choose Compliance (Credentials) Data Source

This property enables you to select a Compliance Datasource that contains the credentials needed to authenticate with the REST service that supplies the data.

 This property name was changed to "Compliance" from "Credentials" in v5.44.1000.

### Add HTTP Header

Users of Process Director v4.42 and higher also have an **Add HTTP Header** button that, when clicked, will add one or more rows of **Name/Value** pairs that will be sent via HTTP header to the REST service.

### Test Rest Call

The **Test REST Call** button will run the Web Service REST call to confirm that it works properly.

XPath queries can be used against REST data in Business Values, so that Business Value properties can be assigned values returned in the XML response to the query. For instance, let's say you have a REST call to return Zip Codes from a particular city and state, and REST response returned the following XML data:

```
<?xml version="1.0" encoding="UTF-8"?>
- <results>
  - <result>
    <zip>92140</zip>
    <city>Mcrd San Diego</city>
    <state>CA</state>
  </result>
  - <result>
    <zip>92101</zip>
    <city>San Diego</city>
    <state>CA</state>
  </result>
  - <result>
    <zip>92102</zip>
    <city>San Diego</city>
    <state>CA</state>
  </result>
</results>
```

Using the XPath syntax `results/result/zip`, you could return a list of the zip codes as a data column in the Business Value, as shown below:

The screenshot shows a configuration window titled "Properties (1 Property)". It contains a table with three columns: "Property Name", "Property Type", and "Property Value". The first row has "value" in the first column, "REST Data" in the second column (with a dropdown arrow), and "results/result/zip" in the third column. To the right of the "Property Value" field are three icons: an up arrow, a down arrow, and a red X. Below the table is a button labeled "Add Property" with a green plus icon.

**i** Business Values also support JSON returns and JSONPath, as well.

For more information about XML or JSON REST Services, and how to use XPath and JSONPath to parse REST Responses, please refer to the [REST Services topic](#) of the Developer's Guide.

## Properties Section

Once you've selected the appropriate data source for the Business Value from the configuration tabs, you can fill out this section by adding the properties that contain the desired Business Value or Values. Simply click the **Add Property** button to add a value property. The value property contains a number of fields that must be configured.

The screenshot shows a configuration window titled "Properties (1 Property)". It contains a table with three columns: "Property Name", "Property Type", and "Property Value". The first row has "VINNumber" in the first column, "Database Column" in the second column (with a dropdown arrow), and "VIN" in the third column. To the right of the "Property Value" field are three icons: a red X, an up arrow, and a down arrow. Below the table is a button labeled "Add Property" with a green plus icon.

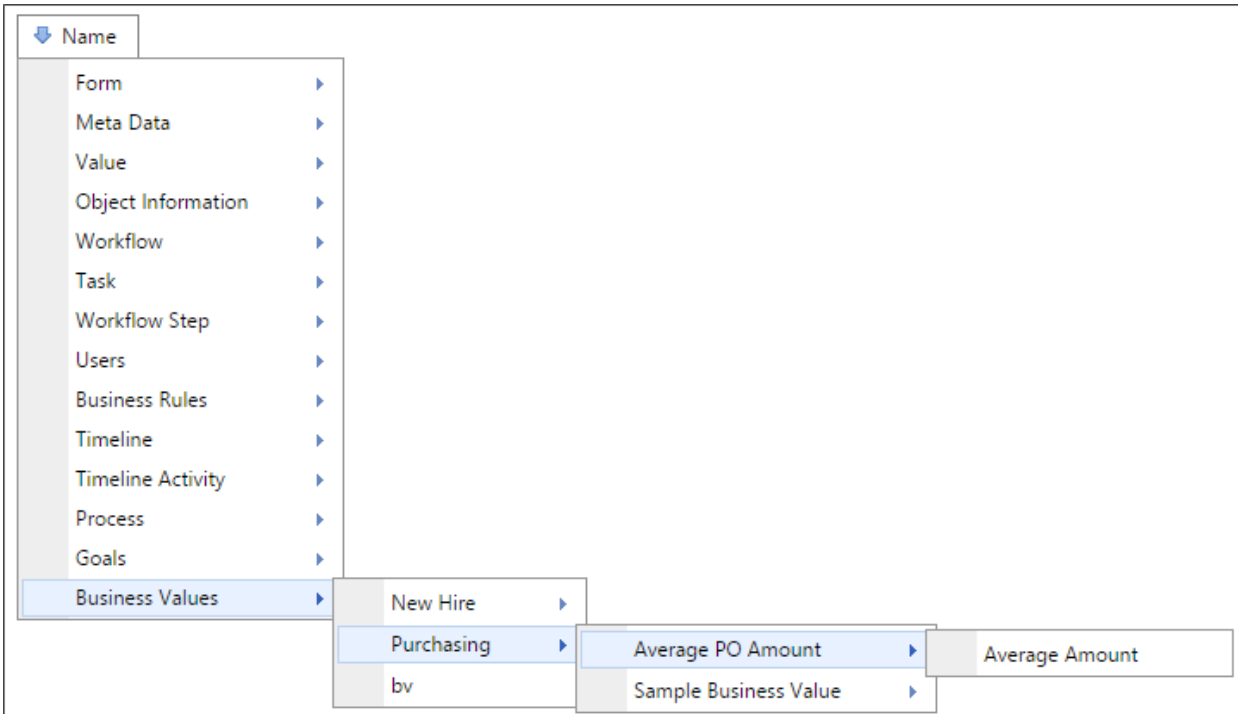
Property	Options	Description
Property Name		The property name will appear as the Business Value property when selecting Business Values from the <a href="#">Condition Builder</a> dropdown menus.
Property Type	Database Column Number of Database Rows Form Field Form Instance Knowledge View Column Number of Knowledge View Rows REST Data Entire REST Response System Variable Dropdown	The type of object from which the value is extracted.
Property Value (Fields)	List of available fields	<p>This dropdown control appears when the Database Column, Form Field, or Knowledge View Column Property Type is selected.</p> <p>For the Database Column, the field will appear as a text box into which you can type the desired database column name.</p> <p>For the Form Field, a <a href="#">Content Picker</a> will be displayed that you can use to select the form field to use for the property.</p> <p>For the Knowledge View Column data type, it will be displayed as a dropdown containing a list of Knowledge View columns from which you can select the desired column containing the value.</p>
Property Value (Operators)	Column Data Sum Average Min Max Average (Ignore 0 values) From First KView Result From Last KView Result	<p>The type of calculation desired to produce the value. Selecting Column Data provides the raw value of each row, which will result in a multi-row Business Value. The remaining calculation operators will produce a single value, such as a Sum or Average of all values in the returned rows.</p> <p>Additionally, If the Data Source is a Knowledge View you can return the first or the last record in the Knowledge View as a scalar (single) value. This is primarily useful when the Knowledge View results are sorted in the Knowledge View definition.</p>

**i** The Form Field and Form Instance data types are only available for users of Process Director v5.12.02 and higher.

The properties returned can be ordered by using the UP and Down arrow icons, and deleted by clicking the red "X" icon.



Each property can be accessed in the [Condition Builder](#) via the value dropdown menu.



## Parameters Section

The [Parameters](#) section enables you to add Parameters to your queries, and in the case of Business Values based on SQL Datasources, supply a test value to apply when the query is tested.



**i** For Process Director v5.26 and higher, Parameters can also be used as column values in both the SQL statement, and the Parameter as a Property Value. For instance, A SQL statement such as "SELECT {\${PARAMETER:column}} from TABLENAME" can be used to return a column for a Property that uses "{PARAMETER:column}" as the Property Value. Thus, the Parameter can supply the



column name for both the SQL Statement and be used as the value of the Property itself. Please note that the Property Name must still be a static value. Similarly, System Variables may be used in the Property values for REST data sources. This enables users to build parameterized filter expressions that can be used to filter the JSONPath/XPath results, e.g.:  
`$.components [? (@.classification >= /Graded/I && @.section = {$PARAM:section})].section`

## Name

The **Name** column enables you to specify a parameter name.

## Test Value

The **Test Value** column will only display when the **SQL Data Source** tab is selected, as only the SQL Datasource supports passing test values to the SQL command. In this column, type the test value you'd like to use when testing the query. Once you've entered a test parameter, you can click the **Test Query** button on the **SQL Data Source** tab to test the query with the test parameters you entered.

Note that, in the example above, the SQL statement uses the parameter in the WHERE clause, with the syntax `{$Parameter:Year}`. The "\$" in the system variable instructs Process Director to format the parameter value as a SQL-encoded string.

## Add Parameter

Click this button to add a new parameter.

Once you've added parameters to a query, every place where you choose the Business Value from the **Choose System Variable** dialog box, the parameter will display in its own section of the dialog, below the Business Value you select. You can use the parameter portion to select from where the parameter's value will be derived.

## Parameter Example

For instance, let's say there's a Form that has a **VendorName** dropdown control for picking a vendor, which uses the Vendor's name as the display text, and the VendorID as the value. Using the Business Value, you can automatically fill one or more fields with the Vendor's company information by using the **VendorName** field as the Vendor parameter, and passing the parameter to the Business Value.

In this example, the **Set Form Data** tab on the Form is configured to set the value of several form fields when the **VendorName** changes.

## SET FORM DATA

Select Event ...  Add Action

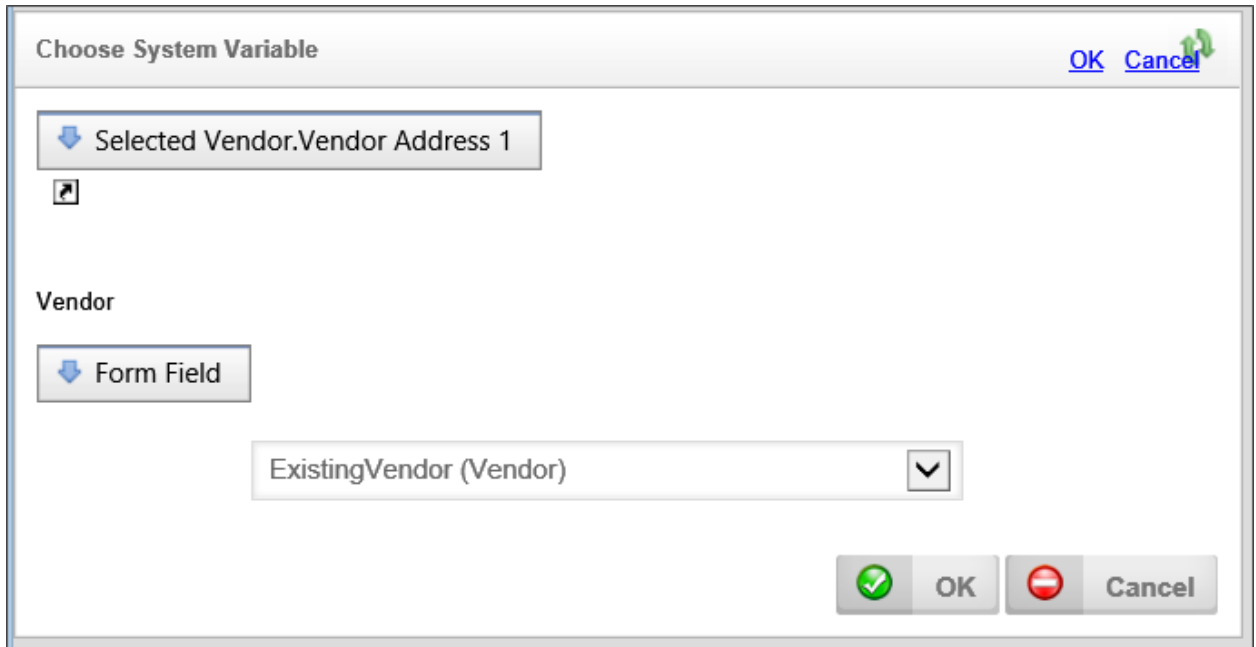
ExistingVendor (Vendor)  Add Condition

[Click to create condition ...](#)  Move to Event...  Add Form Field

Business Value: Selected Vendor  Add Form Field

Vendor	Vendor	...
VendorName (Vendor Name) <input type="button" value="▼"/>	VendorName <input type="button" value="▼"/>	<input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="✖"/>
VendorAddress1 (Vendor Address first line) <input type="button" value="▼"/>	Vendor Address 1 <input type="button" value="▼"/>	<input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="✖"/>
VendorAddress2 (Vendor Address Second Line) <input type="button" value="▼"/>	VendorAddress2 <input type="button" value="▼"/>	<input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="✖"/>
VendorCity (Vendor City) <input type="button" value="▼"/>	VendorCity <input type="button" value="▼"/>	<input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="✖"/>
VendorContact (Vendor Contact) <input type="button" value="▼"/>	VendorPOContact <input type="button" value="▼"/>	<input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="✖"/>
VendorState (Vendor State) <input type="button" value="▼"/>	VendorState <input type="button" value="▼"/>	<input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="✖"/>
VendorZip (Vendor Zip) <input type="button" value="▼"/>	VendorZip <input type="button" value="▼"/>	<input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="✖"/>

If we click on the **Build** button of the **Type** property to open the **Choose System Variable** dialog, we can see the parameter has been set to the value of the **ExistingVendor** field.



The Type property is identified in the Dropdown menu, and, below the dropdown, we set the "CustomerID" parameter to the value of the [CompanyName](#) form field. In this case, the Business Value will return the Customer Type for that specific customer.

In this simple example, we are only returning the Customer Type from the database. We could, however, return any other type of needed information, such as the address, point of contact, phone number, etc., as part of the Business Value as well.

## Documentation Examples <#>

### Related Topics

[Business Values](#): Introduction to the Business Value object.

[Business Value Operations](#): Additional capabilities of Business Values.

## Business Value Operations

Business Values enables you to perform sophisticated data retrieval via the use of parameters, and mathematical aggregation.

### Parameters <#>

An even more powerful way of using Business Values is to use parameters to return values dynamically in your processes. Previously, instead of a Business Value, you'd need to create a Custom Task activity in your Process Timeline, or use a Custom Task in a Form, to return data values to fill fields. As a result, you might have several Custom Tasks, spread out among several applications, each of which would use the same configuration. Any change to the database schema, would require making changes to every single one of those Custom Tasks, making updates tedious and costly in terms of time and effort. While there are certainly times that a database Custom Task will still be needed, the Business Value can replace a Custom Task in nearly all use cases.

Let's take a look at a hypothetical case. We'll assume that we have a database table that stores customer information. We can create a Business Value to provide that information. Let's look at how such a Business Value might be configured.

The screenshot shows the configuration interface for a Business Value. It is divided into several sections:

- SQL Data Source:** Includes tabs for "SQL Data Source", "Knowledge View Data Source", "Form Instances", and "REST Data Source". Below these is a "Please Choose Data Source" section with a "Track Data" field, a "Test Connection" button, and a "Test Query" button.
- SQL Command:** A text area containing the SQL query: `SELECT * FROM [Company_Record_eForm] WHERE bp_Status='Active' AND bp_CompanyID LIKE '%{$PARAMETER:CustomerID}%' ORDER BY [bp_CompanyName]`. To the right is a dropdown menu labeled "[Select Table to Inspect Columns]" and a "Hide Columns" button.
- Properties (4 Properties):** A table with columns for "Property Name", "Property Type", and "Property Value". Each row has a "Test" button with a red 'X' icon.

Property Name	Property Type	Property Value
Name	Database Column	bp_CompanyName
ID	Database Column	bp_CompanyID
Type	Database Column	bp_CompanyType
Count	Number of Database Rows	
- Parameters (1 Parameter):** A table with columns for "Name" and "Test Value". The "Name" column contains "CustomerID" and the "Test Value" column is empty with a red 'X' icon.

Name	Test Value
CustomerID	


The SQL Command for this Business Value retrieves records for active customers. If no "CustomerID" parameter is supplied, the command will simply pull every customer record. If the "CustomerID" parameter is supplied, Business Value will return only a single record. In other words, the same Business Value can return a single or scalar value based on the parameter.

Next, the Business Value uses the database columns to create three properties: the Name, ID, and Type properties will all contain values extracted from the database, In addition, since the Business Value automatically knows how many records have been returned by the SQL Command, the fourth property, Count, will contain the number of records returned.

**i** When a Business Value Property returns no data from a SQL command, it can still be used in a condition, because the value will be set to an empty string, allowing comparisons in the conditions to work correctly.

Once we have created the Business Value, it's accessible by all other objects in Process Director. None of your business users or other implementers need to know anything about how the data is configured to use

the it, because the Business Value properties are available through the Condition Builder dropdowns. They merely need to select the properties they want, which means that, in most cases, implementers no longer need to create a Custom Task to retrieve the data. They no longer need to know or care about the database schema or the SQL syntax to retrieve the data they want. This functionality greatly reduces the level of technical knowledge that business users and implementers need to conduct relatively sophisticated data retrieval operations. With Business Values, technical details about the data, and how to acquire it, is largely irrelevant to implementers. The only person who needs to have any sort of technical knowledge about the database is the creator of the Business Value itself.

 Combined with the scheduling and process initiation available in the Goals feature, Process Director is a powerful process automation system, enabling automatic process initialization based on data from external systems.


## Mathematical Operations #

Another function available for Business Values is the ability to perform mathematical operations on a Knowledge View column. When retrieving any numeric column data, Process Director can perform the following calculations on the values:

- **Sum:** Returns the sum of all the values in the column.
- **Average:** Returns an average of all the values in the column.
- **Min:** Returns the lowest value in the column.
- **Max:** Returns the highest value in the column.
- **Average (ignore 0 values):** After eliminating the rows that contain a value of 0 or null, returns an average of all the remaining values in the column.
- **First:** Returns the first row from the Knowledge View.
- **Last:** Returns the last row from the Knowledge View.
- **Distinct:** Returns only unique rows from the Knowledge View, in a manner similar to a SQL Server DISTINCT operator.

^ Properties (4 Properties)

Property Name	Property Type	Property Value		
<input type="text" value="Name"/>	Knowledge View Column ▼	Name ▼	Column Data ▼	↑ ↓ ✖
<input type="text" value="Type"/>	Knowledge View Column ▼	Object Type ▼	Column Data ▼	↑ ↓ ✖
<input type="text" value="Path"/>	Knowledge View Column ▼	Folder Path ▼	Column Data ▼	↑ ↓ ✖
<input type="text" value="Author"/>	Knowledge View Column ▼	Author ▼	Column Data ▼	↑ ↓ ✖

 Add Property

^ Parameters (1 Parameter)

Name	Test Value	
<input type="text" value="ObjectName"/>	<input type="text"/>	✖

Column Data

Column Data

Sum

Average

Min

Max

Average (ignore 0 values)

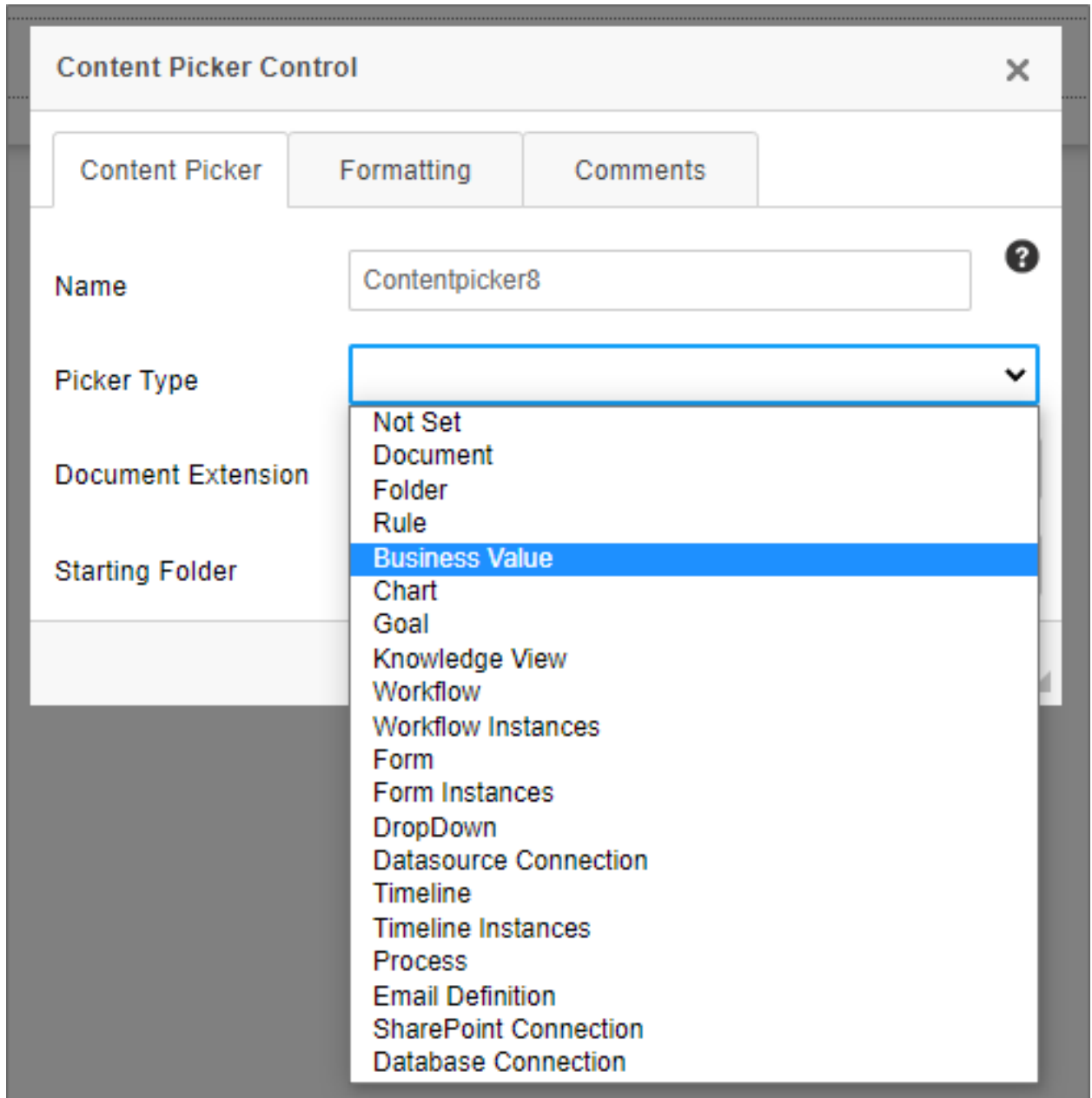
First

Last

Distinct

## Other Usages #

Once a Business Value has been created, it can be selected as a content type from the [Content Picker](#) control when creating Forms.



Business Values can also be accessed through a System Variable. For an example of using a system variable with a parameter, take a look at the following Business Value configuration for a notional Business Value named "CustomerInfo":

SQL Data Source | Knowledge View Data Source | Form Instances | REST Data Source

Please Choose Data Source  
InternalDB ... Test Connection

SQL Command  
SELECT \* FROM [Company\_Record\_Form] WHERE bp\_Status='Active' AND bp\_CompanyID LIKE '%\${PARAMETER:CustomerID}%' ORDER BY [bp\_CompanyName]  
[Select Table to Inspect Columns] Hide Columns

Test Query Add Columns as Properties

Properties (4 Properties)

Property Name	Property Type	Property Value	
Name	Database Column	bp_CompanyName	↑ ↓ ✖
ID	Database Column	bp_CompanyID	↑ ↓ ✖
Type	Database Column	bp_CompanyType	↑ ↓ ✖
Count	Number of Database Rows		↑ ↓ ✖

Add Property

Parameters (1 Parameter)

Name	Test Value	
CustomerID		✖

Add Parameter

So, in this example, let's say we want to return the **Name** property for the customer whose Customer ID is "123" via a system variable. In that case, we could use the following syntax:

```
{BUSINESS_VALUE:CustomerInfo.Name, $CustomerID=123}
```

Please see the documentation on [Business Value System Variables](#) for more information.

## Related Topics

[Business Values](#): Additional capabilities of Business Values.

[Creating Business Values](#): Instructions on Business Value configuration.

## Case Management Overview

Process Director implements Case Management as a core feature of the product.

### What is Case Management?

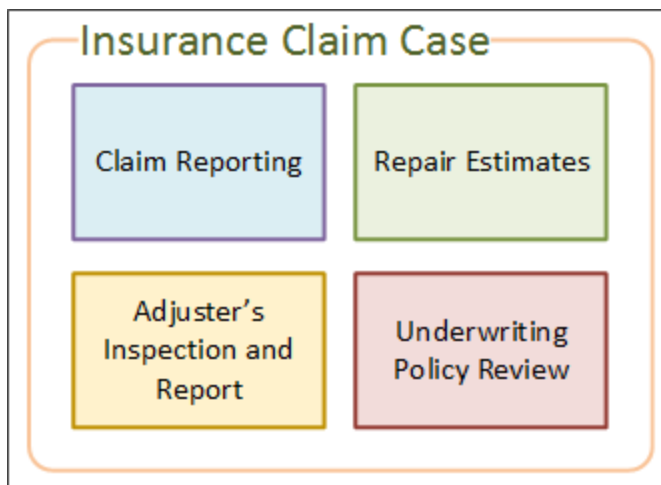
A **Case** is group of processes, transactions, or responses that define a complex Activity, and which can be tracked over a period of time. A Case usually involves actions by many different people, both inside and outside of the organization. Every action, message, response, and document generated during this complex Activity becomes part of the Case. **Case management** is the ability to organize and track Cases, and



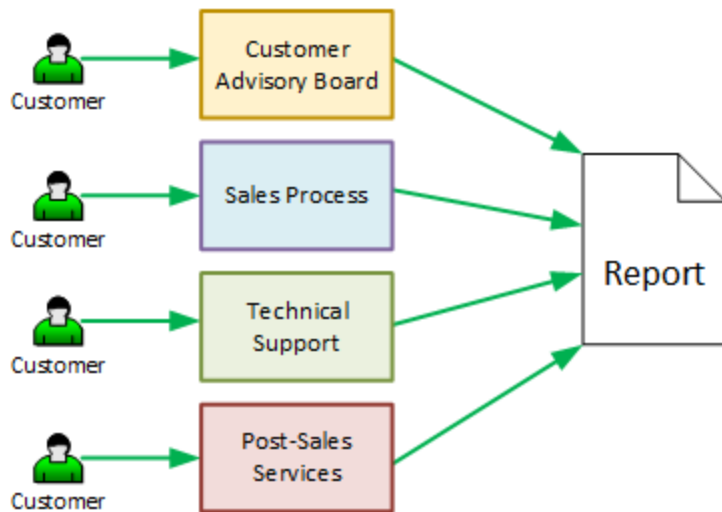
emphasizes the collation of all of the information that relates to the Case, rather than concentrating on a specific process in the way that traditional Business Process Management does. In the context of Process Director, we would define a Case as being a set of related, but separate, processes.

The question immediately arises, "What constitutes an individual Case?" A common misconception is that the Case revolves around a specific person, like a customer. But let's examine this. A customer can do many things, such as participate in a sales process, attend training, or submit a technical support ticket. All of these things can't be a single "customer Case", as they are completely unrelated. Certainly, we might wish to see everything the customer has done, perhaps in a dashboard or report, but the customer can't be the Case, because many of the interactions with the customer are entirely unrelated to each other. An auto insurance claim, on the other hand, is a Case. Once you have a fender-bender, a number of processes have to be completed before the claim is closed. First the claim has to be reported to the insurance company, which reviews the circumstances and determines whether the claim is valid. An adjuster must inspect the vehicle, and submit a report on the damage. Repair estimates must be obtained from one or more repair shops, and the estimate approved. Underwriting must review the policy to determine if changes in risk require higher premiums for the policy. Case management bundles all of these processes into a single claim Case that provides them all with a shared context.

In general, a Case provides a shared context for a set of processes, Forms, and documents that might or might not be otherwise related. Some processes, such as the Repair Estimate, will only run when a claims Case is in progress. Other processes, such as the underwriter review, will happen outside of a claim Case. When you renew your insurance, for example, an underwriter review will be performed. The shared context of the Case enables you to differentiate an underwriter review that occurs when processing a claim from the recurring underwriter reviews that occur when your policy renews. The shared context of a Case enables you to track every process that is started, every form that is submitted, and every document that is uploaded during a Case. This is true whether the processes are related or not. Any object that shares the Case context, becomes part of the Case.



Traditional Business Process Management (BPM) focuses on individual processes, which creates quite a different set of relationships if you want to collate information.



The BPM model is organized around processes, and the primary concern is how the individual process is working, and the details of who participates in the process is secondary. BPM concentrates on a single process, or a parent process with a limited number of child processes, and where the process and/or Form collect all of the data related to the process. Each instance of the process has its own separate life, which isn't dependent upon any other process. Process behavior can vary widely, but only in well-understood and predictable ways that you incorporate into the process' design. The fundamental assumption of the BPM model is that processes will consist of fairly rigid behaviors, with variations that only occur within defined limits.

In the Case Management model, the activities that occur in the Case can't be constrained to a process diagram. Two important characteristics define a Case.

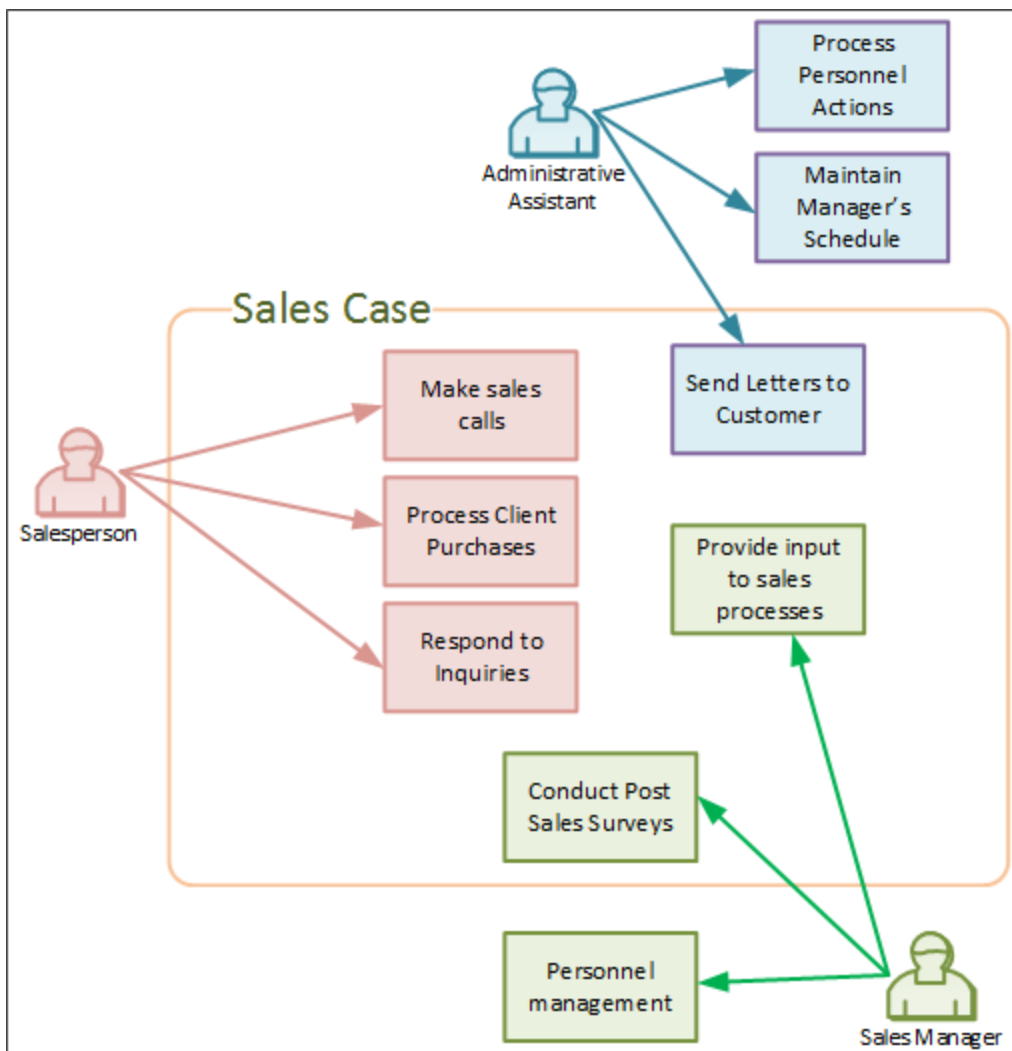
1. The flow of work in a Case, unlike a BPM process, can be nonlinear, recursive and unpredictable.
2. Multiple processes often run in parallel for Case objects.

Two Cases of the same type may require different processes, documents, tasks, or data objects, and activities may need to be performed in a different order. Because of this inherent complexity, the progress of a Case can't be governed by a set of fixed rules like a BPM process. Instead, Case workers typically need to make decisions about how the Case progresses, and what processes and data must be incorporated into the Case. Each individual process in the Case may operate on the more rigid lines of the BPM model, so that activities, milestones, due dates, etc. can be tracked. The Case processes are generally asynchronous with respect to one another, but must share context and data. The overall progress of the Case—what processes must occur, and when—is conducted on a more *ad hoc* basis, and is controlled by the workers or Case manager involved in the Case. In short, Case Management depends on human decision-making, rather than on preconfigured Business Rules.

Case Management styles may also vary widely. Our example of an auto insurance claim is fairly straightforward, because the range of data interactions are well-understood and relatively limited in scope. We know, for example, that an adjuster must inspect the vehicle, and a repair obtained, so our path to completing the Case is known, and every Case will fall within this general structure and reach a known conclusion.

Conversely, a Case that involves a criminal investigation must be highly unstructured and unpredictable. There may or may not be witnesses, physical evidence, multiple suspects, and a variety of crimes that may be charged. Learning new facts may change the outcome of the Case, and we don't know if the Case will end when a suspect is charged, charges are dismissed, or a suspect is never identified. So, Case Management styles can vary from a Case that consists of a collection of known, predetermined procedures, to highly *ad hoc* Cases where the required procedures and outcomes can't be known, and which rely almost entirely on human decision-making.

Because a Case incorporates complex processes that take place over a relatively long period of time, both Case and non-Case workers are likely to participate in the Case. For instance, during a sales Case, a salesperson may work in the context of a Case most of the time. An administrative assistant, on the other hand, may never need to have direct access to a Case, but might be instructed to send a letter to a client, which requires working temporarily in the client's Case for the sole task of sending correspondence.



The Case Management model ensures that only tasks that relate directly to a Case are included in that Case by setting the Case context appropriately for each task that needs to be done. Ideally, the determination of whether a task must be performed in or out of Case context should be determined by the Case

management system, and transparent to the user. When the Administrative Assistant is assigned the task of sending a letter to a customer, work on that task should be automatically placed in the context of that customer's Case, perhaps without the user even being aware of it.

Case Management represents a step forward from traditional BPM processes. One might even argue that many—perhaps most—of what have been seen as traditional BPM processes are actually Cases, and not single processes. Let's take the example of the very common BPM process of employee onboarding.

Employee Onboarding is a complex process that may require tasks be completed by many different parts of the organization. Think of all the things that might need to happen to bring a new employee aboard:

- Administrative processing for insurance and benefits election.
- Mandatory training on policies and procedures.
- Assignment/provisioning of office space.
- Equipment issue.
- Computer/network provisioning.
- Supervisor assignment.
- Order business cards

Two things are important to remember at this point.

1. Every one of the onboarding items above may be full processes in their own right. For example, provisioning of office space may require that Maintenance ensure an office or cubicle is clean, Facilities might need to provide furniture, Communications might need to install a telephone and activate a phone number, and Supply might need to provide a cell phone. Simply providing the new employee a place to work, and the ability to talk to people, requires action from many separate parts of the organization.
2. Not every task shown above will need to be done. A warehouse worker probably doesn't need an office or business cards, while a sales manager will need both. The warehouse worker probably doesn't need a computer personally assigned, but probably will need a network ID to work on, say, inventory management or order fulfillment on one of the warehouse's computers.

These two considerations make the onboarding process very complex. Trying to encapsulate all of these activities into a single BPM process, as has been done traditionally, is a difficult job that requires complex nesting, chaining, and decision logic, and, even then, human decision-making still is required to determine what specific tasks have to be performed for different employees. If we re-imagine employee onboarding as a Case, however, most of that difficult logic is eliminated. The Case would still contain all of the required processes for IT, HR, Facilities, etc., but each Case would be tailored to the specific requirements of the new employee.

The key advantage of the Case Management model is that it marries human collaboration and decision-making to the process engine that executes processes and tracks deadlines and milestones. Moreover, the Case Management model makes all aspects of the Case visible, and doesn't segregate the Case workers to their tiny piece of the process. Thus, Case management combines the advantages of human collaboration and decision-making with the process automation of traditional BPM.

In reality, the line between BPM and Case Management is very blurry. There are very few work processes, outside of rigid compliance and manufacturing processes, that can be completely and accurately

programmed by the BPM process, because nearly every process has some element of *ad hoc* decision-making. Conversely, even high-level strategic processes, such as knowledge work or innovation management, require some level of structure to their individual elements, and aren't completely *ad hoc*. In other words, every process exists along a continuum from pure BPM (rare) to pure Case management (rare). As such, deciding when to implement a process as in BPM, with rather complicated nesting, iteration and chaining logic, and when to implement the process with Case Management, has to be done very much by feel, rather than by any hard and fast rules.

## See also:

[The Case Definition](#)

[Implementing a Case Management Application](#)

## Case Definition

The Case definition specifies all of the properties for a Case. The Case properties create a shared data context that is passed between all of the processes, Forms, and Knowledge Views that are used in the Case. Some changes have been applied to the Case definition for v4.02 and higher, in order to use the Dashboard object, rather than a Workspace, to display the Case folder for a Case instance.

### Configuration Tab

The screenshot shows the 'CONFIGURATION' tab for a Case. On the left is a vertical sidebar with icons for settings, list, link, copy, and document. The main area contains the following fields and controls:

- Case:** A text field containing 'Auto Claim'.
- Icon:** A key icon next to the Case field.
- Case Description:** A text area with the placeholder text 'Enter a brief description of this Object'.
- Case Instance Name:** A text field containing the template 'Claim {CASE:Claim Number} for Policy #{CASE:Policy Number} Submitted on {CASE:Claim Date}'.
- Audit property data changes:** An unchecked checkbox.
- Create Case Instances for existing Form Instances:** A button.

The **Configuration** tab contains the following properties.

**Case:** The name of the Case.

**Icon:** The Icon that will be displayed for each Case instance. Clicking on the default icon will open the **Icon Chooser** dialog box to select the desired icon for the case.

**Case Instance Name:** The naming format for each case instance that is created. The Instance name can include System Variables, as shown in the example below:

Claim {CASE:Claim Number} for Policy #{CASE:Policy Number} Submitted on {CASE:Claim Date}

In this example, the instance name includes three of the Case property system variables to create a unique name for each instance.

**Case Description:** A brief description of the Case's purpose.

**Audit property data changes:** Checking this box will create an audit trail for each change to any of the case properties, enabling all changes to be tracked.

**Create Case Instances for existing Form Instances:** If you click this button, Process Director will find any form definitions that point to this case definition or properties in this case definition. It will create the appropriate case instances and add all the appropriate objects to the case instance. It will search through all form instances, the process associated with the form instance, and all those attachments. You can then select the Instance for which to create a new case.

## Properties Tab

The **Properties** tab contains the list of case properties that create the data context for the case. To add a property, simply click the **Add Property** button. When you do so, a property row will appear with the following columns to configure.

Name	Friendly Name	Description	Data Type	Default Value	Encrypt	Unique	
Claim Number			Text	{CURR_DATE, format=yy}-{SEQ_NUM:{CUF	<input type="checkbox"/>	<input type="checkbox"/>	↑↓✖
Policy Number			Text	None	<input type="checkbox"/>	<input type="checkbox"/>	↑↓✖
Last Name			Text	None	<input type="checkbox"/>	<input type="checkbox"/>	↑↓✖
First Name			Text	None	<input type="checkbox"/>	<input type="checkbox"/>	↑↓✖
Address			Text	None	<input type="checkbox"/>	<input type="checkbox"/>	↑↓✖
City			Text	None	<input type="checkbox"/>	<input type="checkbox"/>	↑↓✖

### Case Properties Section

**Name:** The name of the property that will appear in the dropdown menus for field values and in the Choose System variable dialog box if a **Friendly Name** isn't assigned, and which will be used as the identifier for the property when typing a Case system variable.

**Friendly Name:** A value you can use, if needed, that will display in the user interface instead of the **Name**. This is a useful feature if the case property names are too technical or confusing. For users of Process Director v5.12 or higher, using a Camel Case value (e.g., MyProperty) for the Property's **Name** will automatically create a **Friendly Name** based on the Camel Case **Name** (e.g., My Property).

**Description:** A description of the property's purpose.

**Data Type:** The property's data type. The default data type is Text, and the following data types are available:

- Text
- Number
- Date
- Date/Time

- Yes/No
- User

**Default Value:** The value of the property that will be applied automatically when a new instance is created.

**Encrypt:** A check box that, when checked, will encrypt the value for storage in the database. This is useful for storing sensitive information.

**Unique:** The value won't be repeated, and will be unique to the case instance, i.e., no two case instances will have this same value for the property.

**Order and Delete Icons:** The order in which the properties appear in the Case definition is the same order in which they'll appear in all System Variable dropdown menus. You can change the order by clicking on the up or down arrows for the property's row. Properties can be deleted by clicking the red "X" icon.

### Dashboards Section



The **Dashboards** Section enables you to choose a Dashboard object to use to display the case folder. The dashboard should contain Knowledge Views, Charts, reports, or other objects related to the Case definition. When the dashboard is opened in the context of a Case, only results that are applicable to the selected case instance will be displayed in the Dashboard. For information about how to configure a Dashboard, please see the [Dashboards](#) topic.



### Set Case Properties Tab

To set a case property click the **Add Row** button to add a property row to the page. Each row consists of three columns to configure.

The **Case Property** column contains a dropdown from which you can choose the Case property whose value you wish to set.

The **Set Case Property** column contains an object picker that opens the **Choose System Variable** dialog box to enable you to choose the value you wish to use to set the property value. You can choose any available System Variable, or can manually configure a string or text value.

The **Condition** column contains a link to open the **Condition Builder**. You can configure any desired condition or condition set that, when evaluated as true, will set the Case property to the configured value. The condition is continuously evaluated, much like **Start When** conditions in a Process Timeline.

You can change the order of the rows by clicking on the up or down arrow buttons () , and you can delete rows by clicking on the red X icon button().

## Creating a SQL View #

Sometimes, you might wish to access the stored data about a case definition more easily. So, just as with a Form, Process Director enables you to create SQL views based on case definitions. Each Case Definition has a [Case Property Data SQL View](#) tab.

Case Icon OK ▾

Auto Claim

### CASE PROPERTY DATA SQL VIEW

A SQL View can simplify external access to the database tables. It will present a form or case in the database as a single table with columns that represent the form fields or case properties. You must be a System Administrator to be able to create the VIEW from this page. The VIEW is designed to give easy external access, not for performance.

^ Create new or modify an existing SQL view

Create new SQL view

^ Fields to use in the SQL View

<input checked="" type="checkbox"/> Claim Number	<input checked="" type="checkbox"/> Policy Number	<input checked="" type="checkbox"/> Last Name	<input checked="" type="checkbox"/> First Name
<input checked="" type="checkbox"/> Address	<input checked="" type="checkbox"/> City	<input checked="" type="checkbox"/> State	<input checked="" type="checkbox"/> Zip
<input checked="" type="checkbox"/> VIN	<input checked="" type="checkbox"/> Year	<input checked="" type="checkbox"/> Make	<input checked="" type="checkbox"/> Model
<input checked="" type="checkbox"/> Adjuster	<input checked="" type="checkbox"/> Adjuster Inspection Date	<input checked="" type="checkbox"/> Repair Amount	<input checked="" type="checkbox"/> Repair Complete Date
<input checked="" type="checkbox"/> Open Claim	<input checked="" type="checkbox"/> Claim Date		

Select All Select None

Re-generate View

SQL Create View Command

View Name  
AutoClaim

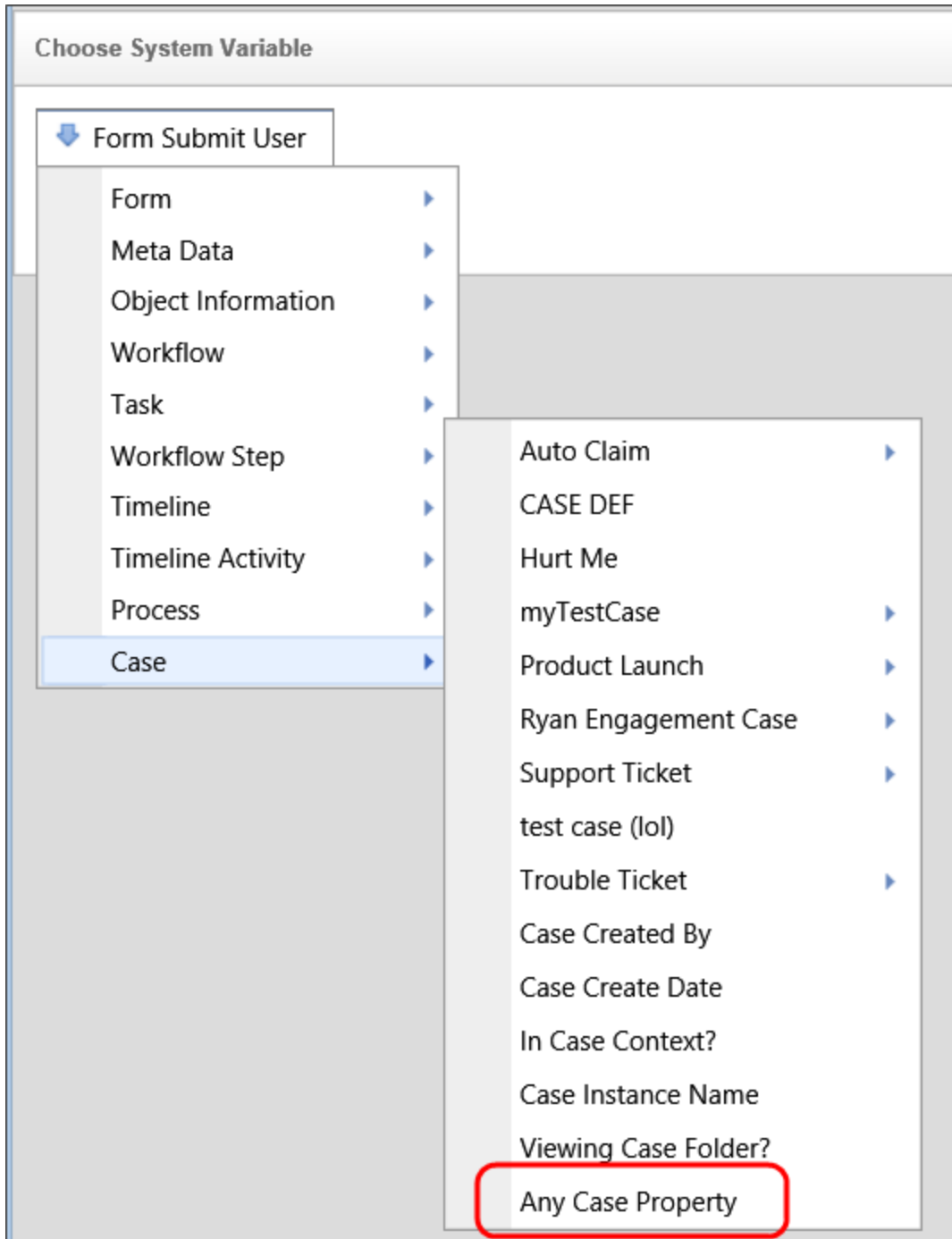
View Description

This tab operates exactly like the [Form Data SQL View](#) tab on a Form Definition. Please refer to the [Create View \(SQL\) topic](#) in the Database Guide for detailed information on how to configure this tab.

## Case Definition Super Search #

In the [Condition Builder](#), the System Variable dropdown includes an [Any Case Property](#) option that will enable a search across all case properties in a Case definition for a specified value.





## Process Director Versions Prior to v4.02

In the initial versions of Process Director that support Case Management (v3.82 to v4.01) there is a Workspaces tab, from which you can display the workspace you wish to use to display the Case Folder. In these versions, you must configure a workspace with the [This workspace is used for case instances](#) property checked to display the case folder information.

## See Also:

[Case Management Overview](#)

[Implementing a Case Management Application](#)

## Implementing a Case Management Application

Case Management is complex, so there's no single process for creating a Case Management Application that is perfectly applicable to all Cases. There are, however, some general guidelines to follow when creating the application.

At minimum, a functional Case Management Application will require that you configure the following objects:

1. **Case Definition:** The Case definition provides the shared data context for all of the objects in each case.
2. **Form(s):** One or more Forms will be needed, with Form fields that set and/or display the Case properties.
3. **Knowledge Views:** One or more Knowledge Views will be needed to show a list of Forms, documents, or process that are related to the case instance. Often, these Knowledge Views are used to display the lists as portlets in the case Dashboard. Additionally, a Knowledge View that returns Case instances will be needed to open the case Dashboard when the instance is selected from the Knowledge View results.
4. **Dashboard:** A case Dashboard will display all of the objects associated with the case, such as Forms or Documents, usually via Knowledge Views. The Dashboard will also provide a method to submit new Forms or otherwise begin new processes related to the Case, usually via navigation buttons added to the dashboard. A Case Dashboard displays only the objects related to a single case.



The initial versions of Process Director that supported Case Management (v3.82 - v4.01) used Workspaces to display Case folders/objects. This UI convention was superseded by the Dashboard object in v4.02 and higher, as the dashboard provides a more versatile method of presenting case objects.

For illustrative purposes, this topic will display elements from a simple, notional Case Management Application, with examples of how the various objects are configured to work with the Case. In our sample case, we will track an automobile insurance claim, and will link three different processes as part of that claim:

1. **Claims Reporting:** A Claim form is filled out, a claims administrator will review it, and an adjuster will be assigned. Ultimately, the claim will be processed and paid, or denied.
2. **Insurance Adjuster's Inspection and Report:** An adjuster will inspect the damage, and make a report which will be reviewed by the claims administrator.
3. **Repair Estimate Approval:** A repair estimate will be admitted, and the claims administrator will approve or deny it.

As we progress through this topic, we'll look at how these different processes are linked to the Case.

## Case Definition #

The starting point for any Case management application is the Case definition. As mentioned previously, the Case definition supplies the data context that all objects in the case will share. With that in mind, the first question that must be answered is, "What data should be stored as Case properties?"

Most obviously, the case should have a Sequence Number, or some other unique property that identifies a specific case instance. Beyond that, however, configuring the Case definition's properties requires some forethought about the property data that needs to be stored. The properties should include any data that is specific to the case instance, such as the date the case was started, or finished. Also, the Case will store data as it exists as the specific point in time that the Case is created or the data is entered. If there is data that will change over time, such as an address, or the name of a point of contact, then that data is a good candidate for inclusion as a case property as well. The sample auto claims case for this topic illustrates the type of data that should be stored in the case instance.

Case Properties							
Name	Friendly Name	Description	Data Type	Default Value	Encrypt	Unique	
Claim Number			Text	{CURR_DATE, format=yy}-{SEQ_NUM.{CURR_D ...	<input type="checkbox"/>	<input type="checkbox"/>	↑ ↓ ✖
Policy Number			Text	None	<input type="checkbox"/>	<input type="checkbox"/>	↑ ↓ ✖
Last Name			Text	None	<input type="checkbox"/>	<input type="checkbox"/>	↑ ↓ ✖
First Name			Text	None	<input type="checkbox"/>	<input type="checkbox"/>	↑ ↓ ✖
Address			Text	None	<input type="checkbox"/>	<input type="checkbox"/>	↑ ↓ ✖
City			Text	None	<input type="checkbox"/>	<input type="checkbox"/>	↑ ↓ ✖
State			Text	None	<input type="checkbox"/>	<input type="checkbox"/>	↑ ↓ ✖
Zip			Text	None	<input type="checkbox"/>	<input type="checkbox"/>	↑ ↓ ✖

In this example, the **Auto Claim** case contains a **Claim Number** property, which uses a formatted sequence number as a default value, and which creates a unique identifier for the **Auto Claim** instance. A number of the properties, such as the VIN number, make, model, and year, of the vehicle, as well as the Customer's contact information, contain information that will change over time. For instance, over the course of several years, the customer may add or remove vehicles for the policy, or may move to a different address. Finally, there are fields such as the Adjuster's Name or the repair estimate amount that are only relevant to this specific Case instance.

In the real world, you probably won't configure your Case definition properly on the first pass. As you create the case, you'll probably run into other data items that you'd like to store as case properties. So, building the Case definition will be an iterative process, where you'll continuously refine your Case properties as you build the Forms and processes that will be part of the case.

## Dashboard #

The **Dashboard** section of the Case Definition enables you to specify a dashboard to use to display the case folder.

**^ Dashboards** + Add Dashboard

Name to display on Tab	Dashboard
Claim #{CASE:Claim Number	Claim # <span style="float: right;">... ↑ ↓ ✖</span>

Note that, as with most text-based properties, you can use System Variables as part of the **Name to display on Tab** property.

With the Case Dashboards, whenever a user opens a case instance, the Dashboard appears automatically, displaying all of the associated files, Forms, documents, etc., that are relevant to the case.

Admin Profile Claims Personnel Home Page User Profile Claim #16-000034

Close case instance

**Case Reports (1 items)**

<input type="checkbox"/>	Name ▲	Size	Author	Last Updated	Version	Action
<input type="checkbox"/>	Adjuster Inspection Form Submitted On 3/18/2016 1:08 PM	8 fields	Dale Franks	3/18/2016	-	

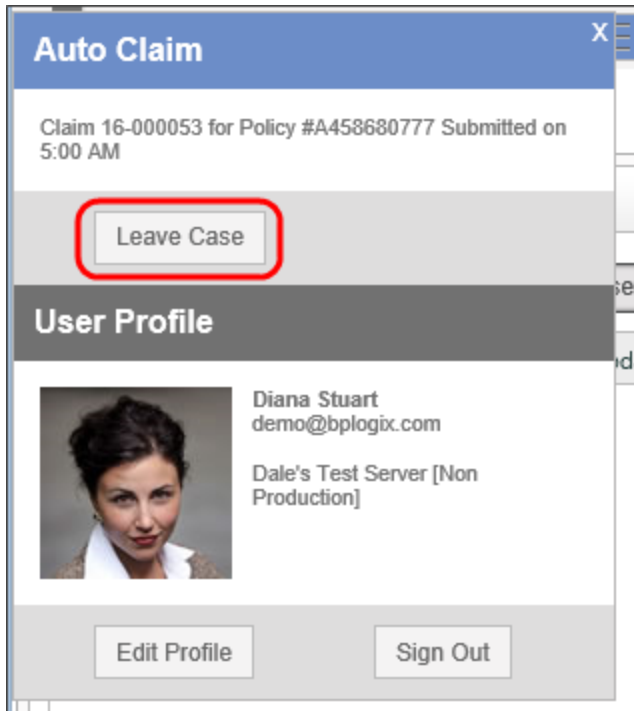
**Case Estimates (1 items)**

<input type="checkbox"/>	Name ▲	Size	Author	Last Updated	Version	Action
<input type="checkbox"/>	Repair Estimate Submitted On 3/18/2016 1:09 PM	22 fields	Dale Franks	3/18/2016	-	

**Current Case (1 items)**

Claim Number ▲	Policy Number	Adjuster	Claim Date	Actions
16-000034	A458806188	Dale Franks	3/18/2016	

When you are in Case context, the **User Profile** box automatically updates to track the case that you're in. When you've completed your work on the case, simply click the **Leave Case** button to exit the Case context.




Additional buttons can be added to the Case dashboard, such as buttons to open Forms, or display other related Knowledge Views.

To learn more about creating and configuring Dashboards, please see the [Dashboards](#) topic.

## Forms #

With the Case Management feature, Form definitions have a property located in the **Options** section of the **Form Options** tab, called, **Automatically create an instance of this Case (optional)**. When you set a Case definition for this property, the Form, when submitted, will create a new Case instance. All subsequent processes, Forms, etc. that are created in the context of the case will be linked to the Case instance.

Form Name    Icon 

Auto Claim

**PROPERTIES**

**Description**

Enter a brief description of this Object

**Instantiated Form Name**

Auto Claim # {CASE:Claim Number}

Form Fields are Enabled by Default

Entire form is read-only when [Click to create condition ...](#)

**Options**

Automatically start this process after Form is submitted (optional)	Auto Claim	...
Workflows Associated with Form (optional)		...
Timelines Associated with Form (optional)	Auto Claim	...
Default Groupname for Form Instances		
Form Script (optional)		...
Automatically create an instance of this Case (optional)	Auto Claim	...
Select a Content List background image for this form (optional)	car.png	...

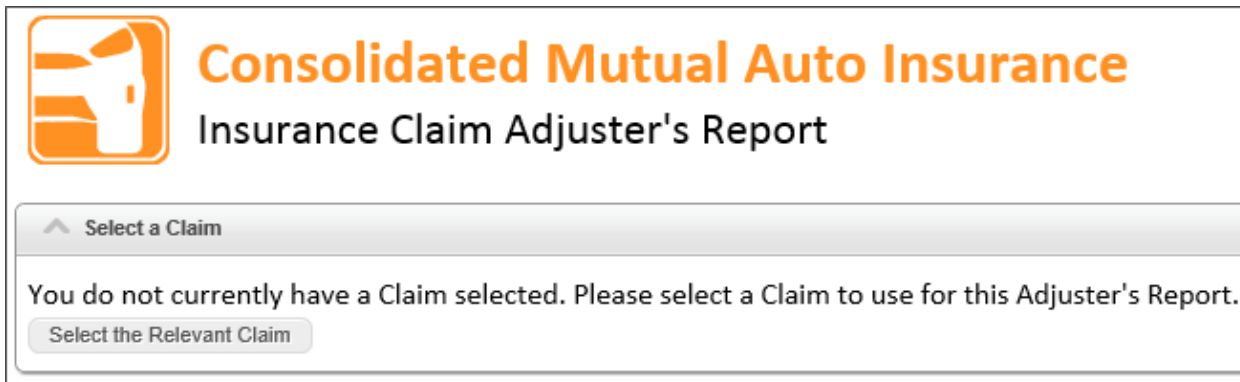
The Case Management functionality also provides some new configuration options for Form fields. The field properties in a Form definition are enabled to use a Case property as a default value and/or to save the field value as a Case property.


The Auto Claim Form for our sample auto claims application uses case properties extensively. In our sample application, policyholder information containing the customer's contact and vehicle information are extracted from an external data source, and filled into the form automatically when the user enters a policy number.

All of the fields for the Policyholder and Vehicle information are disabled and displayed as text on the Form, as we don't want the information edited here. We simply need to display it, and store it to the Case properties. Once the Case properties have been stored, they become the shared data context that can be accessed by any Process Director object that touches the case. When a form instance is saved for any reason, it will update any case instance properties where the form field is configured to point to a case property.

For some Forms, you won't need to store any information to the Case properties, or open a new Case instance, but may still need to associate the Form's data with the case. In the case of the sample project, for example, the [Claims](#) Dashboard has buttons in the navigation area to create a new adjuster's report and a new repair estimate. When you open a new adjuster's report or repair estimate from the Claims Dashboard, the new forms are automatically associated with the case. This is a good solution for creating case-related Forms that will only be opened by case workers, in the context of the Case Dashboard.

For Forms that can be submitted by non-case workers, but which must still be associated with the case, you can have the workers set the Case manually by using an [AttachKView](#) control on the Form that will force users to select the case before filling out the Form. In the sample project, the [Adjuster Inspection Form](#) implements this method of selecting a claim to associate with the Adjuster's Report.



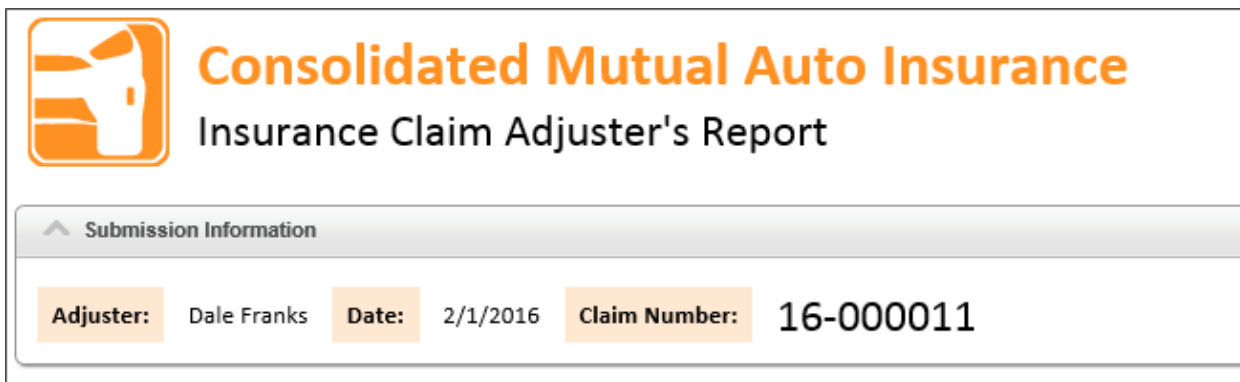
 **Consolidated Mutual Auto Insurance**  
Insurance Claim Adjuster's Report


^ Select a Claim

You do not currently have a Claim selected. Please select a Claim to use for this Adjuster's Report.

Select the Relevant Claim

Clicking on the [Select a Relevant Claim](#) button opens a Knowledge View that displays the currently active insurance claims. When you select the desired claim from the Knowledge View, the Form switches to the appropriate case context.



 **Consolidated Mutual Auto Insurance**  
Insurance Claim Adjuster's Report

^ Submission Information

Adjuster: Dale Franks    Date: 2/1/2016    Claim Number: 16-000011



The choice of how to set the case context for the Form—whether by opening the Form from the Case Dashboard, or by allowing users to manually select the Case—is really an issue of how you want to design how users will interact with the case. Process Director supports both methods.

When assigning a form instance to a case instance using the [AttachKView](#) control, you can switch the form instance to a different case instance by selecting a different case instance from the picker.

In addition, you can set Case properties manually, or from a Form field, using the [Set Case Property](#) Custom Task. Additionally, you can add a form instance to a case via configuration on the Form Definition, through a Case Custom Task, etc. If a form instance is being added to a case, Process Director will check all the form fields on the definition for that form instance and see if any form fields are configured to set case properties, and if so, set the case property with the new form's data.



## Processes #

Because you associate the Form with the Case, in most cases there are no special configuration settings that need to be set on the Process Timelines. The Forms are already "painted" with the case, and some of that "paint" transfers to the processes automatically. The sample project is configured in this way.

There may, however, be times when you do need to associate a process with a case, so there is a Timeline Activity type called the Case Activity. Please see the documentation on Timeline Activity types for more information about this feature.

## Knowledge Views #

On the [Properties](#) tab of a Knowledge View, you can associate a Case with Knowledge View by choosing the desired case from the [Case Definitions Associated with Knowledge View \(optional\)](#) property.

Knowledge View Name Icon  OK 

Current Case

---

## PROPERTIES

---

**Description**

Enter a brief description of this Object

---

**Knowledge View Instance Name**

{KV\_DEF\_NAME} {{KV\_NUM\_ROWS}} items

---

**Forms Associated with Knowledge View (optional)** Auto Claim ...



**Workflows Associated with Knowledge View (optional)**  ...

**Timelines Associated with Knowledge View (optional)**  ...

**Case Definitions Associated with Knowledge View (optional)** Auto Claim ...

**Script Associated with Knowledge View (optional)**  ...

Next, in the **Options** tab of the Knowledge View definition, You can configure the Knowledge View to return Case instances. When you do so, clicking on a Case instance in the Knowledge View's results will open the case instance in the Case Dashboard you configure.

Knowledge View Name Icon  OK 

Current Case

---

## OPTIONS

---

**Allow These Actions on Objects**

<input type="checkbox"/> Copy / Move	<input type="checkbox"/> Delete	<input type="checkbox"/> Properties	<input type="checkbox"/> Meta Data	
<input type="checkbox"/> Permissions	<input type="checkbox"/> Export	<input type="checkbox"/> Run	<input type="checkbox"/> Attach Link	<input type="checkbox"/> Cancel Process

---

**Return These Object Types**

<input type="checkbox"/> Form Definitions	<input type="checkbox"/> Workflow Definitions	<input type="checkbox"/> Timeline Definitions	<input type="checkbox"/> Knowledge View Definitions	<input type="checkbox"/> Case Definition
<input type="checkbox"/> Form Instances	<input type="checkbox"/> Workflow Instances	<input type="checkbox"/> Timeline Instances	<input checked="" type="checkbox"/> Case Instances	<input type="checkbox"/> Documents
<input type="checkbox"/> Business Rules	<input type="checkbox"/> Data Sources	<input type="checkbox"/> DropDown Objects	<input type="checkbox"/> Folders	<input type="checkbox"/> Applications
<input type="checkbox"/> Reports	<input type="checkbox"/> Charts	<input type="checkbox"/> Dashboards	<input type="checkbox"/> Data Lists	
<input type="checkbox"/> Business Values	<input type="checkbox"/> Goals	<input type="checkbox"/> Machine Learning Definitions	<input type="checkbox"/> Stream Actions	

Finally, the Knowledge View's **Filter** tab is enabled to use Case properties as a Knowledge View filter.

Knowledge View Name: Current Case

Icon: [Icon]

**FILTER**

Perform search immediately when Knowledge View is loaded

Claim Number ... Contains String {ClaimNumber}

Indeed, the Case properties automatically appear in the [Choose System Variable](#) dialog box for use in the [Condition Builder](#).

**Choose System Variable** OK Update Cancel [Refresh]

Auto Claim.Claim Number (Claim Number)

- Form
- Meta Data
- Object Information
- Workflow
- Task
- Workflow Step
- Timeline
- Timeline Activity
- Process
- Case**
  - 10a. Case
  - Auto Claim**
- (eSVQRCode)

Auto Claim.Claim Number (Claim Number)

Auto Claim.Policy Number (Policy Number)

Auto Claim.Last Name (Last Name)

Auto Claim.First Name (First Name)

Auto Claim.Address (Address)

OK Cancel

Because Case management requires the use of a Case Dashboard that shows you all of the information relevant to a specific case, you'll need to create a number of Knowledge Views that will appear in the Case Dashboard portlets. In the sample Case Management Project, the [Claims](#) Dashboard uses two Knowledge Views, [Case Reports](#) and [Case Estimates](#), to display all of the Adjuster's Reports and Repair Estimates, respectively, that are associated with the case.

**i** When running a Knowledge View from inside a form that is in a case context, the results of the Knowledge View will automatically be limited to objects in that same case instance. No additional configuration is required.

### Administrative Note

When running administrative commands in the process administration interfaces, the commands will run in the context of the case instance of which the process is a member.

## Conclusion

Much of the work that is required to match the various objects with a particular Case are done behind the scenes, automatically by Process Director. As long as your Case definition, Forms, and Dashboards are properly configured, Case management should be a relatively seamless process.

## See Also:

[Case Management Overview](#)

[Case Definition](#)

## Charts

Users of Process Director v4.0 and above have access to the Chart object. The Chart object displays a graphical representation of values from Knowledge Views and Business Values to enable easier dashboarding—from either the Process Director interface, or in Forms—without needing the more advanced capabilities of the Advanced Reporting component.


## Configuring a Chart #

Obviously, a chart can show many different types of data, and the data can be quite complex. For the purposes of this documentation topic, we'll use a simple Business Value that returns two columns of data, a Symbol column that contains the stock symbols for a number of companies, and a Quantity column, with the number of shares of each stock in the portfolio.

Just as with any other Process Director object, to create a new chart, select "Chart" from the **Create New...** dropdown located in the upper right corner of the Process Director interface. The **New Chart** screen will appear, enabling you to enter the **Chart Name** for your new chart. When you click the **OK** button, the new Chart object will be created, with the name you've entered. The Chart's options screen will then be displayed, and you can enter a fuller explanation of your chart's purpose in the **Description** field.

The remainder of the options screen consists of several tabs, each of which enable you to configure different aspects of the chart.

## Properties Tab #

Chart Name Icon 

Sample Chart

### PROPERTIES

Description

Enter a brief description of this Object

Chart Title

Chart Type

Show Legend  Show Category Axis Labels  Show Value Axis Labels

Select a Content List background image for this chart (optional)

Select a background image from this list for this chart (optional)

Additional Body CSS Styling

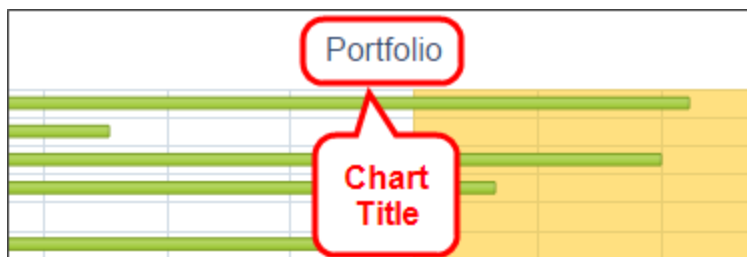
Enter additional CSS to include on your chart.

Chart URL: <http://localhost/chart.aspx?ext=1&id=4f2a02d7-b319-4feb-a9ee-56a1291e3898>

The **Chart Options** section is where you'll set the overall configuration of your chart.

### Chart Title

This is the name that will appear in the Chart's Title area. The Chart's title can use System Variables, which Process Director will parse to display properly.



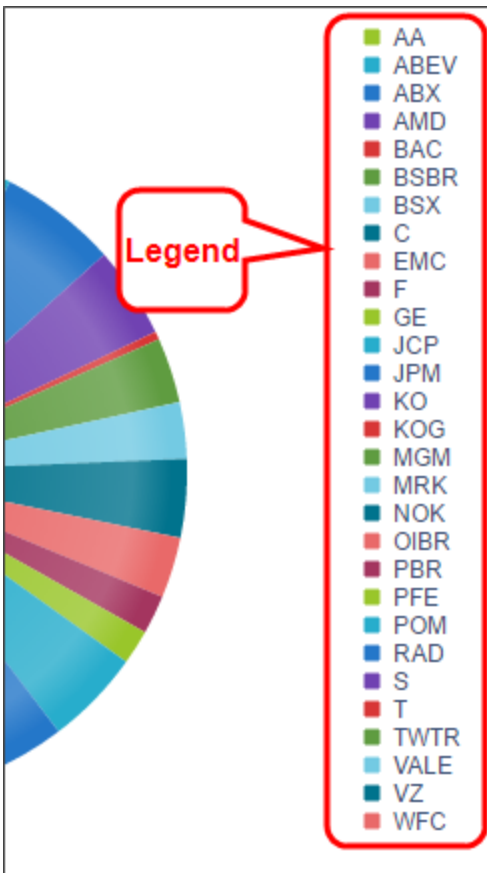
### Chart Type

You can select the type of chart you wish to display from this dropdown control. The available Chart types are:

- Area
- Bar
- Column
- Pie
- Donut
- Line
- Linear Gauge
- Radial Gauge
- For Process Director v 6.0.100 and higher, a new chart type, the Gantt Chart, is also configurable.

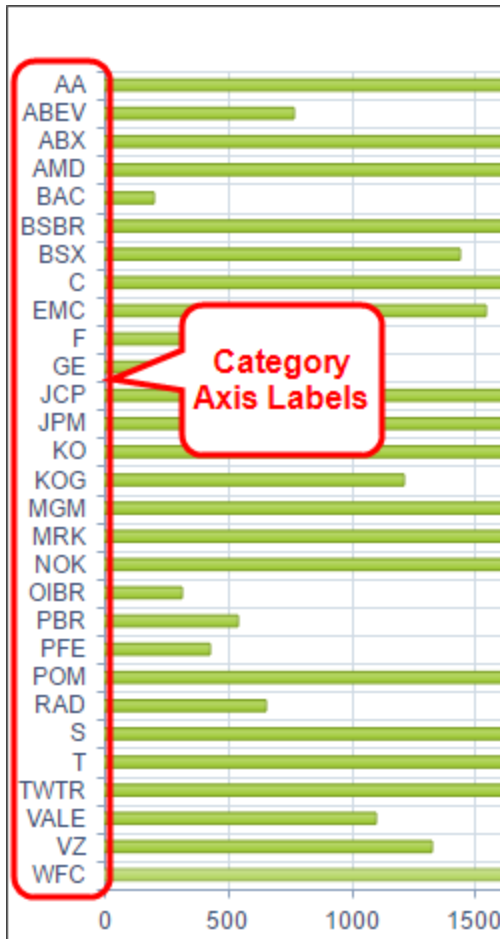
### Show Legend

Checking this box will display the chart legend.



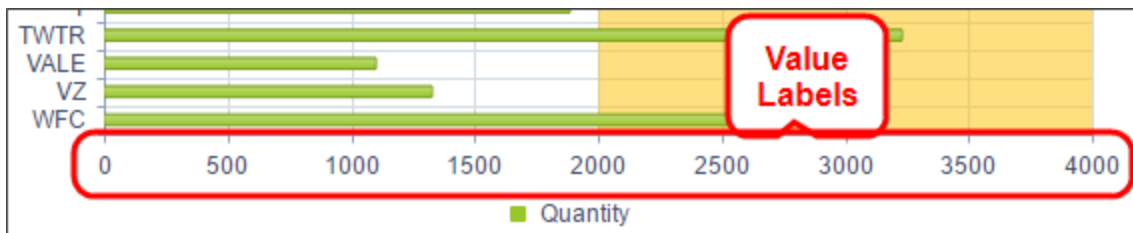
### Show Category Axis Labels

Checking this box will display labels for the category axis.



### Show Value Axis Labels

Checking this box will display labels for the value axis.

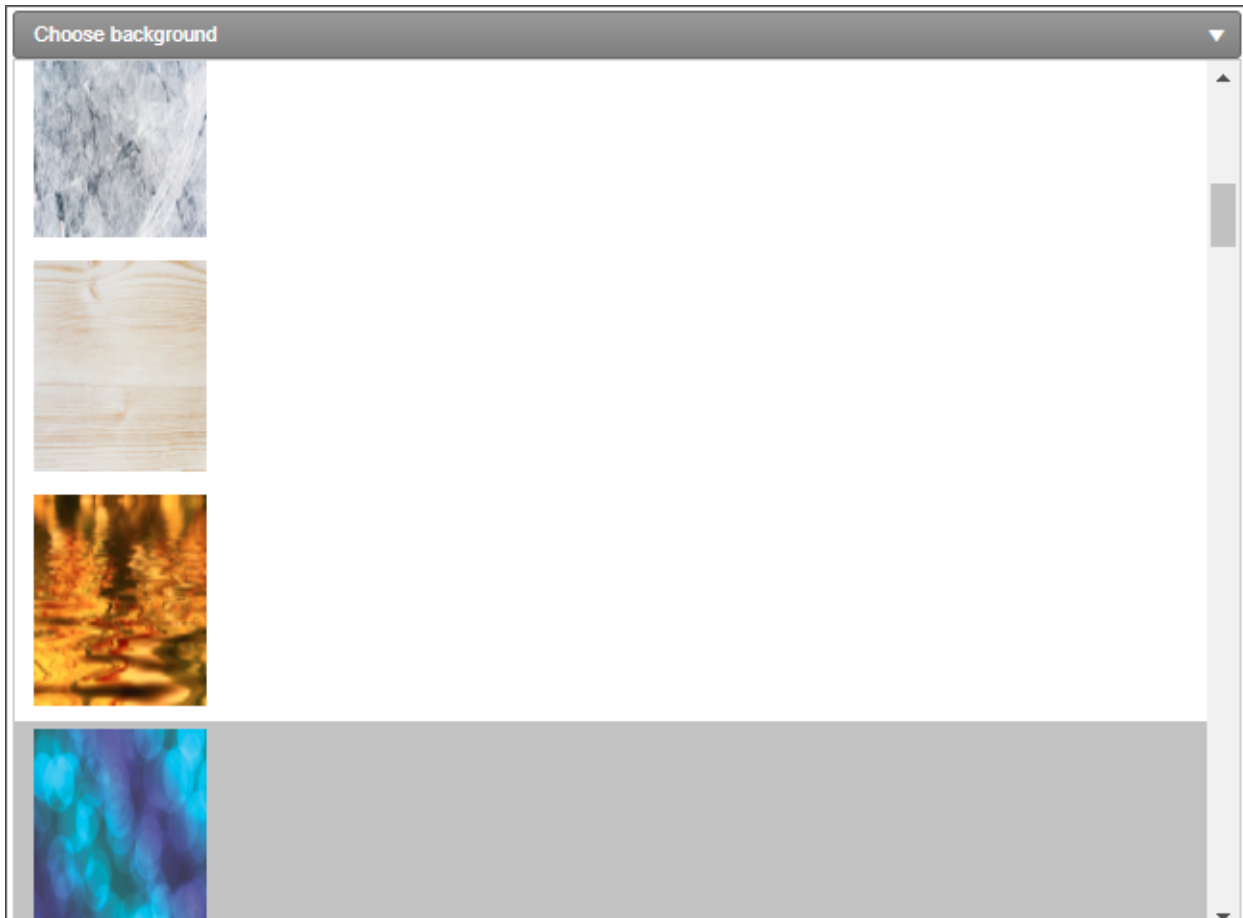


### Select a Content List Background Image for this Chart

You can select an image from the [Content List](#) that will display as the background image for the chart.

### Select a background image from this list for this chart (optional)

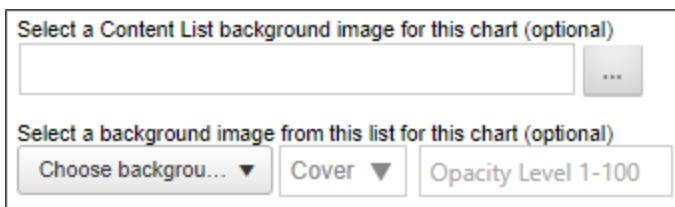
This option enables you to select an image from a dropdown control that contains a number of stock background images, and will use that image as a background image for the Form.



Next to the image dropdown, an additional Background Image Sizing dropdown enables you to specify how the background image will appear on the form. The following options are available:

- Cover: The image will expand to fill the entire page background.
- Center: The image will appear in its actual size, centered on the page vertically and horizontally.
- Tile: The image will appear in its actual size, aligned to the top left of the window, and will tile vertically and horizontally.
- Tile X: The image will appear in its actual size, aligned to the top left of the window, and will tile along the X-axis only.
- Tile Y: The image will appear in its actual size, aligned to the top left of the window, and will tile along the Y-axis only.

Finally, an **Opacity Level** property enables you to set the opacity percentage by setting a value from 0 to 100.






## Additional Body CSS Styling


You can apply CSS styles to the chart to control the visual properties of the chart's body, such as background colors, etc.


## Options Tab #

Chart Name      Icon 

IssuesByTeam

### CHART OPTIONS

Style    Default 

Autosize    Height and Width 

Automatic refresh (seconds)    0

Do not show this object in 'Items I Can Run' Knowledge Views

### Style

You can select a large number of different visual color styles for your chart from this dropdown.

### Autosize

You can set the autosize options for your chart so that it displays properly in different viewports that display in different sizes. The available options are:

- Height and Width
- Height Only
- Width Only
- Fixed Size

If you choose "Fixed Size", **Width** and **Height** settings controls will appear that enable you to set a specific height and width, either as a percentage of the viewport size, or in a fixed number of pixels.

### Automatic refresh (seconds)

You may enter the number of seconds you'd like to pass before the chart auto-refreshes with the most recent data. The default value for this setting is '0'. When the automatic refresh is set to "0", the chart won't auto-refresh.

### Do not show this object in 'Items I Can Run' Knowledge Views

Checking this option will hide the Chart from the [Items I can Run](#) Knowledge View.

## Chart Data Source Tab #

For most [Chart Types](#), Process Director presents a fairly simply set of Data Source Properties.

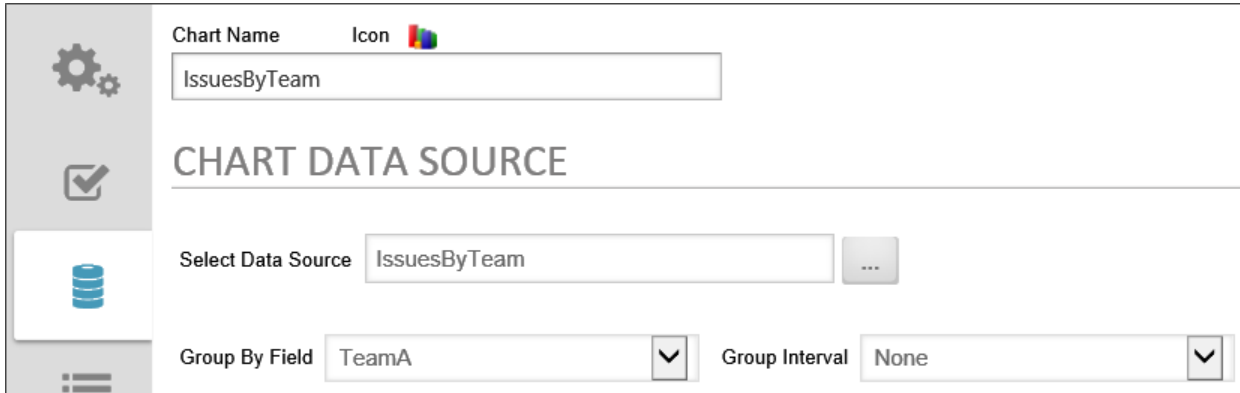


Chart Name Icon

IssuesByTeam

### CHART DATA SOURCE

Select Data Source IssuesByTeam ...

Group By Field TeamA ▼ Group Interval None ▼

This section of the options screen enables you to specify the data source for your chart. You may choose either a Knowledge View or a Business Value as the data source.

### Select Data Source

This [Object Picker](#) enables you to browse through the [Content List](#) to select the data source object for the Chart. You can choose either a Business Value or a Knowledge View as the data source for the Chart.

### Group By Field

Once you've selected a data source, this dropdown control will display all of the columns in the data source. Selecting a column will set that column as the X-Axis of the chart, and will group all of the Chart's Y-Axis values by the column value that you select here.

### Group Interval

If the chart column is date-based, you can set the time interval you'd like to use to add an additional grouping by date. The available options are:

- None
- Minute
- Hour
- Day
- Day of Week
- Week
- Month
- Year
- Quarter

### Show Axis Labels with No Values


For Process Director v5.44.1100 and higher, this property appears when you select a grouping level for which some groupings will produce no values, e.g., grouping by hour, when some hours have no data.

Checking this property box will display the empty group with the appropriate group label and a 0 value for that group on the chart. If this property box isn't checked, the chart will skip that group in the series, and the 0 value won't be displayed on the chart, which is the default behavior.

## Gantt Charts

 This topic is a work in progress for the future release of Process Director v6.0.100. It is subject to change without notice until release.

For Process Director v 6.0.100 and higher, the Gantt chart was added as a selectable **Chart Type**. The Gantt chart has several additional properties on this tab, which specifically enable you to display Gantt charts.

Chart Name Icon 

Gantt Chart

### CHART DATA SOURCE

Select Data Source	Gantt Data	<input style="border: none; background-color: #f0f0f0; padding: 2px 5px;" type="button" value="..."/>
ID	TaskID ▼	Show Column <input type="checkbox"/>
Start	StartDate ▼	
End	EndDate ▼	
Title	TaskName ▼	
Parent	Parent ▼	
Completion	Complete ▼	
Order By	TaskID ▼	
DrillDown Target	[Select Field] ▼	
Scale	Week ▼	

To properly construct a Gantt Chart, the data you use must contain several data fields that can be assigned to each task that will be displayed on the Gantt chart. These properties are associated with the

different data source properties of the Gantt chart via selecting them from this tab. The data source should contain fields that specify the following elements:

- Task or item ID
- Start date
- End date
- Task name or title
- The ID of the parent task
- The completion status of the task
- And, finally, an optional grouping field

Assuming that all of the correct data fields are present, the data source properties below can be fully configured.

### **ID**

The data field used to provide the ID of each task.

### **Show Column**

This checkbox will, when checked, display the Task ID as a column in the Gantt Chart. This property is unchecked by default.

### **Start**

The data field used to provide the start date for each task.

### **End**

The data field used to provide the end date for each task.

### **Title**

The data field used to provide the title or name for each task.

### **Parent**

The data field used to provide the ID for each task's parent task.

### **Completion**

The data field used to provide the completion status for each task. This field's data should be in percentile format, e.g., "75%", rather than in decimal format, e.g., "0.75".

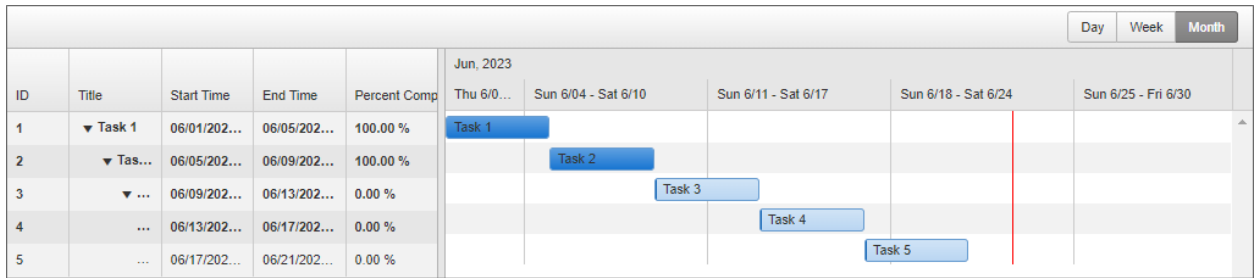
### **Order By**

The data field used to specify the order in which each task is shown, from top to bottom.

### **Scale**

The calendar element you'd like to use for the overall scale of the Gantt chart, e.g., Day, Week, Month, etc.

When properly configured, the Gantt Chart can be displayed with all of its appropriate data.



## Chart Series Tab #

You can add multiple data series to a chart by clicking the **Add Series** button for every data series you'd like to add. The series entries can be ordered by clicking the up/down arrows on the right side of the tab, and a series can be deleted by clicking the red "X" icon.

This section enables you to set the properties of the Chart's values. This section is where you configure the Y-Axes for the chart. You can have multiple series and generate combination charts.

### Data Source Field

The field that contains the Chart's data. This Field is selected from a dropdown control. A condition can also be set to show the series under whatever evaluable condition the user desires.

### Chart Type

The Chart Type setting will only be displayed when the Line, Area, or Column **Chart Types** are selected in the **Chart Options** section. If you choose the Column, Line, or Area chart types in the **Chart Options**

section, then you may choose either Column, Line, or Area as the Chart Type for your chart. You can, therefore, have multiple Y-Axes, in which case you might wish to display each Y-Axis differently. For example, you might wish to have one series display as a column, and a second series display as a line, in order to create a combination chart.

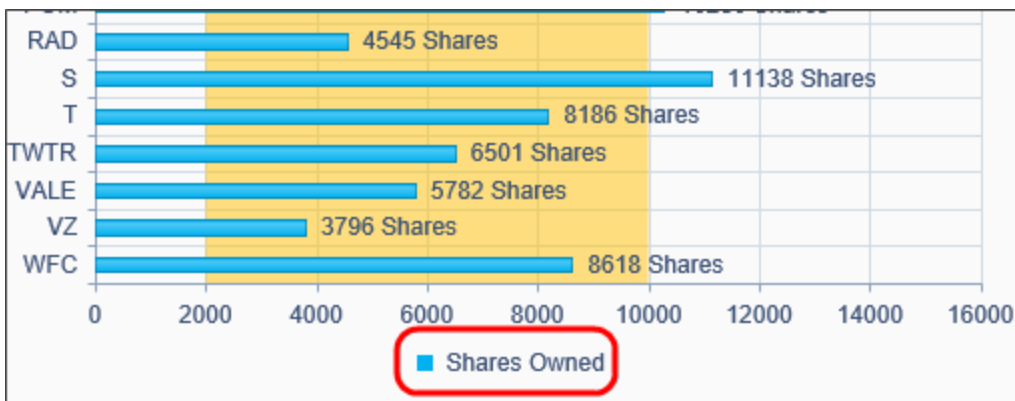
### Aggregate

You can aggregate the values in the Y-Axis by using one of the following mathematical aggregation functions:

- **None:** No aggregation will be applied.
- **Count:** The number of records that match a given Group value.
- **Sum:** The sum of all records that match the Group value.
- **Average:** The average value of all records that match the Group value.
- **Min:** The lowest value contained in all records that match the Group value.
- **Max:** The highest value contained in all records that match the Group value.
- **Average Exclude 0:** The average value of all records that match the Group value, excluding those records that have a zero value from the calculation.
- **First:** The first value contained in all records that match the Group value.
- **Last:** The last value contained in all records that match the Group value.

### Series Name

You can enter a name for the chart series, which will appear as the title of the series in the Chart Legend.



### Series Filter

You can click the [Condition Builder](#) link to create a condition to use to filter the data series.

### Custom Color

You can select a Custom color to display for the series by selecting the custom color from a color selector.

### Axis

This dropdown enables you to set the Value as the primary or a secondary Y-Axis. Charts can have multiple axes, and chart styles with multiple axes can be displayed as combined charts, e.g., column and line charts with primary and secondary axes.

## Stack Type

This setting enables you to set the stack type for stacked bar and column charts. The available stack types are:

- None
- Normal
- 100%

When a stack type other than "None" is selected, the **Group Name** text box will appear, enabling you to label each group type in the stack.

## Missing Values

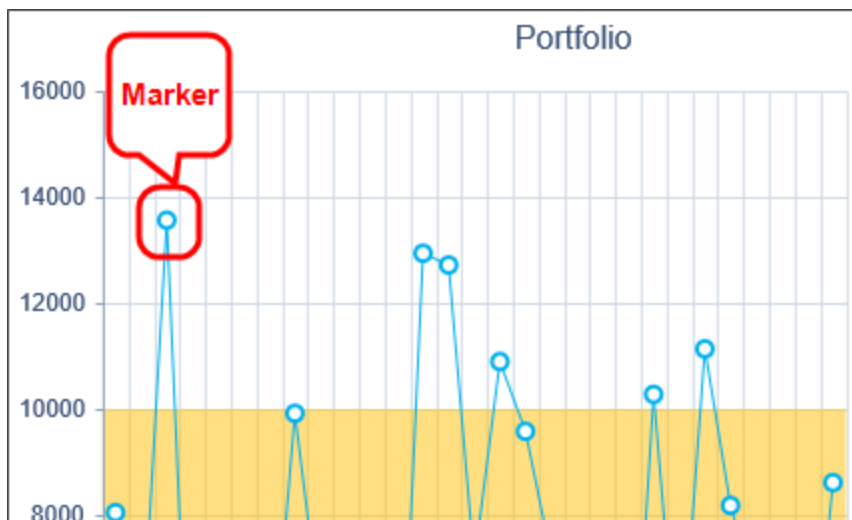
For certain types of chart, such as line charts, this dropdown control provides options on how you'd like the chart to display missing values in the data. The following choices are available:

- **None:** The missing data isn't displayed in the chart, and the chart series skips the display of these data points so that the remaining data points are displayed seamlessly without a gap in the data.
- **Gap:** Display a gap in the chart where the missing data should be.
- **Zero:** Display the missing data as having a value of 0.
- **Interpolate:** Missing data will be displayed as having a value that is the median between the prior and subsequent data points.

## Marker

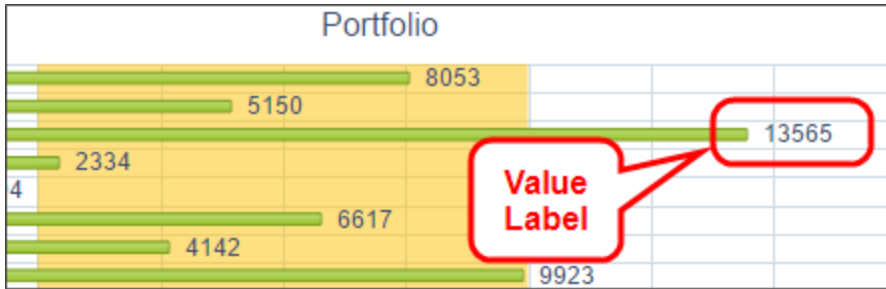
For certain types of chart, such as line charts, , this option enables you to select the shape of the marker denoting each data point. The options for the marker are:

- Circle
- Cross
- Square
- Triangle



## Show Value Labels

Checking this box will display a value label for each data item.



When this setting is checked, a Value Label Template text box appears that enables you to define how the label should appear. The template requires the use of chart syntax to display the values you wish to display. The values are denoted by Chart keywords. The following Keywords are available:

KEYWORD	DESCRIPTION
value	Returns the point value
category	Returns the category name - X Axis value for grouped items.
series.name	Returns the name of the series - X Axis value for non-grouped items.
percentage	Displays the percentage as a decimal between 0 and 1 – Applies to Pie, Donut and Stacked charts only.

Keywords must be surrounded with pound (#) symbols and prefaced with an equal (=) sign:

`#=value#`

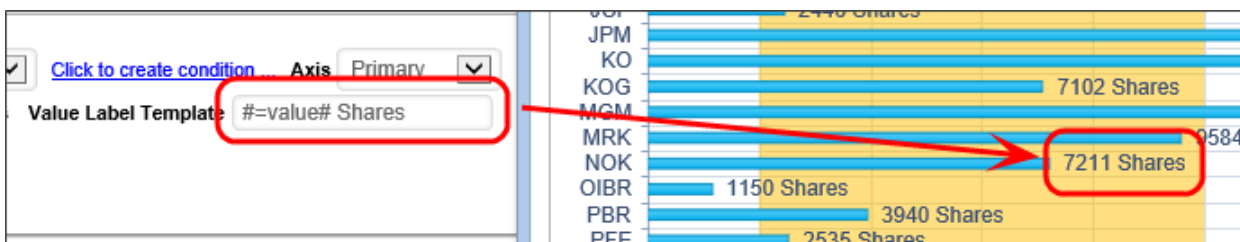
Keywords can be combined with other keywords and non-keyword text to create labels:

`#=value# #=category#(s)`

`#=series.name# -- #=value#`

Use `\n` to create a line break in your label

`#=series.name#: \n #=value#`



Numbers contained in the series label can use Telerik's [Kendo Formatting extensions](#) to specify the number's format. Some examples of the syntax might be:



FORMAT	DESCRIPTION
<code>#=value.format("N3")#</code>	Produces a number with 3 decimal places
<code>#=value.format("C0")#</code>	Produces a currency value with 0 decimal places
<code>#=kendo.format('\{0:p2}\',percentage)#</code>	Formats the percentage value as a percentage with 2 decimal places

## Available Formats

FORMAT	DESCRIPTION
N	Number (e.g. the number 10500 will be formatted as 10,500.00).
C	Currency (e.g. the number 10500 will be formatted as \$10,500.00) note: Process Director automatically formats all values stored as currency using the "C" identifier.
P	Percentage (e.g. the number 10500 will be formatted as 1,050,000.00% because 1 is equal to 100%)
E	Exponential (e.g. the number 10500 will be formatted as 1.05e+4)

## Conditional Logic

You can use if statements to apply label values conditionally with the following syntax:

```
#if (condition) {# expression #} else {# expression #}#
```

### Example

```
# if ( value == "0" ){# No Hours #} else {# #=value# Hours #}#
```

If the value is "0" the label displays "No Hours"; otherwise, the value displays as "X Hours", where X is the number of hours.

## Other Documentation

For more information about Kendo UI Templates, see [Telerik's documentation on the Kendo UI](#).

### Show Tooltip

You can display a tooltip that will appear when users mouse over the chart. If you select this option, the [Tooltip Template](#) text box will appear, into which you can enter the template for the tooltip's text, just as you did with [Value Labels](#).

### Drill down target

You can set a drill down target that will appear when you click on the data series. You can choose to show the drill down data in a content object such as a Knowledge View, or at a custom URL. If you choose a Content Object for the drill-down data, the Content Picker will appear to enable you to select the desired object. Similarly, if you choose a custom URL, a text box will appear, into which you can type the URL for the Drill down target. For users of Process Director v5.26 or higher, the drilldown URL can contain system variables, which will be parsed and replaced by the appropriate text at run-time.

For Gantt Charts in Process Director v6.0.205 and higher, the [Drill-Down Target](#) is generated automatically by referencing the Form Instance from which the data is generated.

Additionally, you can direct where the drill down object will appear by selecting it from the **Drill down opens** dropdown that offers the following options:

- **Popup:** A popup window will display the drill down object.
- **Replace:** The current chart will be replaced in its location by the drilldown object.
- **Above:** A split screen will appear, displaying the drill down object above the original chart.
- **Below:** A split screen will appear, displaying the drill down object below the original chart.
- **Left:** A split screen will appear, displaying the drill down object to the left of the original chart.
- **Right:** A split screen will appear, displaying the drill down object to the right of the original chart.
- **Portlet 1:** If the original chart appears in a workspace portlet, the object in Portlet 1 will be replaced by the drill down object.
- **Portlet 2:** If the original chart appears in a workspace portlet, the object in Portlet 2 will be replaced by the drill down object.
- **Portlet 3:** If the original chart appears in a workspace portlet, the object in Portlet 3 will be replaced by the drill down object.
- **Portlet 4:** If the original chart appears in a workspace portlet, the object in Portlet 4 will be replaced by the drill down object.



If you select a portlet that doesn't exist in the workspace, the drill down object will appear in a popup window.

When using the Chart in a Dashboard instead of a Workspace, the "portlet" selections above can also be used to open a Dashboard portlet as the target by configuring the **Name** property of the Dashboard portlet to match the portlet you select in the **Drill down target** property of the chart series. For example, if you select the "Portlet 1" selection from the **Drill down Target** dropdown, you can change the **Name** property of another Dashboard portlet to "Portlet 1" and the renamed portlet will be used as the dropdown target for the chart series.

When using the drill-down feature of a chart series, the Chart object can pass a series of variables to the drill-down object. For example, you may want to use a Knowledge View as a Drill-Down object, so that when you click on a chart's data series, Process Director opens a Knowledge View containing a logical view of the series data. In order for this to work properly, you must pass the appropriate variables to the Knowledge View's to filter the data. The Chart, therefore, passes the following variables to drill-down objects.

**bpcCategory:** The Category of a Data Series, taken from the Group By field, if any.

**bpcSeriesType:** The Series Type of a chart Data Series, i.e., Bar Column, Pie, etc.

**bpcSeriesName:** The Series Name of a chart Data Series. This will be the name of the series from the **Series Name** property if you've given the series a name, or the field name of the series if you've not entered a **Series Name**. This is the value that is displayed in the Chart's Legend for the series.

**bpcValue:** The value of the Data Series.

**bpcPercentage:** The percentage value of the Data Series in decimal format. This variable is only available with Pie, Doughnut and 100% Stacked charts.

**!** Sometimes, variables must be passed through multiple objects before reaching a Knowledge View, or other object. For instance, you may have a Form that calls a Chart, which, in turn, calls a Knowledge View via a drill-down. The data on the Form is passed to the Chart via a variable that is accessed using the normal `{VAR:ParmName}` syntax. However, when the chart does a drill-down to a Knowledge View, the Knowledge View does not have access to the original `{VAR:ParmName}` variable. Instead you need to append two underscores and the number "2" (`_2`) to the variable name, e.g. `{VAR:ParmName__2}`. When you do so, Process Director treats the "`{VAR:ParmName__2}`" in the Knowledge View as a pointer to the original value for `{VAR:ParmName}`, so that the proper value is automatically available to the Knowledge View.

### *Drill-down Variables Implementation Example*

Let's assume that we want to use a Knowledge View as the data source for our charts. Any of the variables listed above can be used directly in the **Filter** tab of a Knowledge View as variables. Here's a simple example:

Location	=	String	{VARIABLE:bpcCategory}	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
[AND]						
Vehicle	=	String	{VARIABLE:bpcSeriesName}	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

The Chart will pass the variable values automatically, and a simple Knowledge View will return the correct data based on the filters that get passed via the variables.

But, now let's look at a more complex scenario. For this next example, let's assume we have a Knowledge View that returns some information about a fleet of vehicles. As part of this information, the Knowledge View has a **Mileage** column for each vehicle the Business Value returns. This **Mileage** column gets its data from a Business Value that returns the mileage for a specific vehicle. The Business Value has two parameters, the Location and the Vehicle, from which it determines what vehicle to use to return that vehicle's mileage to the Knowledge View column. The underlying Business Value for the **Mileage** column might be configured like this:

### CONFIGURE

SQL Data Source
Knowledge View Data Source
REST Data Source

Please Choose Data Source

...
Test Connection

Database Table

```

SELECT * FROM Sample_Vehicle_Usage_Form
) Vehicle
WHERE Vehicle.bp_Location = '{PARAMETER:Location}'
AND Vehicle.bp_Vehicle = '{PARAMETER:Vehicle}'
                    
```

▼
Hide Columns

Test Query

---

Properties (1 Property)

Property Name	Property Type	Property Value
<input type="text" value="Mileage"/>	<input type="text" value="Database Column"/> ▼	<input type="text" value="nMileage"/>

Add Property

---

Parameters (2 Parameters)

Name	Test Value
<input type="text" value="Location"/>	<input type="text"/> ✖
<input type="text" value="Vehicle"/>	<input type="text"/> ✖

Add Parameter

So, how do we make this return the proper data to the Knowledge View's **Mileage** column?

In the **Columns** tab of the Knowledge View, when you configure the Business Value to use for the **Mileage** column, you can use a Chart Variable as one of the Business Value's parameters, as shown below.

Knowledge View Name

### COLUMNS

This section of the Knowledge View creation allows you to define which columns are displayed in the view

Second line for each entry

Default Column Sort Order:  ▼  ▼

Column Name	Column Data	When
<input type="text" value="Location"/>	<input type="text" value="Location"/>	Alw
<input type="text" value="Vehicle"/>	<input type="text" value="Vehicle"/>	Alw
<input type="text" value="Mileage"/>	<input type="text" value="Sample Total Mileage Business Value.Mileage"/>	Alw
<input type="text" value="Show Location"/>	<input type="text" value="Location"/>	Alw
<input type="text" value="Show Series Name"/>	<input style="font-family: monospace; font-size: small;" type="text" value="{VARIABLE:bpcSeriesName}"/>	Alw
<input type="text" value="Show BV"/>	<input style="font-family: monospace; font-size: small;" type="text" value="{BUSINESS_VALUE:Sample Total Mileage E"/>	Alw

Choose System Variable

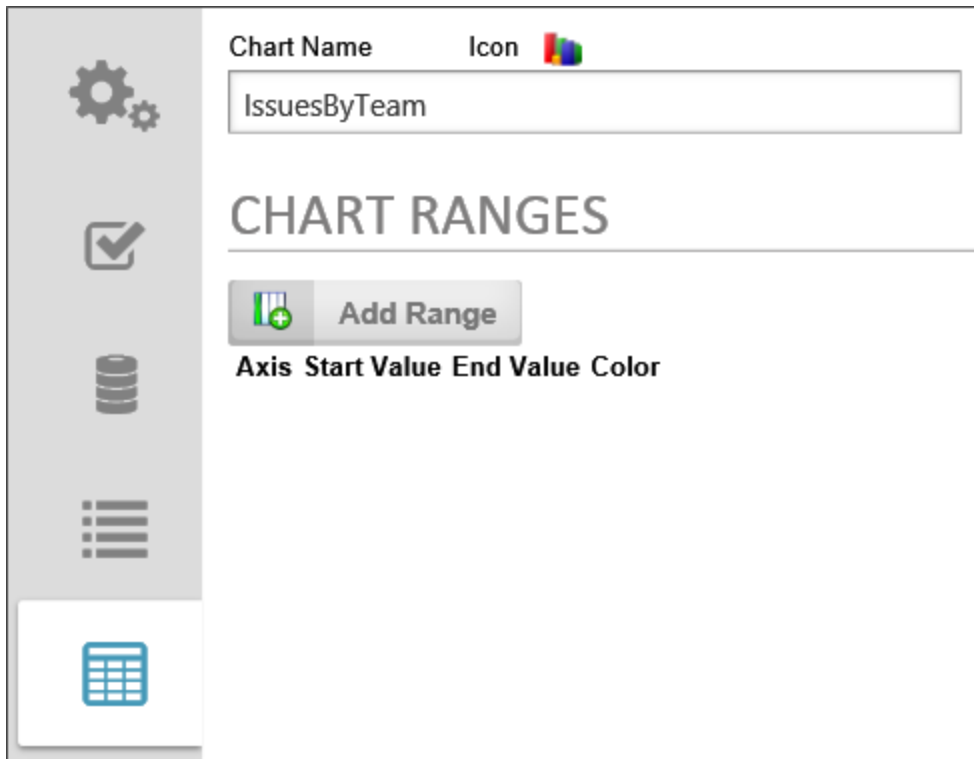
Location

Vehicle

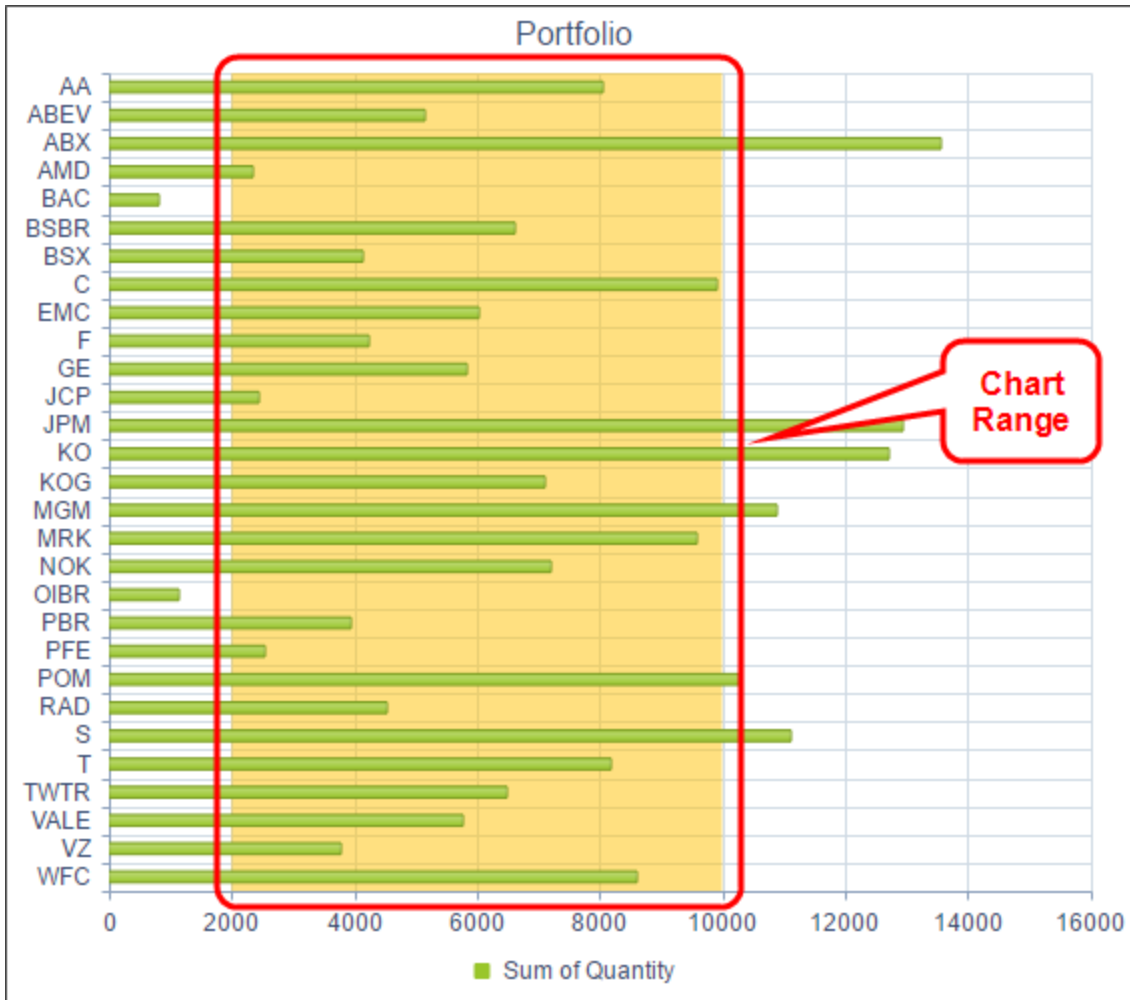
The Chart will pass the `bpcSeriesName` variable to the Knowledge View, which will, in turn, pass it along to the underlying Business Value as a parameter, so that the mileage for the specified vehicle is returned from the Business Value to populate the `Mileage` column.

You can add additional series from your data source by clicking the `Add Series` button.

### Chart Ranges Tab #



This configuration section enables you to modify the chart display to add different colors to specified ranges of the chart's background.



### Axis

Enables you to set the range on the primary or secondary axis.

### Start Value

The value that begins the range.

### End Value


The Value that ends the range

### Color

Enables you to set the color for the range.



Additional ranges can be added by clicking the [Add Range](#) button. The range entries can be orders by clicking the up/down arrows on the right side of the tab, and a range can be deleted by clicking the red "X" icon.

## Dashboards















Dashboard Object Name      Icon 

test ✕





### DASHBOARD DEFINITION





**Widgets**

-  Text
-  Html
-  URL
-  Chart
-  KView
-  Global KView
-  Form
-  Form Instance
-  Report
-  Document
-  Image
-  Button
-  Clock
-  Activity Journal

**Move Widgets**

-  Snap to top
-  Snap to left
-  Snap to right
-  Snap to bottom

**Size Widgets**

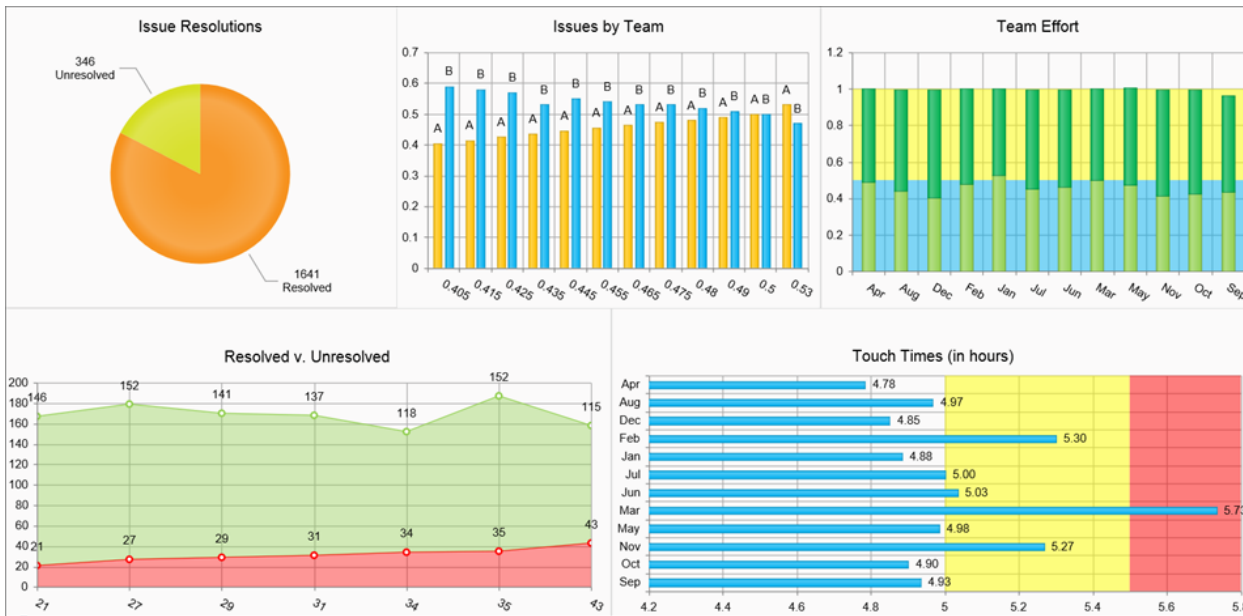
-  Full width
-  20% width
-  Full height
-  20% height

Users of Process Director v4.02 and higher have access to the Dashboard object. The Dashboard Object enables you to create a custom screen that displays existing Charts, KViews, Forms and other objects. The following objects may be displayed in a Dashboard:

- Text
- Custom HTML
- A web page, identified by a URL
- Chart
- Knowledge View
- Global Knowledge View
- Form
- Form Instance
- Report
- Document
- Image
- Button
- Clock
- Activity Journal

Please refer to the [Dashboard Widgets](#) topic for detailed information about configuring the properties for each type of widget.

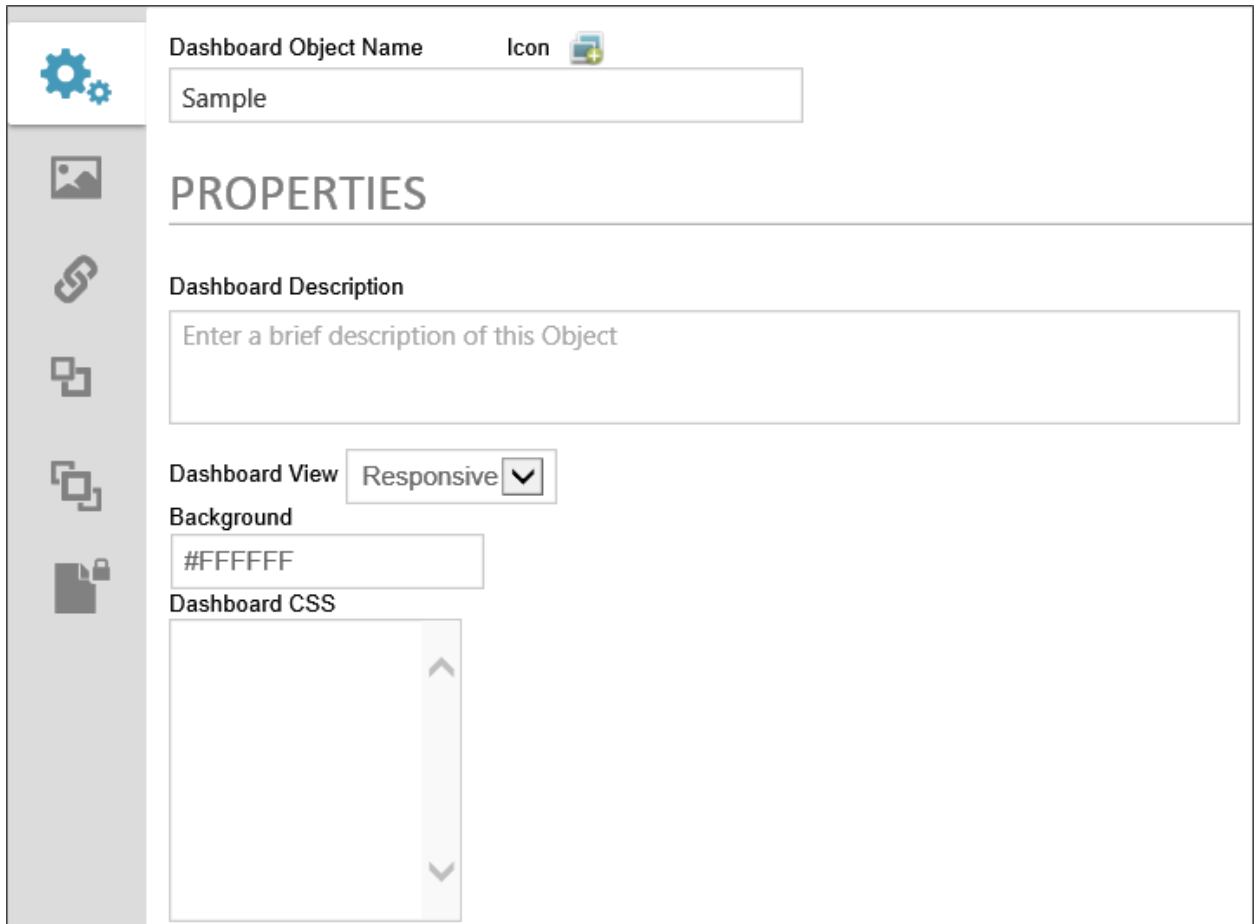
Using the Dashboard, designers can configure the number, size, and placement of the objects they wish to display, placing each object in a Widget. Objects can be placed in a dashboard asymmetrically as well. In addition, buttons can be placed in a dashboard to navigate to other dashboards, open Forms, display Knowledge Views, etc.



## Properties Tab

Clicking the **Properties** tab of the Dashboard definition will display the general settings for the dashboard.





### Icon

The icon that will be used for the dashboard. Clicking on the icon will open the [Icon Chooser](#) window, Selecting an icon from the dialog will set it as the Dashboard's icon.

### Dashboard Object Name

The name that will appear in the [Content List](#).

### Dashboard Description

A brief description of the dashboard's purpose.

### Dashboard View

You have the option of viewing the dashboard in responsive format, which will make objects automatically resize to the size of the viewport, or in Fixed-Size format. If you choose the fixed-size format, you'll be presented with **Height** and **Width** input boxes that enable you to set the height and width, in pixels, of the dashboard object.



### Background

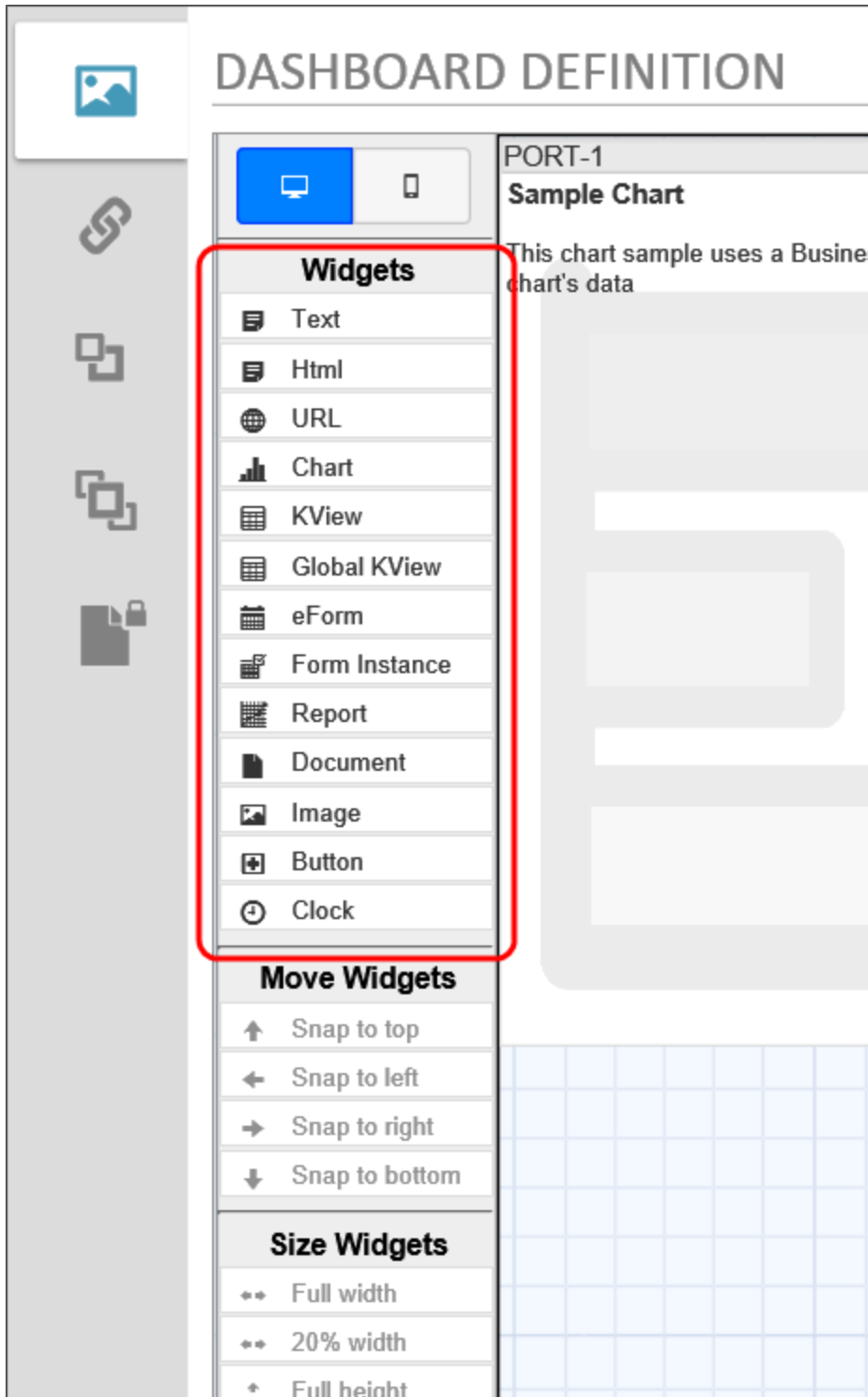
A named HTML or hexadecimal color that will display in the Dashboard's background.

### Dashboard CSS

A CSS stylesheet that you can create for the Dashboard. You can create named styles and apply those styles to other dashboard elements in the element's **CSS Class** property.

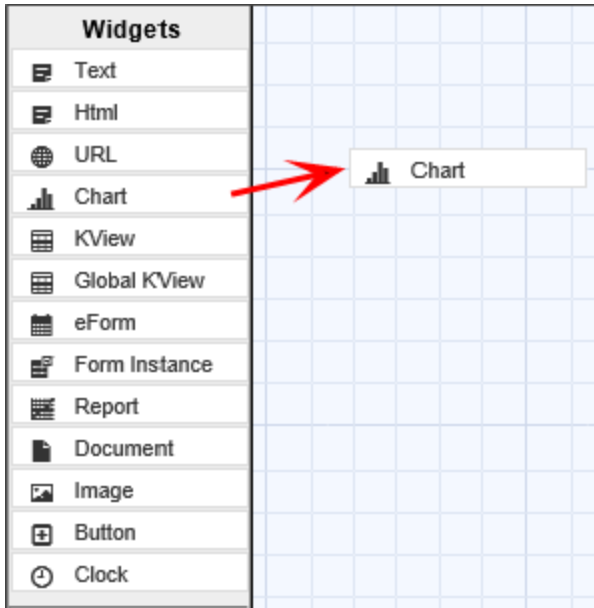
### Dashboard Definition Tab

Objects are added to the Dashboard design space—which appears as a sheet of graph paper—by creating Widgets. The available Widgets are listed in the **Widgets** toolbar of the **Dashboard Definition** Tab.

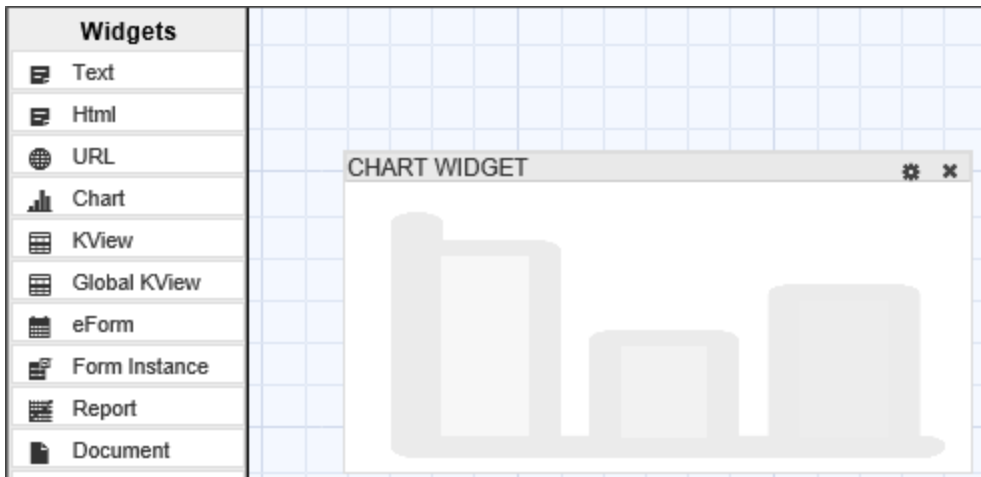


While some Widgets, like the Text or Button widgets, are configured via simple text configuration, most Widgets must contain an existing Process Director object. That is to say that you must first create the charts, Knowledge Views, Reports, etc. that you wish to display in the Dashboard prior to creating the Dashboard object. To create a Widget in the design space, simply drag the desired object type from the

Widgets toolbar and drop it onto the design space. As you drag the Widget, a cursor the shows the type of Widget you are dragging will follow your mouse drag to the desired location in the design space.



Once you release mouse button and drop Widget on the design space, the appropriate placeholder for the Widget will appear.



**i** As you drag Widgets onto the design surface, or move/resize Widgets, you may find that a Widget is partially occluded by another one. You can hover the mouse over a partially occluded Widget to bring it to the front for easier editing.

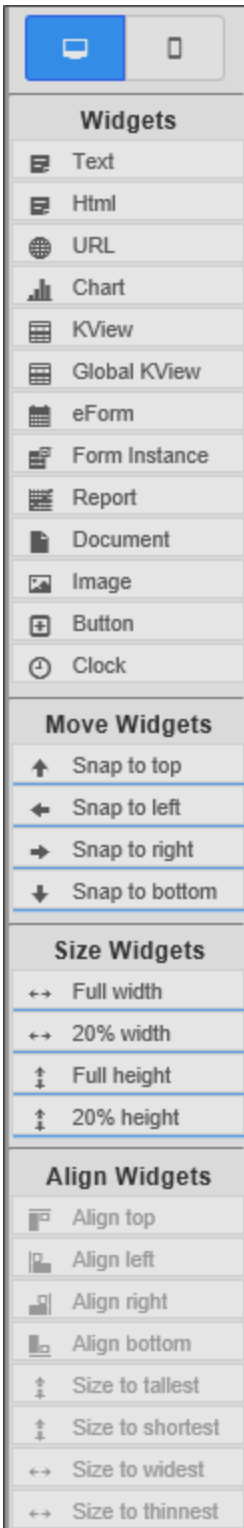
## Configuring the Widget

Each Widget can be repositioned by using the title bar to drag and drop the Widget to a new location. Widgets can also be resized by dragging and dropping the right or bottom border of the Widget. The Widget's title bar displays the name of the Widget on the left side, and displays two icons on the left.



- The **Settings** icon will open the properties dialog box for the Widget.
- The **Delete** icon will remove the Widget from the design space.

Clicking the **Select** Icon enables the **Moving** and **Sizing** options. Multiple Widgets can be selected at the same time, which enables the **Align Widgets** options.



Clicking the title bar of a Widget selects it and highlights the Widget or deselects it if it has already been selected. Selecting a Widget activates the Move Widgets and Size Widgets options on the toolbar. Selecting multiple Widgets activates the Align Widgets options, which enable you to align and size multiple widgets.

The Move Widgets settings will automatically snap the selected Widget to the selected location on the design space. The Size Widgets settings will automatically resize the Widget to the selected size. The Align Widgets settings will snap the selected widgets to the specified alignment or size with respect to each other.

## Widget Properties

You can open the [Widget Properties](#) dialog box for a Widget by clicking the [Settings](#) icon.

### *Show Title Bar*

You can hide the title bar for a Widget by unchecking the [Show title bar](#) property.

### *Display*

The [Display](#) option enables you to select when you wish the widget to display. The available options are:

- **Always:** Show in all viewports.
- **Desktop Only:** Only show in Desktop viewports.
- **Mobile Only:** Only show on mobile device viewports.

### *Title*

The text displayed in the title bar can be set by editing the [Title](#) property. (A default title is given to each Widget when it is created.)

### *Refresh (Seconds)*

You can set the widget to refresh its display regularly at the interval, in seconds, that you set in this property.

### *CSS Class*

If you have a custom CSS stylesheet, you can specify the CSS class you'd like to use for the display of the widget.

### *Name*

An Name/ID you can set to refer to the widget in a script or other objects.

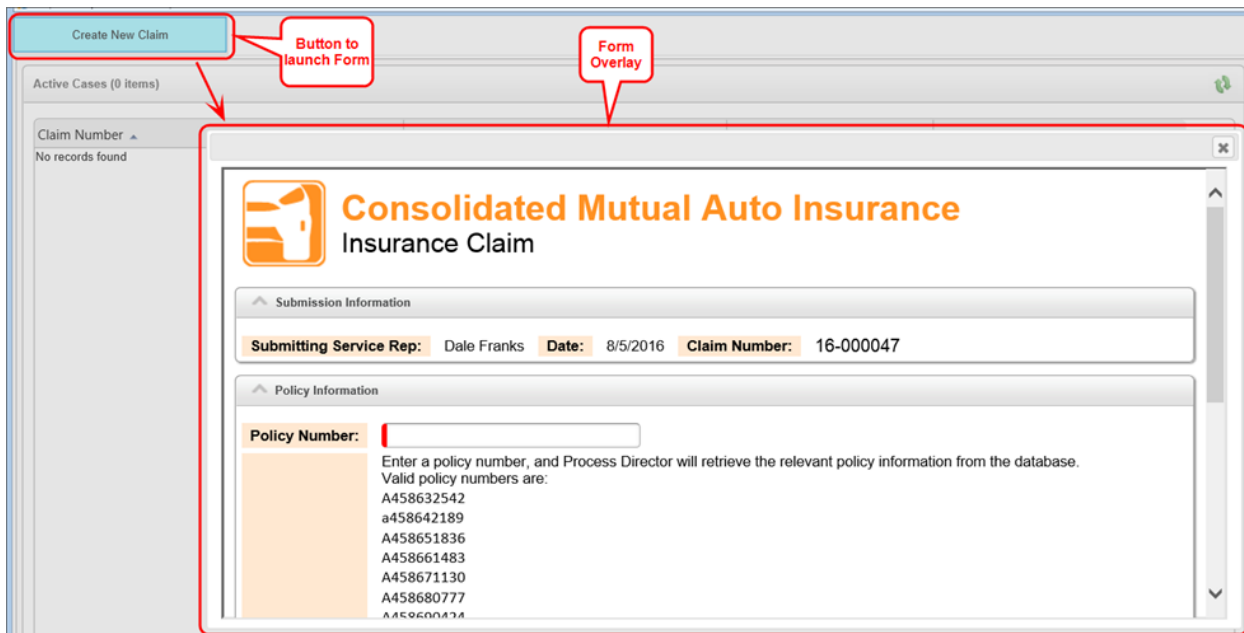
### Choose [Object]

A **Content Picker** control enables you to select the Process Director object that is displayed in the Widget. Once you've finished setting the Widget properties, you can save and close the Properties dialog box by clicking the **Apply** button. Clicking the **Cancel** button will discard your changes and close the dialog box. All of the Widgets in the Dashboard will automatically resize themselves as the size of the dashboard window changes.

**i** Process Director v5.42 and higher displays a Minimize/Maximize icon for Dashboard widgets that are configured to display the Header Bar.

## Launching Forms

You can launch a Form from a Dashboard object. Dashboard buttons can be used to launch Forms, and Knowledge Views can be used to display Form instances. When a Form is launched from a dashboard object, the Form will appear in an overlay on the Dashboard.



The Form that appears in the overlay is fully functional, and will operate just as it would if it were opened outside a dashboard.

When a Form instance is displayed in the Dashboard using the Form Instance widget, Process Director will automatically find the task for the Form Instance, if any, that is associated with the current user, and will display the form in task context, i.e., as if the Form was opened from the [Task List](#).

## Dashboard Widgets

The Dashboard object enables you to display a number of Widget objects as portlets in the Dashboard. Each type of widgets has a set of properties, some of which are specific to that widget. The various Widgets and their configurable properties, are described below.



## Text #

The Text widget enables you to place arbitrary HTML-formatted text into a widget.

### Show Title Bar

Unchecking this check box will hide the title bar for the widget on the Dashboard. The default value is Checked.

### Display

This dropdown enables you to choose when to display the widget. The default setting is "Always". The available options are:

- Always
- Desktop Only
- Mobile Only

### Title

This text box enables you to specify the title that appears on the widget's Title Bar.

### Refresh (Seconds)

This setting enables you to specify the number of second that can elapse before the widget refreshes automatically. The default setting is "0", which means the widget won't refresh.

### CSS Class

If you are using a custom style sheet, you can specify a style name from that style sheet with this setting.

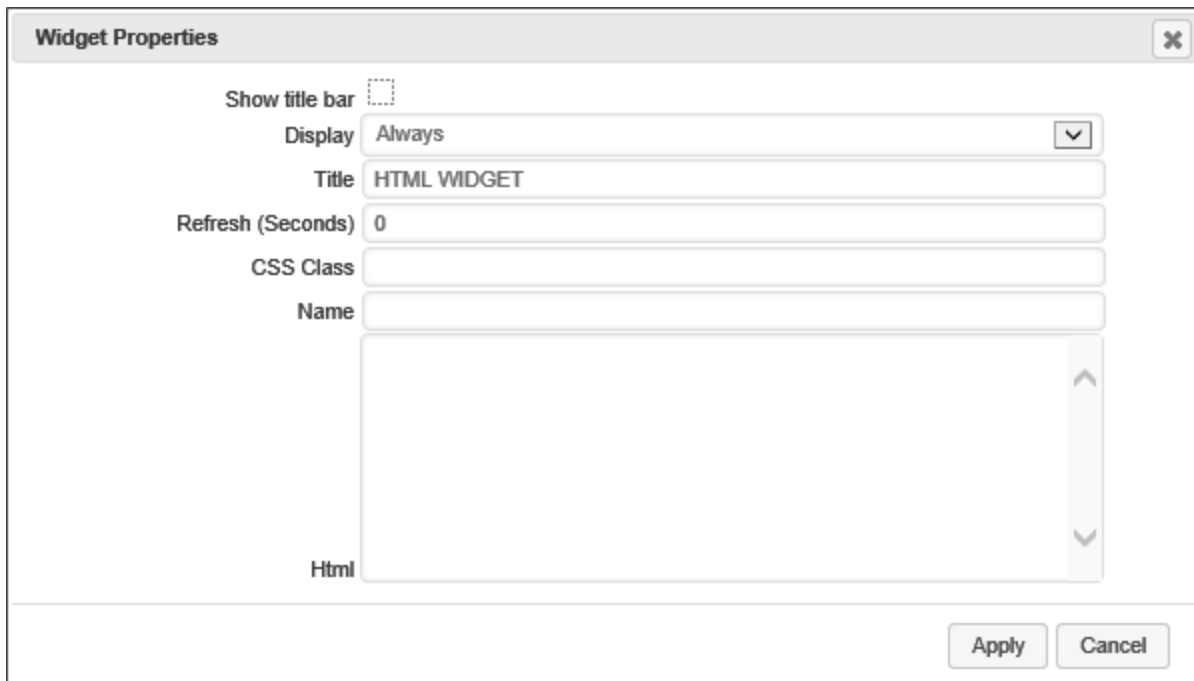
## Name

An object name that you can use to refer to the widget programmatically, or from other elements in the UI.

The bottom portion of the Widget Properties box contains a rich-text editor into which you can type and format the text you wish to display in the widget.

## HTML #

This widget enables you to enter custom HTML to display in the widget.



The screenshot shows a dialog box titled "Widget Properties" with a close button (X) in the top right corner. The dialog contains several settings:

- Show title bar**: A checked checkbox.
- Display**: A dropdown menu set to "Always".
- Title**: A text box containing "HTML WIDGET".
- Refresh (Seconds)**: A text box containing "0".
- CSS Class**: An empty text box.
- Name**: An empty text box.
- Html**: A large text area for entering HTML content, with a vertical scrollbar on the right side.

At the bottom right of the dialog are two buttons: "Apply" and "Cancel".

### Show Title Bar

Unchecking this check box will hide the title bar for the widget on the Dashboard. The default value is Checked.

### Display

This dropdown enables you to choose when to display the widget. The default setting is "Always". The available options are:

- Always
- Desktop Only
- Mobile Only

### Title

This text box enables you to specify the title that appears on the widget's Title Bar.

### Refresh (Seconds)

This setting enables you to specify the number of second that can elapse before the widget refreshes automatically. The default setting is "0", which means the widget won't refresh.

### CSS Class

If you are using a custom style sheet, you can specify a style name from that style sheet with this setting.

### Name

An object name that you can use to refer to the widget programmatically, or from other elements in the UI.

### HTML

This textarea control accepts the raw HTML code that you wish to display.

### URL #

This widget displays the web page that appears at the URL you specify.

### Show Title Bar

Unchecking this check box will hide the title bar for the widget on the Dashboard. The default value is Checked.

### Display

This dropdown enables you to choose when to display the widget. The default setting is "Always". The available options are:

- Always
- Desktop Only
- Mobile Only

### Title

This text box enables you to specify the title that appears on the widget's Title Bar.

### Refresh (Seconds)

This setting enables you to specify the number of second that can elapse before the widget refreshes automatically. The default setting is "0", which means the widget won't refresh.

## CSS Class

If you are using a custom style sheet, you can specify a style name from that style sheet with this setting.

## Name

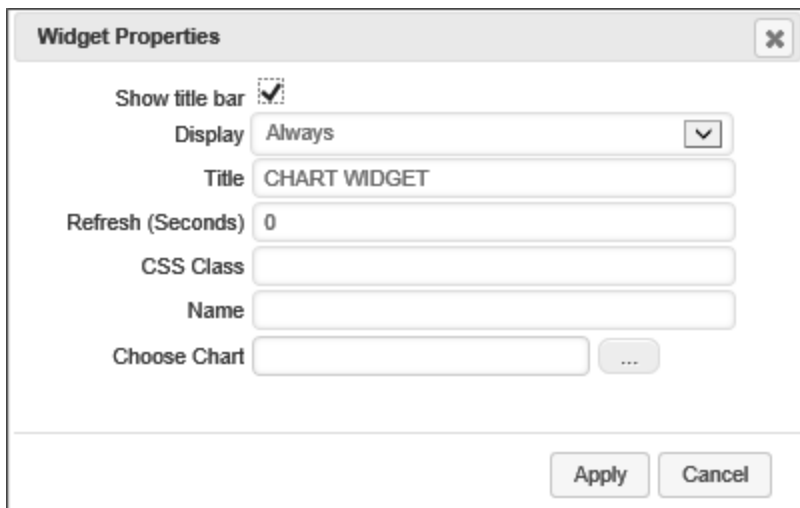
An object name that you can use to refer to the widget programmatically, or from other elements in the UI.

## URL

This is the page URL of the web page you wish to display.

## Chart #

This widget will display a Process Director Chart object.



The screenshot shows a 'Widget Properties' dialog box with the following fields and controls:

- Show title bar**: A checked checkbox.
- Display**: A dropdown menu currently showing 'Always'.
- Title**: A text input field containing 'CHART WIDGET'.
- Refresh (Seconds)**: A text input field containing '0'.
- CSS Class**: An empty text input field.
- Name**: An empty text input field.
- Choose Chart**: An empty text input field followed by a button with three dots.
- Buttons**: 'Apply' and 'Cancel' buttons at the bottom right.

### Show Title Bar

Unchecking this check box will hide the title bar for the widget on the Dashboard. The default value is Checked.

### Display

This dropdown enables you to choose when to display the widget. The default setting is "Always". The available options are:

- Always
- Desktop Only
- Mobile Only

### Title

This text box enables you to specify the title that appears on the widget's Title Bar.

### Refresh (Seconds)

This setting enables you to specify the number of second that can elapse before the widget refreshes automatically. The default setting is "0", which means the widget won't refresh.

### CSS Class

If you are using a custom style sheet, you can specify a style name from that style sheet with this setting.

### Name

An object name that you can use to refer to the widget programmatically, or from other elements in the UI.

### Choose Chart

An object picker that enables you to select the Chart that you wish to display.

## KView #

This widget will display a Process Director Knowledge View object.

### Show Title Bar

Unchecking this check box will hide the title bar for the widget on the Dashboard. The default value is Checked.

### Display

This dropdown enables you to choose when to display the widget. The default setting is "Always". The available options are:

- Always
- Desktop Only
- Mobile Only

### Title

This text box enables you to specify the title that appears on the widget's Title Bar.

### Refresh (Seconds)

This setting enables you to specify the number of second that can elapse before the widget refreshes automatically. The default setting is "0", which means the widget won't refresh.

## CSS Class

If you are using a custom style sheet, you can specify a style name from that style sheet with this setting.

## Name

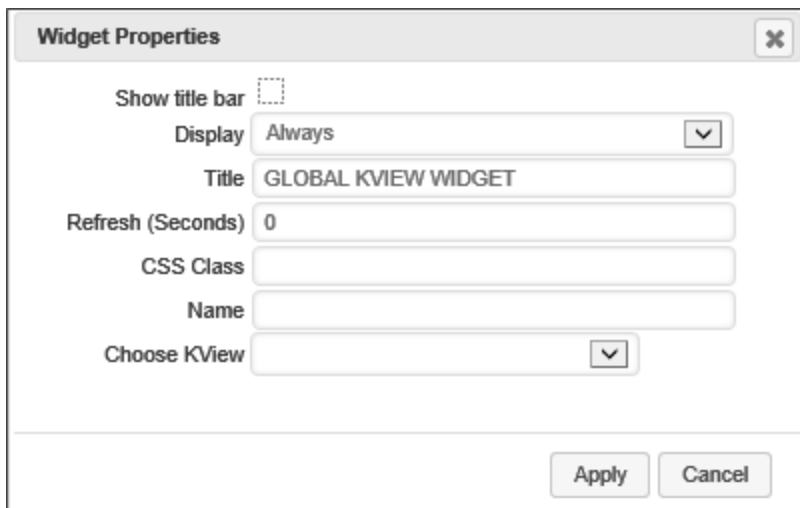
An object name that you can use to refer to the widget programmatically, or from other elements in the UI.

## Choose KView

An object picker that enables you to select the Knowledge View that you wish to display.

## Global KView #

This widget will display a Process Director Global Knowledge View object.



The screenshot shows a 'Widget Properties' dialog box with the following fields and values:

- Show title bar:
- Display: Always (dropdown)
- Title: GLOBAL KVIEW WIDGET (text box)
- Refresh (Seconds): 0 (text box)
- CSS Class: (empty text box)
- Name: (empty text box)
- Choose KView: (empty dropdown)

Buttons: Apply, Cancel

## Show Title Bar

Unchecking this check box will hide the title bar for the widget on the Dashboard. The default value is Checked.

## Display

This dropdown enables you to choose when to display the widget. The default setting is "Always". The available options are:

- Always
- Desktop Only
- Mobile Only

## Title

This text box enables you to specify the title that appears on the widget's Title Bar.

## Refresh (Seconds)

This setting enables you to specify the number of second that can elapse before the widget refreshes automatically. The default setting is "0", which means the widget won't refresh.

## CSS Class

If you are using a custom style sheet, you can specify a style name from that style sheet with this setting.

## Name

An object name that you can use to refer to the widget programmatically, or from other elements in the UI.

## Choose KView

An object picker that enables you to select the Global Knowledge View that you wish to display.

## Form #

This widget will display a Process Director Form as a new form instance.

## Show Title Bar

Unchecking this check box will hide the title bar for the widget on the Dashboard. The default value is Checked.

## Display

This dropdown enables you to choose when to display the widget. The default setting is "Always". The available options are:

- Always
- Desktop Only
- Mobile Only

## Title

This text box enables you to specify the title that appears on the widget's Title Bar.

## Refresh (Seconds)

This setting enables you to specify the number of second that can elapse before the widget refreshes automatically. The default setting is "0", which means the widget won't refresh.

### CSS Class

If you are using a custom style sheet, you can specify a style name from that style sheet with this setting.

### Name

An object name that you can use to refer to the widget programmatically, or from other elements in the UI.

### Choose Form

An object picker that enables you to select the Form that you wish to display.

### Form Instance #

This widget will display the most recent form instance of a selected Form Definition.

The screenshot shows a dialog box titled "Form Instance Widget Properties" with a close button in the top right corner. The dialog contains the following fields and controls:

- Show title bar**: A checkbox that is currently checked.
- Display**: A dropdown menu with "Always" selected.
- Title**: A text input field containing "FORM INSTANCE WIDGET".
- Refresh (Seconds)**: A text input field containing "0".
- CSS Class**: An empty text input field.
- Name**: An empty text input field.
- Choose Form**: An empty text input field followed by a button with three dots.
- Form Instance**: A dropdown menu with "Latest Form Instance for selected Form" selected.

At the bottom right of the dialog are two buttons: "Apply" and "Cancel".

### Show Title Bar

Unchecking this check box will hide the title bar for the widget on the Dashboard. The default value is Checked.

### Display

This dropdown enables you to choose when to display the widget. The default setting is "Always". The available options are:



- Always
- Desktop Only
- Mobile Only

### Title

This text box enables you to specify the title that appears on the widget's Title Bar.

### Refresh (Seconds)

This setting enables you to specify the number of second that can elapse before the widget refreshes automatically. The default setting is "0", which means the widget won't refresh.

### CSS Class

If you are using a custom style sheet, you can specify a style name from that style sheet with this setting.

### Name

An object name that you can use to refer to the widget programmatically, or from other elements in the UI.

### Choose Form

An object picker that enables you to select the Form definition for which you wish to display the most recent form instance.

### Form Instance

A dropdown that enables you to select either the latest Form instance of the selected form, or the Form instance that initiated the current Case instance, when used in a Case dashboard.

## Report #

This widget will display a selected Process Director Report.

The screenshot shows a 'Widget Properties' dialog box with the following fields and controls:

- Show title bar**:
- Display**:  (dropdown menu)
- Title**:
- Refresh (Seconds)**:
- CSS Class**:
- Name**:
- Choose Report**:

Buttons at the bottom:

### Show Title Bar

Unchecking this check box will hide the title bar for the widget on the Dashboard. The default value is Checked.

## Display

This dropdown enables you to choose when to display the widget. The default setting is "Always". The available options are:

- Always
- Desktop Only
- Mobile Only

## Title

This text box enables you to specify the title that appears on the widget's Title Bar.

## Refresh (Seconds)

This setting enables you to specify the number of second that can elapse before the widget refreshes automatically. The default setting is "0", which means the widget won't refresh.

## CSS Class

If you are using a custom style sheet, you can specify a style name from that style sheet with this setting.

## Name

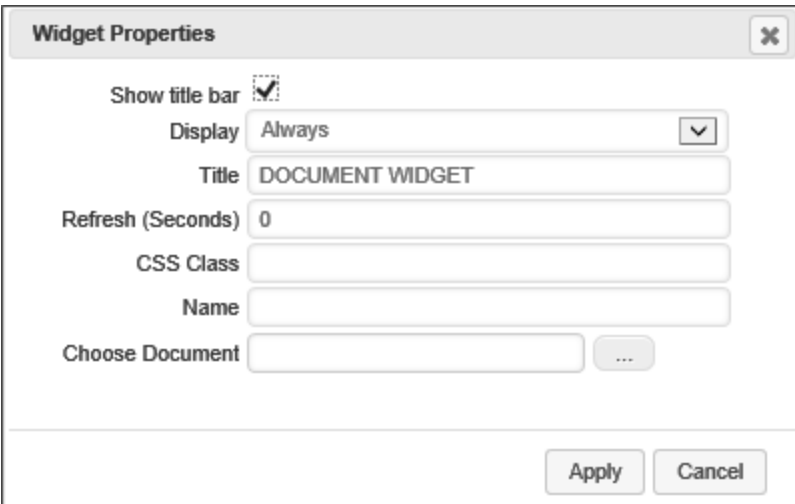
An object name that you can use to refer to the widget programmatically, or from other elements in the UI.

## Choose Report

An object picker that enables you to select the Report object that you wish to display.

## Document #

This widget enables you to display a document from the [Content List](#).



The screenshot shows a dialog box titled "Widget Properties" with a close button (X) in the top right corner. The dialog contains several settings:

- Show title bar**: A checked checkbox.
- Display**: A dropdown menu set to "Always".
- Title**: A text input field containing "DOCUMENT WIDGET".
- Refresh (Seconds)**: A text input field containing "0".
- CSS Class**: An empty text input field.
- Name**: An empty text input field.
- Choose Document**: A text input field followed by a button with three dots "...".

At the bottom of the dialog are two buttons: "Apply" and "Cancel".

## Show Title Bar

Unchecking this check box will hide the title bar for the widget on the Dashboard. The default value is Checked.

### Display

This dropdown enables you to choose when to display the widget. The default setting is "Always". The available options are:

- Always
- Desktop Only
- Mobile Only

### Title

This text box enables you to specify the title that appears on the widget's Title Bar.

### Refresh (Seconds)

This setting enables you to specify the number of second that can elapse before the widget refreshes automatically. The default setting is "0", which means the widget won't refresh.

### CSS Class

If you are using a custom style sheet, you can specify a style name from that style sheet with this setting.

### Name

An object name that you can use to refer to the widget programmatically, or from other elements in the UI.

### Choose Document

An object picker that enables you to select the Document object that you wish to display.

## Image #

This widget enables you to display an image from a URL, or that is contained in the [Content List](#).

The screenshot shows a dialog box titled "Widget Properties" with a close button in the top right corner. The dialog contains the following fields and controls:

- Image alt text:** A text input field with a vertical cursor.
- Display:** A dropdown menu currently showing "Always".
- CSS Class:** A text input field.
- Name:** A text input field.
- Image source:** A dropdown menu currently showing "[Select]".

At the bottom of the dialog, there are three buttons: "Apply", "Cancel", and "Delete".

### Image Alt Text

The text that should be placed in the ALT attribute of the HTML IMG tag that will be used to display the image.

## Display

This dropdown enables you to choose when to display the widget. The default setting is "Always". The available options are:

- Always
- Desktop Only
- Mobile Only

## CSS Class

If you are using a custom style sheet, you can specify a style name from that style sheet with this setting.

## Name

An object name that you can use to refer to the widget programmatically, or from other elements in the UI.

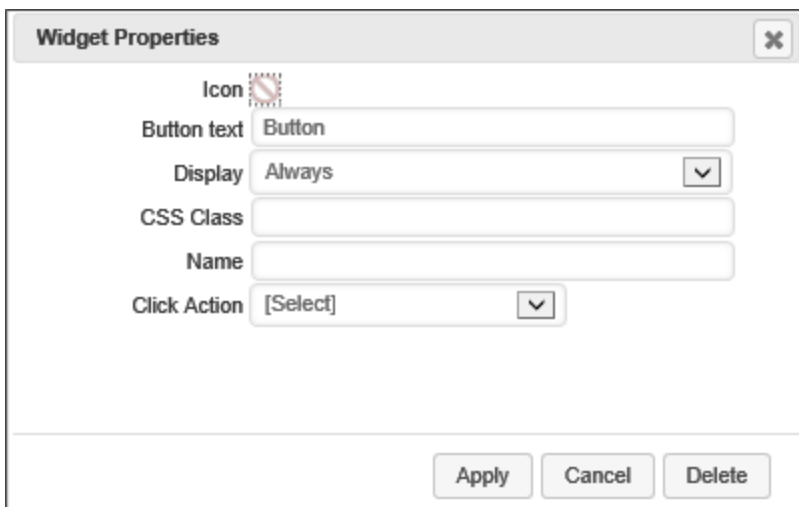
## Image Source

This dropdown enables you to set the source of the image to display. The following options are available:

- **URL:** If this option is selected, Process Director will display a [URL](#) text box, into which you can input the fully-qualified URL of the image.
- **Document:** If this option is selected Process Director will display a [Choose Document](#) object picker from which you can choose an image stored in the [Content List](#).

## Button #

This Widget will display a configurable button on the dashboard.



The screenshot shows a dialog box titled "Widget Properties" with a close button (X) in the top right corner. The dialog contains several configuration fields:

- Icon:** A small square icon with a red 'X' over it, indicating a missing or broken image.
- Button text:** A text input field containing the word "Button".
- Display:** A dropdown menu currently set to "Always".
- CSS Class:** An empty text input field.
- Name:** An empty text input field.
- Click Action:** A dropdown menu currently set to "[Select]".

At the bottom of the dialog, there are three buttons: "Apply", "Cancel", and "Delete".

## Icon

This icon chooser, when clicked, will display the [Icon Chooser](#) window to enable you to select the icon that should appear on the button.

### Button text

This property specifies the text that will appear on the button.

### Display

This dropdown enables you to choose when to display the widget. The default setting is "Always". The available options are:

- Always
- Desktop Only
- Mobile Only

### CSS Class

If you are using a custom style sheet, you can specify a style name from that style sheet with this setting.

### Name

An object name that you can use to refer to the widget programmatically, or from other elements in the UI.

### Click Action

This dropdown enables you to select the action that Process Director will take when the button is clicked by a user. You can select a URL or [Content List](#) object to open. When you select an object, the appropriate controls will appear to enable you to specify the object you wish to display, e.g., a content picker for a Content List object, or a text box for a URL. Alternatively, you can set the button to log off the user, or to close the case workspace, if the dashboard is being used in the context of a case.

### Open in Popup

If you select an object to open, it will, by default, be displayed in an overlay panel to the dashboard when the button is clicked. Checking this checkbox, however, will make Process Director open the object in a separate popup window.

### Clock #

This widget will display an analog clock, with a sweep second hand, and configurable text that displays in the bottom portion of the clock face.

### Display

This dropdown enables you to choose when to display the widget. The default setting is "Always". The available options are:

- Always
- Desktop Only
- Mobile Only

### Title

This text box enables you to specify the title that appears on the widget's Title Bar.

### Date/Time text format

This setting enables you to enter a format string to specify the date or time information you wish to display in the clock face's text area. There are a large number of format strings that can be used, as shown below. The default format string is "llll".


### Localized Time/Date Formats

Format Type	Format String	Output
Time	LT	8:30 PM
Time with seconds	LTS	8:30:25 PM
Month numeral, day of month, year	L	09/04/1986
	l	9/4/1986
Month name, day of month, year	LL	September 4 1986
	ll	Sep 4 1986
Month name, day of month, year, time	LLL	September 4 1986 8:30 PM
	lll	Sep 4 1986 8:30 PM

Format Type	Format String	Output
Month name, day of month, day of week, year, time	LLLL llll	Thursday, September 4 1986 8:30 PM Thu, Sep 4 1986 8:30 PM

### Advanced Formatting

Format Type	Format String	Output
Month	M	1 2 ... 11 12 1
	Mo	st 2nd ... 11th 12th
	MM	01 02 ... 11 12
	MMM	Jan Feb ... Nov Dec
	MMMM	January February ... November December
Quarter	Q	1 2 3 4
	Qo	1st 2nd 3rd 4th
Day of Month	D	1 2 ... 30 31
	Do	1st 2nd ... 30th 31st
	DD	01 02 ... 30 31
Day of Year	DDD	1 2 ... 364 365
	DDD <sub>o</sub>	1st 2nd ... 364th 365th
	DDDD	001 002 ... 364 365
Day of Week	d	0 1 ... 5 6
	do	0th 1st ... 5th 6th
	dd	Su Mo ... Fr Sa
	ddd	Sun Mon ... Fri Sat
	dddd	Sunday Monday ... Friday Saturday
Day of Week (Locale)	e	0 1 ... 5 6
Day of Week (ISO)	E	1 2 ... 6 7
Week of Year	w	1 2 ... 52 53
	wo	1st 2nd ... 52nd 53rd
	ww	01 02 ... 52 53
Week of Year (ISO)	W	1 2 ... 52 53
	Wo	1st 2nd ... 52nd 53rd
	WW	

Format Type	Format String	Output
		01 02 ... 52 53
Year	YY	70 71 ... 29 30
	YYYY	1970 1971 ... 2029 2030
	Y	1970 1971 ... 9999 +10000 +10001
<div style="border: 1px solid #0056b3; border-radius: 15px; padding: 10px; background-color: #e6f2ff;">  This formatting complies with the ISO 8601 standard for dates past the year 9999         </div>		
Week Year	gg	70 71 ... 29 30
	gggg	1970 1971 ... 2029 2030
Week Year (ISO)	gg	70 71 ... 29 30
	gggg	1970 1971 ... 2029 2030
AM/PM	A	AM PM
	a	am pm
Hour	H	0 1 ... 22 23
	HH	00 01 ... 22 23
	h	1 2 ... 11 12
	hh	01 02 ... 11
	k	12 1 2 ... 23 24
	kk	01 02 ... 23 24
Minute	m	0 1 ... 58 59
	mm	00 01 ... 58 59
Second	s	0 1 ... 58 59
	ss	00 01 ... 58 59
Fractional Second	S	0 1 ... 8 9
	SS	00 01 ... 98 99
	SSS	000 001 ... 998 999
Time zone	z or zz	EST CST ... MST PST
	Z	-07:00 -06:00 ... +06:00 +07:00
	ZZ	-0700 -0600 ... +0600 +0700
Unix Timestamp	X	1360013296
Unix Millisecond Timestamp	x	1360013296123



## CSS Class

If you are using a custom style sheet, you can specify a style name from that style sheet with this setting.

## Name

An object name that you can use to refer to the widget programmatically, or from other elements in the UI.

## Chose timezone

This dropdown list contains the timezone settings for a large number of world locations, as well as User local time, UTC, and Zulu time. The default setting for this property is User Local time.

## Activity Journal #

For Process Director v5.0 and higher, the Activity journal enables you to display Milestones and threaded discussions in a Dashboard.

The Journal widget has the following configuration options available:

### Title

The title you desire to display in the title bar of the dashboard widget.

### Scope

The **Scope** property enables you to determine at what level to capture events. By default, the scope is limited to just Activity log events that are specified in the Form definition. You can change the scope to the following settings:

SCOPE	DESCRIPTION
Limit to Case	Only show events from objects that are part of the current Case.
No Limit	Show any events from any object. This setting uses the widest possible scope, and will show any matching events.

## Display

This dropdown enables you to choose when to display the widget. The default setting is "Always". The available options are:

- Always
- Desktop Only
- Mobile Only

## Display

This dropdown enables you to choose when to display the widget. The default setting is "Always". The available options are:

- Always
- Desktop Only
- Mobile Only

## Allow User Comments

Checking this property will enable users to participate in threaded discussions in the Journal widget.

## Milestone Groups

The **Milestone Groups** property enables you to optionally display selected Milestones, in addition to, or in lieu of, user comments. To select the Milestone Groups s you'd like to display, click the **Choose Milestone Groups** button to open the Milestone chooser dialog, and select the desired Milestone Groups to display.

## CSS Class

If you are using a custom style sheet, you can specify a style name from that style sheet with this setting.

## Datasource Objects

The Datasource object enables you to create a data source that can connect to an external database. The Datasource object can then be used in multiple areas of Process Director. An example use of the Data-source object is with a Database Connector Custom Task. With a Database Connector Custom Task you must specify a data source either manually (by entering a connection string) or by using a Datasource object which was created once and used throughout Process Director.

Organize Datasources in root folders of the partition, separate from any implementation folders, so that implementers don't have to export Datasources when the implementation is exported from development to production. Exports and Imports are name-based, and relative folder position-based. Consequently, the root folder containing database sources should be mirrored on development and production servers, so

that both the names and relative folder positions are exactly the same in both environments. If the mirroring is correct, implementations will be imported/exported between development and production without losing contact with the Datasources.

## Datasource Object Configuration #

The screenshot shows the 'Datasource Object Configuration' dialog box. At the top, there is a title bar with a gear icon on the left, a lock icon in the center, and an 'OK' button on the right. Below the title bar, the main content area is titled 'PROPERTIES'. The first section is 'Datasource Connection Name', which has a text input field containing 'TrainingDatasource' and an 'Icon' button with a small green icon. The second section is 'Description', which has a text area with the placeholder text 'Enter a brief description of this Object'. The third section is 'Datasource Type', which has a dropdown menu currently set to 'Internal Database'. Below this, there is an 'Enable Cache' checkbox (which is unchecked), a 'Clear Cache' button, and the text 'Current Cache Entries: 0' and 'Current Cache Hits: 0'. The fourth section is 'Cache Timeout in seconds', which has a text input field and the following text: 'Regardless of the setting, the cache will be cleared automatically by the system any time it determines it is appropriate to do so. However, if a number of seconds is indicated, the cache will never be older than the specified interval.' At the bottom of the dialog, there is a 'Select Tables' button with a blue downward arrow icon.

There are a number of database types that are supported by the Datasource object, which are listed in the [Datasource Type](#) dropdown. In addition to supporting the internal Process Director database, the Datasource object also supports a wide variety of other types of data sources :

- SQL Server
- Oracle
- Access
- Teradata
- PostgreSQL Npgsql
- PostgreSQL Devart
- MySQL
- SharePoint
- Twitter
- Google Sheets
- Amazon Database
- Windows File Path

- Dropbox
- OneDrive
- Facebook
- Salesforce.Com
- Box (Box.net)
- MS Dynamics CRM
- Azure IoT Hub
- SFTP (SSH File Transfer)
- Compliance Data Source
- OAuth

## Other Datasource Types

To see more information about different [Datasource Types](#) and their configuration, please refer to the following topics:

- [Common Datasources](#)
- [Excel Datasources](#)
- [File Datasources](#)
- [SharePoint Data Sources](#)
- [Social Datasources](#)

## Datasource Caching

Starting with Process Director v3.76, database caching for Datasource objects is implemented on all SQL Datasource objects, including Datasources for SQL Server, Access Oracle, and PostgreSQL. When caching is enabled on a Datasource, Process Director will retrieve the data from the database one time, then store the returned data in memory until the cache expires. Any time the database call is subsequently made, Process Director will first check the cache to see if the cache is still active, and, if so, will return the data from the cache, rather than attempting to retrieve the data from the database. If the cache is expired, Process Director will perform the database call, and reset the cache timeout.

When you select one of these data types from the [Datasource Type](#) dropdown, the Caching parameters will appear on the configuration form.

Datasource Connection Name Icon

TrainingDatasource

## PROPERTIES

Description

*Enter a brief description of this Object*

Datasource Type

SQL Server ▼ **SQL-type Datasource**

Provider SQL Oracle ODBC DB2 Access

System.Data.SqlClient

Connection

Enable Cache Clear Cache Current Cache Entries: 0 Current Cache Hits: 0 **Caching Options**

Cache Timeout in seconds. Regardless of the setting, the cache will be cleared automatically by the system if a number of seconds is indicated, the cache will never be older than the specified interval.

Select Tables

Test Connection

You can enable caching by checking the **Enable Cache** check box.

You can manually clear the cache, if necessary, by clicking the **Clear Cache** button. The **Current Cache Entries** label always displays the number of items currently in the cache, and clearing the cache should reset the value to "0".

You can optionally set the **Cache Timeout (seconds)** to the number of seconds that you wish the cache to remain active. If you don't set the **Cache Timeout**, Process Director will automatically time the cache out in around eight hours. The **Cache Timeout** that you should configure for the Datasource will, of course, depend on how current you need the data to be.

When caching is configured, all Business Values and database Custom Tasks that use the Datasource will also use its caching configuration, though you'll need to ensure that you are using the latest version of

the Custom Tasks. Custom Tasks that were created prior to February, 2016, are *not* configured to use Datasource caching.

Users of Datasource caching should notice significant performance increases for data-heavy Forms.

**i** As a security enhancement to Process Director v5.08 and higher, when exporting a Datasource to XML from the Content List, the Datasource Password will *not* be exported. You'll need to re-enter the password when the Datasource is imported again. As a best practice, *Datasources shouldn't be exported between systems*, anyway. They virtually always require reconfiguration on the target machine, and it's easy to wipe out a working Datasource pointing at production data and replace it with another pointing at test data that you imported from your test system. Even worse, when you go back and refresh your test systems from production, the same thing can happen in reverse, causing you to corrupt your production data with test data.

**i** As a security enhancement to Process Director v6.0 and higher, passwords in connection strings and password fields are obfuscated by default, and will not display unless the Eye icon adjacent to the field is clicked, which will show the password in clear text.

## Common Datasource Types

There are several datasource types you'll commonly use for connections to external data.

### SQL-Based Data Sources #

SQL Server, Oracle, Postgress SQL, Access, etc, are all server-based database system that use connection strings to access the database server. In most cases, you'll need to configure the following fields to create the database connection to the server:


Field	Description
Provider	The database provider string
Connection	The database connection string

Next to the **Provider** property, a series of buttons will, when clicked, automatically set the **Datasource Type** and **Provider** properties, and supply a sample connection string in the **Connection** property. You can then edit this sample connection to the correct settings for the database server.

The screenshot shows a configuration form for a Datasource. At the top, there is a dropdown menu labeled "Datasource Type" with "SQL Server" selected. Below this is a "Provider" section with several buttons: "SQL", "Azure SQL", "Oracle", "ODBC", "Access", and "MySQL". The "SQL" button is currently selected. Below the buttons, there are two text input fields. The first field is labeled "System.Data.SqlClient" and is the "Provider" field. The second field is labeled "Connection" and contains a masked connection string (represented by dots). To the right of the "Connection" field, there is an "Eye" icon. A red callout box with a white background and a red border points to this icon, containing the text: "Eye" icon to unmask the Connection field's value.

For Process Director v6.0 and higher, the **Connection** property is masked, by default, to obfuscate the values displayed in the connection string, such as the server name, password, etc. The property can be unmasked by clicking the "Eye" icon adjacent to the field.

While this is the quickest way of setting up the database connection, you can also manually configure the **Database Type**, **Provider**, and **Connection** string. For more information on database connection strings please refer to [connectionstrings.com](http://connectionstrings.com).

 Due to a bug in the newer MySQL .NET drivers, the MySQL 6.9.12 or older driver must be used. This is a bug introduced in the MySQL .NET driver version 6.10 and higher ([MySQL Bug #89159](#)).

### Compliance Data Source #

The Compliance Datasource is available to users of Process Director v5.15 and above. This type isn't a database connector, but, instead, provides a secure storage for **User ID** (username) and **Password** attributes for use elsewhere. These stored attributes can be accessed via the SDK API in custom scripts to provide the login information needed to access some external data source. Additionally, the attributes can be accessed in Custom Tasks that require user names and passwords, such as Fill Fields from REST, as well as Business Values, to provide required login information, without providing it in the configuration of the Custom Task or Business Value.

**Datasource Type**  
Compliance Data Source ▼
This datasource allows credentials to be securely stored and used by APIs, Custom Tasks, Business Values, etc.


**User ID**

**Password**

For certain connections and authentication schemes (e.g. REST with HTTP Headers), the connection requires additional data in the form of keys (names) and values.

Add Value

Name	Value
<input style="width: 95%;" type="text"/>	String may contain SysVars (e.g. "CURR_DATE"). <span style="float: right; color: red; font-weight: bold;">✘</span>

 Test as built-in User

This Datasource enables you to store the attributes securely, separate from the Custom Task or Business Value, to help prevent a compromise of the login data, while still enabling designers to incorporate secure logins in Custom Tasks or Business Values, without having to know the actual login information.

The attributes are *not* accessible via system variable, for security reasons.

Users of Process Director v5.42 and higher can also add Name/Value pairs to the Datasource, for connections that require additional data in the form of keys (names) and values.

A **Test as Built-In User** button enables you to provide the credentials of a built-in Process Director user account, then test it to ensure the account works properly.

## OAuth Data Source #

The OAuth Data Source provides a generic OAuth connector for data sources that use this authentication method.

The screenshot shows the configuration interface for an OAuth Data Source. At the top, there is a 'Datasource Connection Name' field containing 'OAuth Connector' and an 'Icon' field with a stack of coins icon. Below this is a 'PROPERTIES' section. The 'Description' field contains the placeholder text 'Enter a brief description of this Object'. The 'Datasource Type' is set to 'OAuth' with a dropdown arrow, and there is a checkbox labeled 'Use Refresh Token to generate Access Tokens on demand'. Below these are three text input fields for 'Client Token', 'Client Secret', and 'Access Token'. At the bottom, there is a section for additional data with an 'Add Value' button and a table with columns 'Name' and 'Value'. The 'Value' field contains the text 'String may contain SysVars (e.g. "CURR\_DATE").' and a red 'X' icon.

When using the OAuth Datasource, you must provide the **Client Token**, **Token Secret**, and **Access Token** for the OAuth authentication. Some OAuth systems enable you to dynamically generate access tokens for each connection, so a property named **Use Refresh Token to generate Access Tokens on demand** will, when checked, replace the Access Token property with a **Refresh Token** property, which enables the system to pass the refresh token to the OAuth system to dynamically generate a new access token for each connection instance. The new tokens will be generated with an expiration date/time from the OAuth system, and will remain valid only until the specified expiration, after which, a new token will be automatically requested.




Users of Process Director v5.42 and higher can also add Name/Value pairs to the Datasource, for connections that require additional data in the form of keys (names) and values.


## Internal Datasources #

Process Director provides connection to two different internal [Datasource Types](#).

 For Cloud customers, access to these internal Datasources may be restricted, requiring BP Logix to create the Datasources for you.


The **Internal Database Datasource Type** will connect directly to the Process Director database. No other Properties are necessary when using the Internal Database type.

 BP Logix does NOT recommend the use of the Internal Database Datasource for anyone but highly experienced SQL database specialists. Its use risks altering the core Process Director database, and if you use it, you do so ENTIRELY at your own risk. Be warned!

 When a user attempts to create an Internal Database Datasource, the Datasource Type will default to "Other" if the user doesn't have permission to create an Internal Database connection.

The **Internal User Database Datasource Datasource Type** grants access to a separate database that's included with Process Director, whose purpose is to store data in SQL format from Excel file imports, or other custom data you might need to use. It contains a few SQL Views that are exposed from the Process Director Database for administration or system management purposes, but is otherwise empty. This [Database Type](#) is designed for your use, to store any data you might desire.

## oData Datasources #

 CData, the provider of oData connections, has unique license terms for using these connection types. They are, therefore, licensed separately. Customers using Datasources for Salesforce or MS Dynamics should contact customer support to enable these features in their license.

Some datasources that connect to external applications such as Salesforce, MS Dynamics, etc., are provided via the oData database connector.

<b>Datasource Type</b> Salesforce.com ▼	This datasource connector requires that you have licensed the CData.com integration drivers.
<b>User ID</b> <input type="text"/>	
<b>Password</b> <input type="text"/>	
<b>Security Token</b> <input type="text"/>	

<b>Datasource Type</b> MS Dynamics CRM ▼	This datasource connector requires that you have licensed the CData.com integration drivers.
<b>Complete URL to the MS Dynamics CRM web application</b> <input type="text"/>	
<b>User ID</b> <input type="text"/>	
<b>Password</b> <input type="text"/>	

The properties available for these Datasources may vary slightly, as the two examples above show. For specific information about implementing the oData license for Salesforce, please refer to the [Salesforce Licensing and Datasources topic](#).

## Other Datasource Types

To see more information about different [Datasource Types](#) and their configuration, please refer to the following topics:

- [Excel Datasources](#)
- [File Datasources](#)
- [SharePoint Data Sources](#)
- [Social Datasources](#)

## Salesforce Licensing and Datasources

Customers who wish to use data connections to Salesforce, using the appropriate Datasource object, must first obtain an appropriate Open Data (oData) connector to Salesforce from CData. Salesforce oData

licenses can be purchased from [CData's web site](#), using the **Server Licensing** tab.

# Pricing Options

The Salesforce ADO.NET Provider may be licensed through one of our multi-source ADO.NET Provider subscriptions. Questions? We're happy to help! [Call](#) our office, or email us as at [sales@cdata.com](mailto:sales@cdata.com).

[Server Licensing](#) | [Developer Licensing](#) | [Enterprise / Team](#)

## Server License

Salesforce ADO.NET Provider  
Single Server License

# QUOTE

For use with commercial Web Apps, BI, Analytics, ETL, and server tools, like SQL Server Analysis & Reporting Services. [i](#)

[About Server Licensing](#)

[Request Quote](#)

Once the license has been purchased, you will receive the appropriate installation and licensing files. In addition, your **Account Settings** section of the CData web site will give you access to a Security Token for account, as well as the ability to create or reset Security Tokens.

The screenshot shows the Salesforce user interface. At the top, there is a navigation bar with 'Sales' and various menu items like 'Home', 'Leads', 'Accounts', etc. Below this is a search bar and a 'Quick Find' box. The main content area is titled 'Reset My Security Token' and contains a warning message: 'When you access Salesforce from an IP address that isn't trusted for your company, and you use a desktop client or the API, you need a security token to log in. What's a security token? It's a case-sensitive alphanumeric code that's tied to your password. Whenever your password is reset, your security token is also reset.' Below the message is a yellow box with a lock icon and the text 'After you reset your token, you can't use your old token in API applications and desktop clients.' At the bottom of this section is a 'Reset Security Token' button. On the left side, there is a sidebar menu with 'My Personal Information' expanded, showing options like 'Advanced User Details', 'Authentication Settings for External Systems', 'Change My Password', 'Connections', 'External Credentials', 'Grant Account Login Access', 'Language & Time Zone', 'Login History', 'Personal Information', and 'Reset My Security Token' (which is currently selected).

You **must** have a valid Security Token from CData in order for your Salesforce Data Source object to connect with your data. In addition you **must** also request API Access from Salesforce, separately, since API Access is **disabled by default** in the Professional Edition of Salesforce. Please refer to the [SalesForce web site](#) for more information about requesting API Access.

## Installing the Salesforce License

Depending on whether you're a Cloud or On-Premise customer, the installation process will differ.

### Cloud Customer Installation Process

Cloud customers must provide BP Logix with all of the installation and licensing files received from CData. BP Logix will install the license(s) on the appropriate cloud installation(s) for you.

### On-Premise Installation Process

On-Premise customers will have to install the connector on their own Process Director servers, using the process below.

1. Run the installation file on your Process Director server.
2. Follow the steps provided by CData to install your license. If given options on which binary to use, use the version in C:\Program Files\CData\CData ADO.NET Provider for Salesforce 2023\lib\netstandard2.0.
3. Copy all files **except** the **install-license.dll** file from C:\Program Files\CData\CData ADO.NET Provider for Salesforce 2023\lib\netstandard2.0 to the bin folder of your Process Director website folder, which for a default installation, will be C:\Program Files\BP Logix\Process Director-5.44.600\website\bin.
4. Open the **web.config** file of your Process Director installation for editing.
5. Locate the following line, which is located under configuration\system.data\DbProviderFactories:

```
<add name="CData ADO.NET Provider for Salesforce" invariant="System.Data.CData.Salesforce" description="CData ADO.NET Provider for Salesforce" type="System.Data.CData.Salesforce.SalesforceProviderFactory, System.Data.CData.Salesforce, Version=15.0.0.40, Culture=neutral, PublicKeyToken=cdc168f89cffe9cf" />
```

6. Change this line to read as follows:

```
<add name="CData ADO.NET Provider for Salesforce 2023 (.NET Standard 2.0)" invariant="System.Data.CData.Salesforce" description="CData ADO.NET Provider for Salesforce 2023 (.NET Standard 2.0)" type="System.Data.CData.Salesforce.SalesforceProviderFactory, System.Data.CData.Salesforce" />
```

7. Save and close the **web.config** file.
8. Restart the Process Director App Pool in Internet Information Server (IIS).

## Creating the Data Source Object

To create a Salesforce Data Source object, navigate to the folder location where you'd like the Data Source object to reside. BP Logix recommends that Data Sources be created in a [\[Data Sources\]](#) folder at the root of your installation partition.

1. From the **Create New** menu, select **Data Source** to open the **Create Data Source** screen.
2. In the **Create Data Source** screen, provide a **Data Source Name**, e.g., "Salesforce Connector", then click the **OK** button to create the Data Source.

3. Set the **Datasource Type** property to *SalesForce.com*.
4. Set the **UserID** property to the account User ID that's associated with the Security Token you obtained from your SalesForce account, and the **Password** property to the password used by the account.
5. Enter the valid SalesForce Security Token into the **Security Token** property.
6. Click the **Update** menu button to save your settings.
7. Click the **Test Connection** button to test the connection to your SalesForce data. Assuming you have both a) correctly configured the properties of the Data Source and b) been granted API access from SalesForce, and specified above, the connection test should display a success message. You can now close the Data Source object.

If you have not requested API Access from SalesForce, an error message will be displayed.

Description

Enter a brief description of this Object

[500] Could not execute the specified command: API\_DISABLED\_FOR\_ORG: API is not enabled for this Organization or Partner

Datasource Type

Salesforce.com

This datasource connector requires that you have license

You'll need to be granted API Access before continuing, in order to successfully test and use your Salesforce Data Source.

## Other Datasources

To see more information about different [Datasource Types](#) and their configuration, please refer to the following topics:

- [Excel Datasources](#)
- [File Datasources](#)
- [SharePoint Data Sources](#)
- [Social Datasources](#)

## Microsoft Excel Datasources

Process Director can use a Microsoft Excel spreadsheet as a data source, enabling you to check out and edit the spreadsheet whenever needed. Process Director can re-import the data every time you check the spreadsheet back in, to ensure the data always reflects the current values in the spreadsheet.

## Creating the Structure

To create the structure of an internal database table, one must use MS Excel, following these steps:

The names of each of the cells in the top row of the Excel sheet will become the names for the database fields.

	A	B	C
1	<b>Username</b>	<b>Interviewer</b>	<b>Salary</b>
2	Robert	Sally	\$ 12,000.00
3	Richard	Sarah	\$ 9,001.00
4	Ronald	Samantha	\$ 3,383.00

Each row below the top is a place for data to be entered. Enter the proper information under the proper columns.

	A	B	C
1	<b>Username</b>	<b>Interviewer</b>	<b>Salary</b>
2	Robert	Sally	\$ 12,000.00
3	Richard	Sarah	\$ 9,001.00
4	Ronald	Samantha	\$ 3,383.00

Each sheet acts as a different table in the database. Each table can have different field names and different data.

Process Director won't transform the Excel sheet under the following conditions:

Process Director will ignore default sheet names, like "Sheet1", "Sheet2", or any sheet whose name matches the Sheet# pattern, and you can't use spaces in sheet names. Additionally, sheet names that start with the "#" character will also be ignored. Process Director won't transform any sheets whose A1 cell is empty. You need to rename the default sheet names, therefore, with meaningful names. The sheet names you provide will be used to create and name database tables, so you should ensure that sheet names, like field names, don't use spaces or special characters, in order to make them compliant with database naming conventions.

	A	B	C	D
1	<b>Job Name</b>	<b>Employee</b>		
2	VP of Sales	Jim		
3	Lead Mac Dev	Avi		
4	Janitor	Scott		

Sheet1 Sheet2

### Creating a Data Source

To import the data from the Excel spreadsheet, you must create a Datasource that connects to the database you wish to use to store the imported Excel data. You can use any accessible database, including the BP Logix Internal User database, to store the Excel data. See the [Datasource Objects](#) topic for information about how to configure a Datasource.

### Importing Database from Excel File

Once the Datasource is created, you need to import data into it from the Excel file you made. Go to the [Content List](#). Under the **Create New** dropdown menu, select **Document/File** to open the [Upload Document](#) page.

**Upload Document**

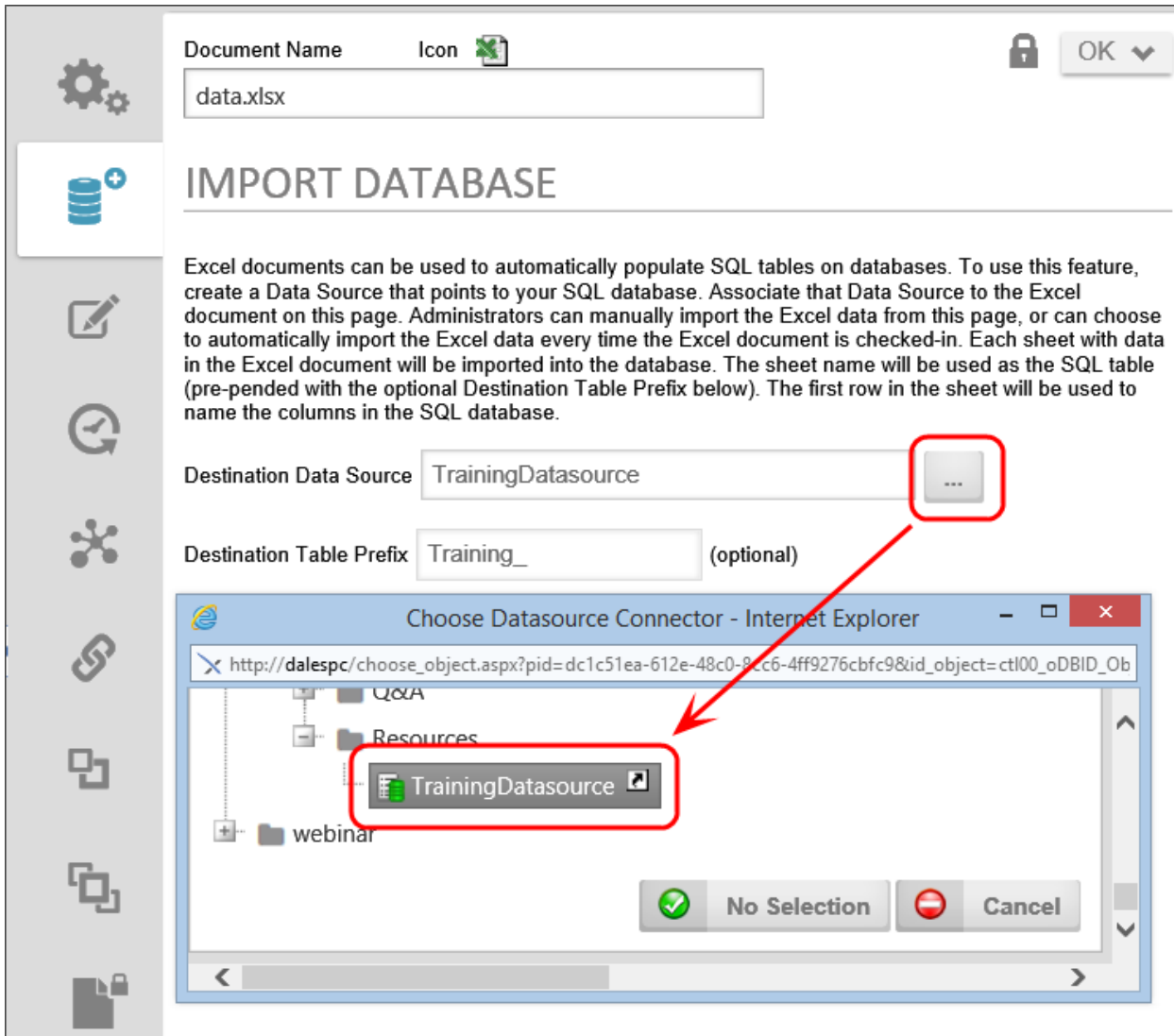
Browse

Use Detailed Upload

✓ Upload
✗ Cancel

On the **Upload Document** page, click the **Browse** button and locate the Excel file in your local file system. Double click the file to save and close the browse dialog box, then click the **Upload** button to upload the file to your partition.

The final step is to take the data from the uploaded Excel file and import it into the Datasource. After the Excel file is uploaded, you'll be presented with the object definition screen for the Excel file. Click on the **Import Database** tab. Click the **Destination Data Source** property's **Build** button to select the Datasource you created earlier.




After selecting the desired Datasource, a series of properties provide additional settings you can configure to govern how the data will be imported.

Option	Explanation
Destination Table Prefix	This is the text string that will be prepended to your existing Excel sheet names in the database. By default, the value of this property is "Excel_". So if



Option	Explanation
	your sheet name was “February_Users” in Excel, when you looked for it in the database, it would be named “Excel_February_Users”
Drop database tables before import	To “DROP” a table is to delete it completely. The difference between “dropping” a table and the “Clear database rows” checkbox is that if you only clear the rows, the format of the information in existing fields (columns) is kept. In other words, in Excel, if you had a date field called “Married” and imported it to the database with date information, Process Director expects date formatted information to be entered in that field. If you opened Excel and altered the “Married” column to be a Boolean (True/False) value, and did not drop the tables when you imported, you'd generate an error.
Create database tables if needed	If you have a previously imported Excel spreadsheet and add a new sheet to it in Excel, Process Director won't import the new sheet as a new table unless this checkbox is checked.
Clear database rows in tables before inserting	This is the difference between recreating a table's information, and appending the information in Excel to the existing table in Process Director. Appending of tables is usually only done by database administrators when building or editing informational databases. If you want the table in PD to look just like what you saw in Excel, select this checkbox.
Automatically import this Excel file after every check-in or import.	After you've added an Excel spreadsheet to your Process Director instance, you'll need to “check out and edit” the file in order to open it and make changes. After your changes are made, you select “upload and check in” and this checkbox will automatically recreate or append the information in your PD database without you having to re-import the file.

 **When importing a project from an XML file into the Content List, if the project contains an Excel file to import into the database, the database import will not be performed if this property is unchecked.**

Now you can click the **Import Tables Now** button to import the Excel data into your Datasource.

### Transform Excel Sheet to a Database Table

This enables you to use a Microsoft Excel sheet and transform the content to a table and data within a Datasource specified in Process Director. This is useful when you need to add data to a database without having to use a database management tool. A form that performs these functions can be distributed from BP Logix and is also included in the sample files Process Director.

#### *Creating the Excel File*

It is important to configure your Excel file to these specifications in order for Process Director to transfer the data correctly.

1. Create an Excel file using Microsoft Excel.
2. Rename the Excel sheet to the name of the table you'd like to create. Please don't name your table starting with "tbl". The form will result in error.
3. Create column names by adding the names of the column to the first row and in each cell. If you don't specify a type, it will be determined automatically based on the value of the first record. Process Director will attempt to get the cell/column type from the Excel column, and set the DB column type accordingly. For instance, if the column header cell is formatted as a date/time cell, Process Director will use the Datetime column type.
4. Optionally set the data type of the column by appending ":type" to the column name, where type is the native database type (e.g. varchar) or one of the following keywords: bp\_string, bp\_bool, bp\_decimal, bp\_int or bp\_datetime. The default data type of bp\_string for Process Director v5.29 and higher is NVARCHAR(MAX), and is VARCHAR(512) for all older versions. You can override the string size using bp\_string(size) to set the string size to a different number of characters.

Marketing:int	
	94
	98

If you use any of the "bp\_\*" keywords, the data will be converted into a SQL data type, as follows:

Keyword	SQL Data Type
bp_string	NVarchar(MAX)
bp_bool	Bit
bp_decimal	Decimal(18,4)
bp_int	Int
bp_datetime	Datetime

5. Add the data to the corresponding rows and columns.
6. Save your Excel file.

Your Excel file should represent a database table with columns and rows. Here is a sample of what it can look like:

The screenshot shows an Excel spreadsheet with the following data:

AccountCode	Name	Address	City	State	Zip	AccountType
SMITLTOP	Smith Family Trust	875 Montrose Ave.	Houston	TX	77084	Trust
BARKOPST	Kevin Barksdale	2314 Paseo del Norte	San Diego	CA	92110	Individual Asset
KITMLLLI	Kitchener Trust	497 Gin Ln.	Southampton	NY	11968	Trust
RKIMIIIIG	Richard King	299 Rametto Rd.	Santa Barbara	CA	93108	Individual Asset
ITALSLNV	Ivan Talosian	11143 Pine Cone Cir	Incline Village	NV	89451	Individual Asset
ARKNIHJI	Arkham Institute	9927 Renaissance Pkwy NE	Atlanta	GA	30308	Institutional Assets

### Transfer Excel Data

1. Load the Excel file.
2. Go to the “Import” tab.
3. Select a Datasource to import your data from Excel.
4. Click on the Run Import button to insert the data.

Additional options allow you to:

- Prefix the destination table name.
- Drop (Delete) tables with the same name
- Clear all rows in table before inserting.
- Create table if it doesn't exist.

### Other Datasource Types

To see more information about different [Datasource Types](#) and their configuration, please refer to the following topics:

- [Common Datasources](#)
- [File Datasources](#)
- [SharePoint Data Sources](#)
- [Social Datasources](#)

### Windows File System Datasources

Process Director can connect to your network's file system using any accessible UNC. The primary purpose of this Datasource is to enable the use of some of the [Content Actions Custom Tasks](#), specifically:

- [Export Files to Filesystem](#)
- [Import Files from Filesystem](#)

Each of these Content Actions Custom Tasks require a Windows File System Datasource to be configured prior to use.

The screenshot shows a configuration window for a Datasource. On the left is a vertical sidebar with icons for settings, links, and documents. The main area is titled 'PROPERTIES' and contains the following fields:

- Datasource Connection Name:** A text box containing 'Sample Datasource' with an 'Icon' button to its right.
- Description:** A text box containing 'File data source for shared document folders'.
- Datasource Type:** A dropdown menu with 'Windows File Path' selected.
- File Path:** A text box containing 'M:\Shared\Documents'.
- User ID:** A text box containing 'UserID'.
- Password:** A text box with masked characters (dots).
- Domain:** A text box containing 'DomainName'.
- Test Connection:** A button with a document icon and the text 'Test Connection'.

To set up the Datasource, you need select "Windows File Path" as the **Datasource Type** property. Once you do, new properties will appear to enable you to configure a valid Windows **UserID**, **Password**, and **Domain** name to configure the properties. The **UserID** and **Password** should be set to a user with adequate permissions to read/write files to the desired file paths in the domain.

For security reasons, the **File Path** property **must** correspond to a file path authorized in the [Allowed Export Locations](#) configured in the custom variables file. There are additional file path properties in the configuration screens for file-based Custom Tasks. This Custom Task file path should be a *continuation* of the File Path property specified in the Datasource. For instance, let's say the file Datasource is configured to use the **File Path** "C:\docs". If you want the Custom Task to export a file to "C:\docs\MyProcessDocs",

then the File Path property in the Custom Task would be "\\MyProcessDocs". The two [File Path](#) properties will be concatenated as "C:\docs\MyProcessDocs".

There are some additional considerations you should keep in mind.

- The file path you specify must exist when the Datasource is created. Process Director will **not** create the folder. If the file path doesn't exist, you'll receive the following access error in the format: "Could not access Folder: X:/Folderpath".
- If you specify a destination folder (ex: 'MyFolder') in a Custom Task that uses the Datasource, it will look within the file path specified in your Datasource. So, if you set "C:\MyDSFolder" as the Datasource [File Path](#), the Custom Task will look within "C:\MyDSFolder\MyFolder" to perform the operation.
- If you don't specify user credentials in your Datasource, the system will use the default app pool user that has been configured for IIS at the Process Director web server. Any permissions needed to access any files will need to be granted to this IIS default app pool user (typically called "DefaultAppPool"). In most cases, this default app pool user will probably not have proper permissions to access the file system.
- If user credentials are used in the Datasource configuration, then the site will need to ensure the DefaultAppPool user has the security policy configured to allow impersonation of other users. The app pool user uses impersonation to connect as the user specified in the Datasource's configuration. Previous versions of IIS defaulted to Impersonation **ON** for users, but in IIS 7.5, the default was changed to Impersonation **OFF**.
- Path names may use spaces, e.g., "C:\Users\username\My Documents".

## SFTP (SSH File Transfer) Datasources <#>

In addition to the standard File System Datasource, users of Process Director v5.12 and higher can use Secure FTP to access and transfer files as well. This Datasource type will enable process director to access a remote file store via secure SSH file transfer. In most cases, the file transfer will be done via the [Export Files to Filesystem](#) or [Import Files from Filesystem](#) Custom Tasks. You must have this datasource configured prior to use by the Custom Tasks.

The screenshot shows a configuration window for a Datasource. At the top, there is a 'Datasource Connection Name' field with the value 'test' and an 'Icon' field with a database icon. Below this is a 'PROPERTIES' section. The 'Description' field contains the placeholder text 'Enter a brief description of this Object'. The 'Datasource Type' is set to 'SFTP (SSH File Transfer)'. The 'SFTP Host' field has a placeholder '(optionally specify a Port by appending ":" and the Port number, e.g. "host:22")'. Below that are fields for 'User ID' and 'Password'. The 'SSH Private Key' section includes a text area with the instruction: 'If using a password-protected Key, the Password field will supply the password for this Key'. At the bottom left, there is a 'Test Connection' button with a green checkmark icon.

The following properties must be configured for this type of Datasource.

**DataSource Type:** SFTP (SSH File Transfer)

**SFTP Host:** The hostname of the secure FTP server where the files are located. You may optionally specify a port to use for FTP by appending it to the hostname after a colon, e.g., ":22".

**UserID:** The User ID of an FTP account that is authorized to access the FTP location.

**Password:** The password of the FTP account.

**SSH Private Key:** The hash string of the private key that must be used to access the data securely.

## Other Datasource Types

To see more information about different **DataSource Types** and their configuration, please refer to the following topics:

- [Common Datasources](#)
- [Excel Datasources](#)
- [SharePoint Data Sources](#)
- [Social Datasources](#)

## SharePoint Data Sources

With the implementation of Microsoft's move to [Modern Authentication](#), using the Microsoft identity platform, logging into cloud-based versions of SharePoint is no longer possible by simply using a user name and password. Legacy installations that use older versions of SharePoint may still do so, but SharePoint has largely implemented an OAuth-based authentication scheme, with additional security provided by the use of encryption certificates.

In Process Director v5.44.1000, Modern Authentication for SharePoint was implemented using the [SharePoint OAuth](#) Datasource, which only gives access to SharePoint at the Tenant (organizational) level.

For Process Director v5.44.1103, The [SharePoint OAuth](#) Datasource was renamed to [SharePoint OAuth \(Tenant\)](#), while a new Datasource [SharePoint OAuth \(Site\)](#), was added to give access to SharePoint at the Site level, rather than at the entire tenant.

The existing [SharePoint](#) Datasource, which uses the simple username/password authentication scheme, is still available for customers who are using older versions of SharePoint. This legacy authentication method should be relevant to only a very small minority of customers, and has been renamed to [SharePoint Legacy](#).



This update to the SharePoint Datasources will require updating the SharePoint Custom Tasks!

### Configuring a SharePoint OAuth (Tenant) Datasource #

Modern Authentication provides much more secure access to SharePoint, but does require a more complex setup process. To set up Modern Authentication between SharePoint and Process Director, you must first create and register an Azure Active Directory (AAD) application. The System Administrator's Guide has instructions for creating the AAD application in the [Configuring Azure for Process Director Integration](#) topic.

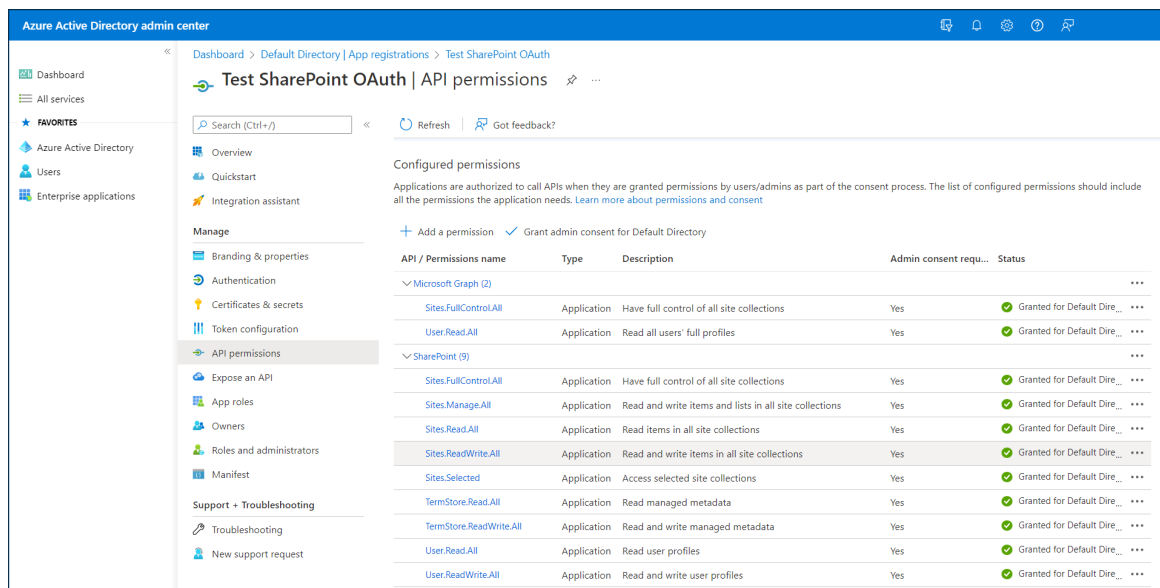
Once you've created the AAD Application, you can begin the process for configuring SharePoint Online.

### Configure SharePoint Online permissions #

To configure the AAD application to use SharePoint with Process Director, you'll need to perform the following configuration steps:

1. If you have access to multiple tenants, use the Directories + subscriptions filter in the top menu to switch to the tenant in which you want to register the application.
2. Search for and select [Azure Active Directory](#).
3. Under [Manage](#), select [App registrations](#), then select your Process Director application. In this example, we'll use "Test SharePoint OAuth" as the AAD Application name, though, of course, the name you use may vary.
4. Click [API permissions](#).

5. Click **Add a permission** and add all permissions displayed below to the **SharePoint** section of the **API Permissions** area:



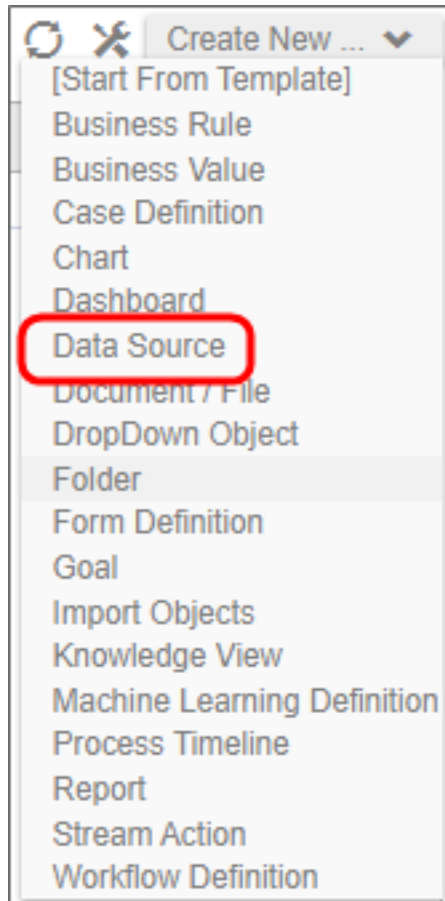
## Create the SharePoint OAuth (Tenant) Datasource #

Now that the application has been fully registered in Azure, and the appropriate SharePoint API permissions have been set, you can create the SharePoint OAuth Datasource in Process Director. Be sure to keep the Azure window open, however, as you'll need to transfer some information from Azure to configure the SharePoint OAuth Datasource. Ensure you've opened the **Azure Active Directory admin center** window to the **Overview** tab of the **App registrations** page of your Process Director integration app. In this example, we'll use the "Test SharePoint OAuth" application we used in the steps above.

### Instructions

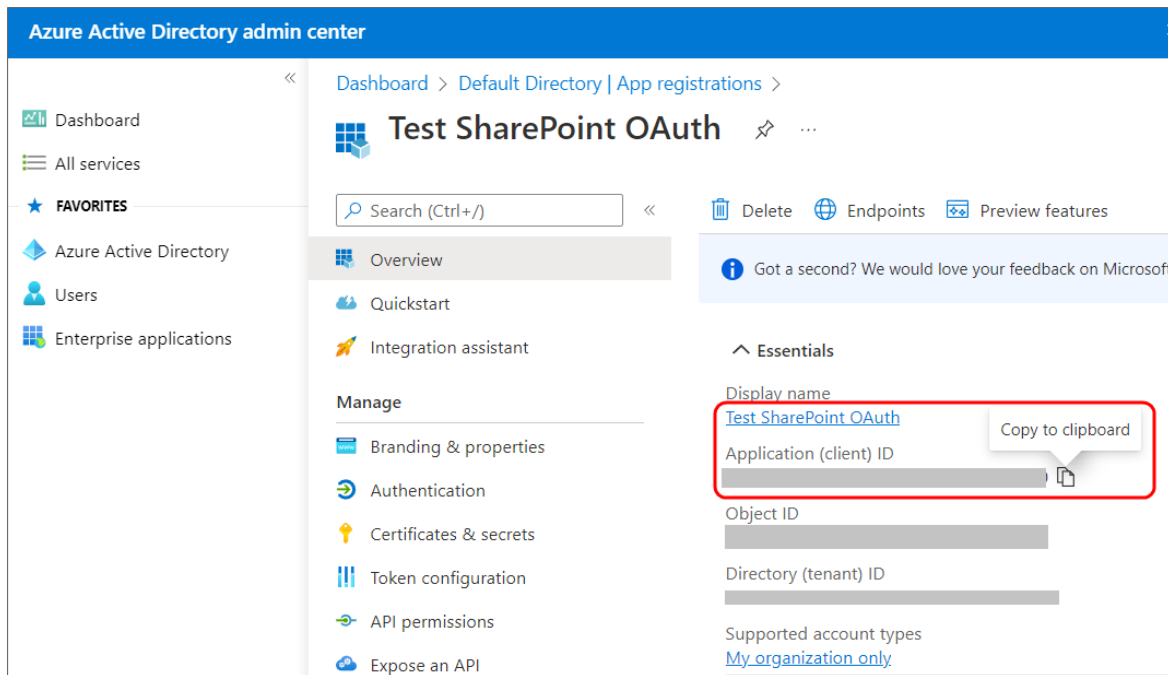
1. Navigate to the Process Director folder in which you want to store the new Datasource, then select **Data Source** from the **Create New** menu.



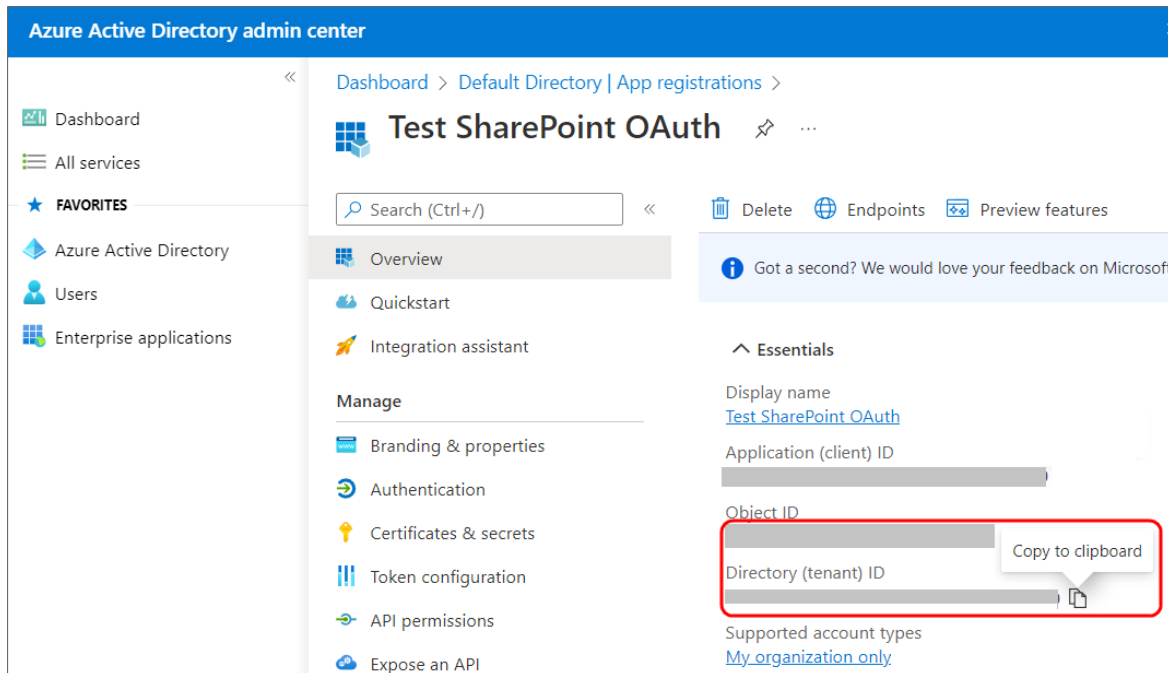


2. In the **Create New Data Source** screen, enter an **Name** for the Datasource, then click the **OK** button to create the Datasource and open its configuration screen.
3. On the **Properties** tab of the Datasource definition, change the **DataSource Type** to "SharePoint OAuth (Tenant)".
4. Set the **SharePoint Site URL** to the URL your SharePoint Online server.
5. To set the **Client ID** property, go to the Azure window, and using the "Copy to Clipboard" icon, copy the value in the **Application (client) ID** property, then paste it into the **Client ID** Property of the

Datasource definition.




6. Similarly, you'll need to copy the value of the **Directory (tenant) ID** property in Azure to the **Tenant ID** property of the Datasource definition.



7. To set the certificate to use for this Datasource, click the **Browse** button for the **SharePoint Certificate File** property, then locate and select the PrivatePublicKeys.pfx file you created earlier (either with certreq.exe or PowerShell).
8. Enter the certificate **Password** that you created for the PrivatePublicKeys.pfx file.

9. Click the **OK** button to save your changes, then update the Datasource definition by selecting **Update** from the **OK** dropdown menu at the upper right corner of the page.
10. Click the **Test Connection** button to ensure that the Datasource can connect properly to SharePoint.

### SharePoint OAuth (Tenant) Datasource Properties

Datasource Connection Name Icon 

SharePoint Datasource

---

#### PROPERTIES

**Description**

Enter a brief description of this Object

**Datasource Type**

SharePoint OAuth ▼

**Sharepoint Site URL**

Client ID

Tenant ID

Sharepoint Certificate File (\*.pfx) Browse

Certificate Password (optional)

In addition to the standard **Description** property, setting the **Datasource Type** property to *SharePoint OAuth* enables configuration of the connection properties listed below.

#### SharePoint Site URL

The fully-qualified URL that connects to the SharePoint installation.

#### Client ID

The value of the **Application (client) ID** property contained in the App Registration screen in Azure.

#### Tenant ID

The value of the **Directory (tenant) ID** property contained in the App Registration screen in Azure.

#### SharePoint Certificate File

A **Content Picker** than enables you to browse to and upload the certificate (.PFX) file to Azure.

#### Certificate Password

The password that you configured for the certificate (.PFX) file when you created it.

## Configuring the SharePoint OAuth (Site) Datasource #

Configuring the [SharePoint OAuth \(Site\)](#) Datasource is far less complex than configuring the tenant-level Datasource, and requires no certificate to be created or uploaded to Azure. To add Process Director as an application in your Azure Active Directory portal at the Site level, complete the steps below after signing into your Azure portal ([portal.azure.com](http://portal.azure.com)):

### 1. Configure SharePoint Site Permissions

1. Navigate to the site you want to configure access for in your tenant. This is typically of the form <https://mytenant.sharepoint.com>, replacing “mytenant” with the appropriate name.
2. Adjust the URL to [https://mytenant.sharepoint.com/\\_layouts/15/appregnew.aspx](https://mytenant.sharepoint.com/_layouts/15/appregnew.aspx).
  - a. Click the buttons to generate both a **Client Id** as well as a **Client Secret**.
  - b. Select the **Client Id** value, copy the text and store the value somewhere safe to be used in later steps in this guide.
  - c. Select the **Client Secret** value, copy the text and store the value somewhere safe to be used in later steps in this guide.
3. Now you need to grant permissions to newly registered app (AKA principal). Navigate to [https://mytenant-admin.sharepoint.com/\\_layouts/15/appinv.aspx](https://mytenant-admin.sharepoint.com/_layouts/15/appinv.aspx). It's important to note the addition of “-admin” to your site's normal name.

Office 365 Admin

SharePoint admin center

site collections

infopath

user profiles

bcs

term store

records management

search

secure store

apps

sharing

settings

configure hybrid

device access

Create Cancel

**App Id and Title**  
The app's identity and its title.

App Id: [727176a-64c9-4697-a713-0b] Lookup

Title: Process Director

App Domain: www.localhost.com  
Example: "www.contoso.com"

Redirect URL: https://www.localhost.com/  
Example: "https://www.contoso.com/default.aspx"

**App's Permission Request XML**  
The permission required by the app.

```
<AppPermissionRequests AllowAppOnlyPolicy="true">
<AppPermissionRequest Scope="http://sharepoint/content/sitecollection" Right="FullControl" />
</AppPermissionRequests>
```

Create Cancel

- a. Add your Client Id as **App Id**.
  - b. Add the XML as shown, reproduced here to aid in copy and paste. Note, there are other, more restrictive options that can be considered listed in Table 1 at Microsoft's documentation topic, [Add-in permissions in SharePoint](#). Be careful using other values as it may prevent Process Director from working correctly.
 

```
<AppPermissionRequests AllowAppOnlyPolicy="true">
  <AppPermissionRequest Scope=
e="http://sharepoint/content/sitecollection" Right="FullControl" />
</AppPermissionRequests>
```
  - c. Set the **Title** to "Process Director".
  - d. Set **App Domain** to the fully qualified domain name of your Process Director deployment.
  - e. Set the **Redirect URL** to the URL of your Process Director deployment.
4. Click **Create**.
  5. Click **Trust It** in the follow-up prompt.

## 2. Configure the Datasource

1. In a Process Director **Content List** folder, select **Data Source** from the **Create New** menu.
2. Supply a **Name** and click **OK** to open the new Datasource definition.
3. Set the **Datasource Type** drop-down to "SharePoint OAuth (Site)".
4. Add the **SharePoint Site URL** for your SharePoint Online installation.
5. Add the **Client ID** (AKA Application Id) and **Client Secret** from SharePoint that you set aside in the steps for **Configure SharePoint Site Permissions** above.
6. Click **OK** then select the **Update** item from the **OK** menu at the top right corner of the page to save the configuration.
7. Click the **Test Connection** button to test your connection to the SharePoint site.

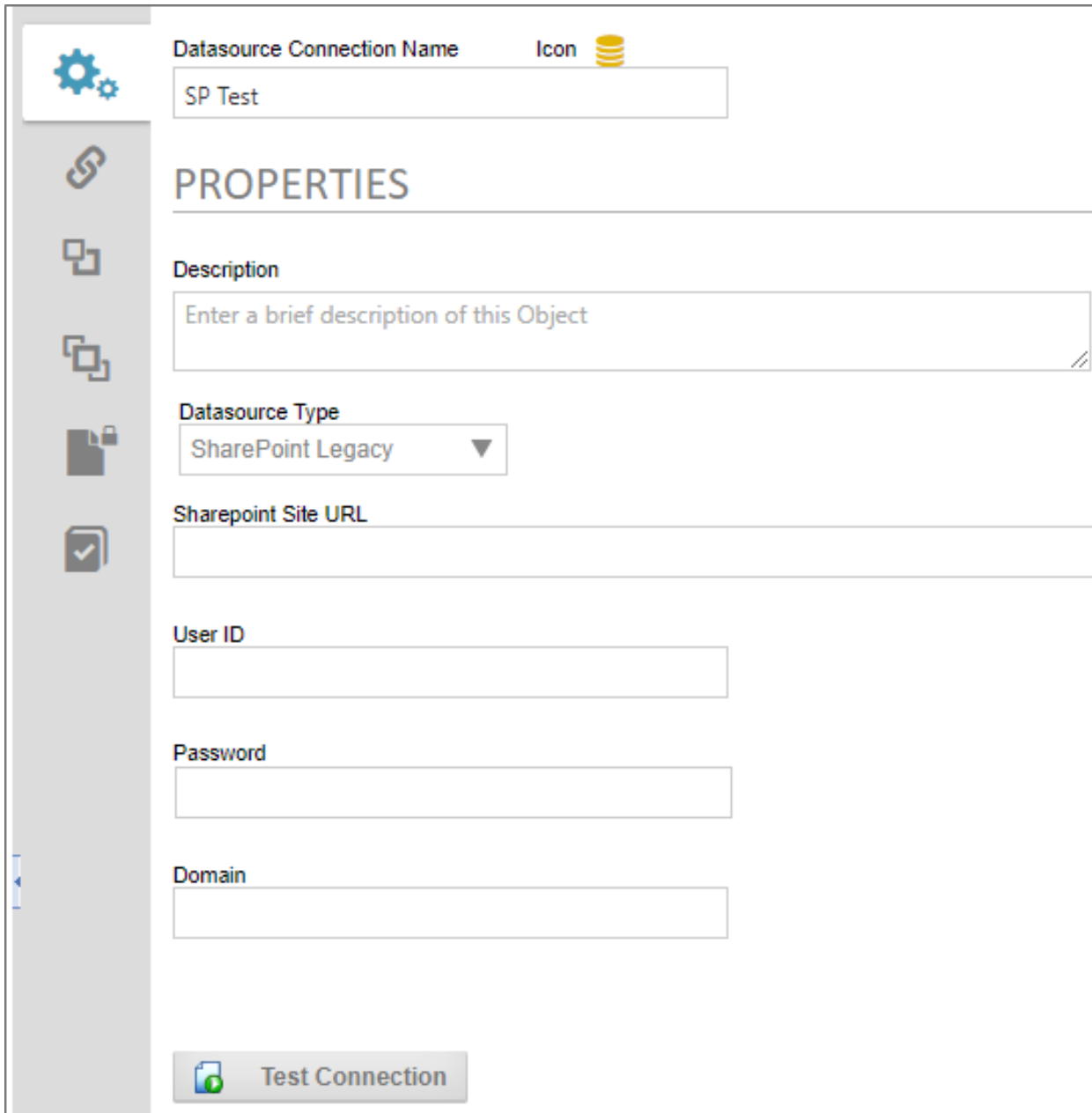
A successful test means that your Datasource is correctly configured and is connecting to the SharePoint site correctly.

## Conclusion

Congratulations! Assuming that you've correctly followed the instructions above, you've now configured both SharePoint Online and Process Director. You can now use this Datasource and the SharePoint Custom Tasks in Process Director to integrate your SharePoint sites and data with Process Director.

## Sharepoint Legacy Datasource #

For connections to pre-OAuth versions of SharePoint, the SharePoint Legacy datasource type enables you to create a datasource connection to the SharePoint server.



The screenshot shows a configuration window for a Datasource Connection. On the left is a vertical toolbar with icons for settings, linking, cloning, saving, and confirmation. The main area is titled 'PROPERTIES' and contains the following fields:

- Datasource Connection Name:** A text box containing 'SP Test'. To its right is an 'Icon' field with a database cylinder icon.
- Description:** A large text area with the placeholder text 'Enter a brief description of this Object'.
- Datasource Type:** A dropdown menu currently set to 'SharePoint Legacy'.
- Sharepoint Site URL:** An empty text box.
- User ID:** An empty text box.
- Password:** An empty text box.
- Domain:** An empty text box.

At the bottom of the configuration area is a button labeled 'Test Connection' with a document icon and a green checkmark.

There are four properties to configure to create this datasource.

The **Sharepoint Site URL** property enables you to enter the fully qualified URL of the Sharepoint server to which you wish to connect.

The **User ID** must be the user ID for a valid SharePoint User, while the **Password** property will be the password for the specified user. The **Domain** property is the SharePoint domain that contains the specified user.

Once you've configured the datasource, you can click the **Test Connection** button and a message banner will appear, notifying you whether the connection was successful.

## Other Datasource Types

To see more information about different [Datasource Types](#) and their configuration, please refer to the following topics:

- [Common Datasources](#)
- [Excel Datasources](#)
- [File Datasources](#)
- [Social Datasources](#)

## Social Media Data Sources

Process Director provides Datasources for some social media services. These social media Datasources allow you to use Process Director to extract information from social media accounts, or to post messages to them. Process Director already has the appropriate application access tokens built into the product, so you can immediately set up Datasources without having to create a Social Media app. You merely need to acquire the appropriate access tokens—a feature that is also built into the product. Note, these Datasources use BP Logix-provided social media applications. For production use, we recommend creating your own social media applications at the appropriate development web site for the social media platform.

## Creating Social Media Data Sources

Social Media applications generally use the OAuth authentication technology, which generally provides that you provide two pieces of information. You need an access token, and an access token secret.

In the Datasource configuration screen, select the appropriate social media Datasource type from the Datasource Type dropdown to display the configuration options for an OAuth-based Datasource.

If you don't yet have your own app created through the relevant developer's portal, you can click the **Get OAuth Tokens** button to obtain an authorization token using Process Director's built-in access token. If you already have an app, simply enter the **Access token** and **Access token secret** in the fields provided. Your app must not be in development mode, and must be publicly available before you can generate the appropriate OAuth Tokens.

You can click the **Test Connection** button to ensure the data connection is valid. Once you've validated the Datasource, click the **OK** button to close and save the Datasource.

You may now use this Datasource with the Social Media Custom Tasks.

There are some additional considerations to keep in mind:

OAuth tokens are site specific. They are only valid for the particular site you are requesting from. If you have OAuth tokens that were generated from a different site, they won't work if you put them into the Datasource configuration. They need to be generated from the IIS server where the Process Director system is installed. This requirement means that if you have separate development, staging, and/or production systems that are hosted on different IIS servers, you'll need to regenerate the OAuth tokens when you move the Datasource from the development to staging to production servers.

## Social Media OAuth Providers and Settings

When you create a social media app, all of the social media sources that use OAuth for authentication provide you with an Access Token and Access Token Secret, though they all have slightly different names. To find the appropriate values to place in the Access Token and Access Token Secret for each of the different social media platforms, please refer to the list below. The list also includes the relevant developer portal for creating apps for the social media platform.

PLATFORM	APP DEVELOPMENT URL	ACCESS TOKEN	ACCESS SECRET	TOKEN
LinkedIn	<a href="https://www.linkedin.com/secure/developer">https://www.linkedin.com/secure/developer</a>	API Key	Secret Key	
DropBox	<a href="https://www.dropbox.com/developers">https://www.dropbox.com/developers</a>	App Key	App Secret	
Microsoft OneDrive	<a href="https://account.live.com/developers/applications">https://account.live.com/developers/applications</a>	Client ID	Client Secret	
Google	<a href="https://console.developers.google.com">https://console.developers.google.com</a>	Client ID	Client Secret	
box.net	<a href="https://apps.app.box.com">https://apps.app.box.com</a>	Consumer Key	Consumer Secret	

## Other Datasource Types

To see more information about different [Datasource Types](#) and their configuration, please refer to the following topics:

- [Common Datasources](#)
- [Excel Datasources](#)
- [File Datasources](#)
- [SharePoint Data Sources](#)

## DropDown Object

This object is used to populate the options available in a [Dropdown](#) control on a form with the data you create in this object. The [DropDown Object](#) is reusable, which means that the same [DropDown Object](#) can populate [Dropdown](#) controls on many Forms. Some examples of data you might want to create with a [DropDown Object](#) might be a list of states/provinces, or countries. Any static list of items can be created using a [DropDown Object](#), then that [DropDown Object](#) can be used to populate any [Dropdown](#) control on any Form controls that needs to use it. Because the [DropDown Object](#) is fully reusable, you may wish to create a [\[DropDown Objects\]](#) folder in the Partition root of your installation, so your implementers can have all the available [DropDown Objects](#) accessible from the same location.


You can create a [DropDown Object](#) by selecting [DropDown Object](#) from the [Create New...](#) menu dropdown. Enter a name for the object and click the [OK](#) button to open the new object definition. Once the object definition opens, navigate to the [Items](#) tab, where you'll perform the primary configuration of the object.

On the [Items](#) tab you can add, edit, or delete dropdown items by using either the [Table View](#) or [Textbox View](#) tabs to configure the data. Both tabs contain exactly the same list items in the same order. The two



tabs merely provide different methods of configuring the same list of items. You can switch between the two tabs at will during configuration.

## Table View Tab #

DropDown Object Name    Icon 


Documentation

**ITEMS**

Table View    TextBox View

Add null/blank entry at top of the DropDown with Text: [Select One]

Display Text	Value (set to Display Text if left blank)	
Item 1	1	↑ ↓ ✕
Item 2	2	↑ ↓ ✕

 Add DropDown Entry

For transition to TextBox view

Character to separate Name from Value :    Character to separate entries Enter

The **Table View** tab can be used to create a list using the product UI.

The first property to configure is labeled, **Add null/blank entry at top of the DropDown with Text**. When checked, this property adds a blank entry to the top of the dropdown list, using the text you enter into the text box as a user prompt when the Form is displayed. Please refer to the section labeled, [Using Blank Values](#), below, for more information about the way this setting affects data entry on the Form.


 **BP Logix recommends that you enable the Null/Blank setting for nearly all use cases.**

Click on the **Add DropDown Entry** button to add an item to the list. Enter a **Display Text** and optional **Value** for each item. If you leave the **Value** column blank, the **Display Text** value will also be saved as the **Value** for the associated **Dropdown** control(s). Otherwise, the **Display Text** will be the text that is displayed to users in the associated **Dropdown** control(s), while the **Value** is the value that will actually be saved with the Form instance.

You can manually set the order of the items to be displayed by using the **Up/Down** arrow icons to reorder the items in the list. You can also remove items from the list by clicking the **Delete** (X) icon.

## TextBox View Tab #

You can manually enter data into your list as rows in the **TextBox View** tab. You can type or copy and paste text directly into the text box.

DropDown Object Name    Icon 

Documentation

ITEMS

Table View    **TextBox View**

Enter data below that will be imported as dropdown rows. You can enter this data directly, or Copy-Paste from another application.

[Select One]:"  
Item 1:1  
Item 2:2

Character to separate Name from Value    :    Character to separate entries    Enter

By default, each line in the text box creates a new entry in the list. The list displayed above is the same list displayed in the **Table View** tab. By default, the configuration syntax for the text box view is to provide the **Display Text**, followed by a colon (:), then the **Value** text, e.g.:

**DisplayText:Value**


Note that our blank item is also displayed as the **first** list item on the **TextBox View** tab. Please refer to the section labeled, [Using Blank Values](#), below, for more information about the way this setting affects data entry on the Form.

## Text List Configuration #

Both the **Table View** and **Textbox View** tabs share two properties that enable you to alter the default configuration for the text view of any list. On the **Textbox View** tab, these are displayed as standalone properties, while on the **Table View** tab, they are contained in a display section labeled, [For transition to TextBox view](#). In either case, they are the same properties, and govern the syntax used for list items in the textbox view.

Character to separate Name from Value    :    Character to separate entries    Enter

- **Character to separate Name from Value:** This property defaults to the colon (:) as the separator character between the **Display Text** and **Value** properties for each list item. You can select a different character to use as the separator by selecting it from the dropdown control for this property.
- **Character to separate entries:** This property defaults to the [ENTER] key (new line) as the separator character between each list item. You can select a different character to use as the separator by selecting it from the dropdown control for this property.

 BP Logix recommends using the default configuration for these two properties, unless some technical issue requires the use of different characters, such as custom scripting/external data issues. In the vast majority of use cases, this won't be necessary.


## Using Blank Values #

When using the property labeled, **Add null/blank entry at top of the DropDown with Text**, the blank entry will be the default entry that is displayed to the user on the form. For instance, if we configure the Property as shown below:

**Add null/blank entry at top of the DropDown with Text:** [Select One]

The user, when filling out the Form, will see the default value of the associated **Dropdown** control as:

Department: [Select One] ▼

 Having a blank entry is required if you wish to make the Dropdown control a required field on the form. If you don't have a blank item as the default value, the associated Dropdown control, when displayed, will default to the first item in the list, and the control will always have a valid value. As such, even if the user makes no change to the control, the Dropdown will always be evaluated as having met the "Field is Required" setting. Only by creating a blank value as the default can the field be evaluated as not having as required value, and fail the "Field is Required" setting.

On the **Textbox View** tab, you can also manually add a blank item as the first item in the list by using the syntax:

`DisplayText:""`

...where **DisplayText** is the text you desire to show to users as a prompt when filling out the form. The two quotation marks ("" ) after the colon specifies that this list item has a blank/null/empty value.

For instance, if you look at the screenshot example in the [Textbox View tab section](#), above, you can see the first line in the text box is:

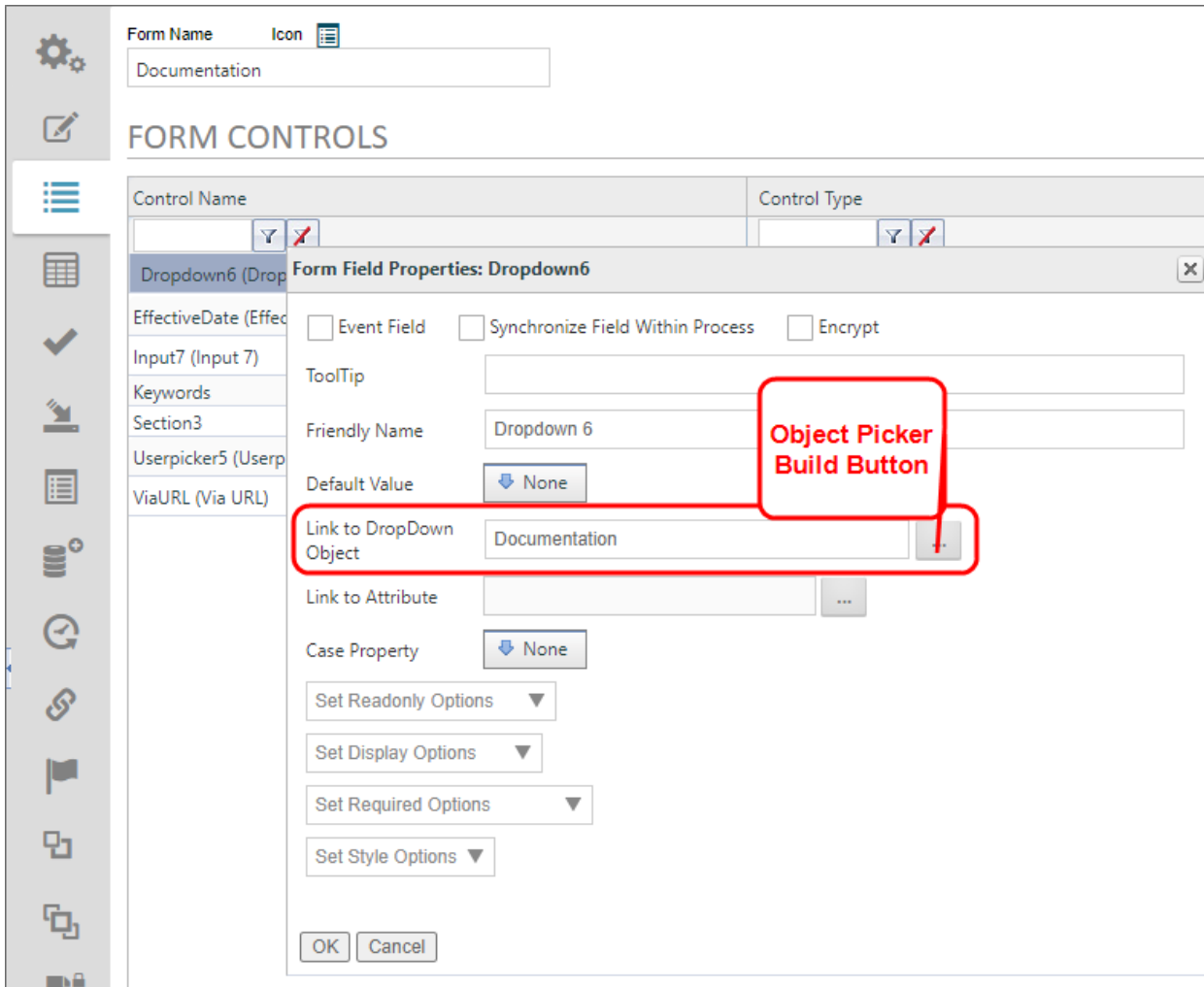
`[Select One]:""`

The blank item must **always** be the first item in the list, because Process Director will always default to the first item in the list as the default value for the **Dropdown** control.

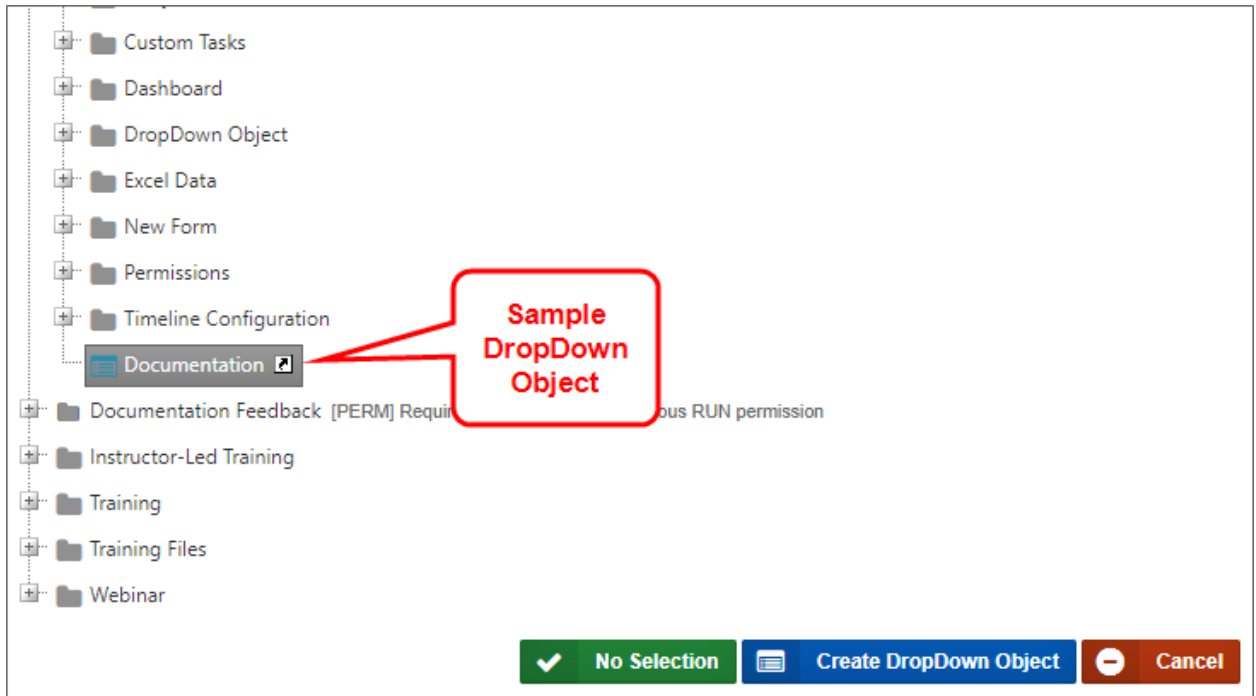
## Associating the DropDown Object with a Dropdown Control #

Once you've finished configuring the [DropDown Object](#), you can save and close your configuration by clicking the **OK** menu button at the top right corner of the screen.

Now that the object has been saved to the [Content List](#), you can associate the [DropDown Object](#) to a [Drop-down](#) control on a Form by opening the Form definition, and navigating to its [Form Controls](#) tab. Open the field properties for the [Dropdown](#) control you wish to associate with the [DropDown Object](#).



The [Link to DropDown Object](#) property is an [Object Picker](#) that opens a [Content List](#) navigation screen when you click the **Build** button.



You can navigate to the desired **DropDown Object** to select it, and the **Dropdown** control will automatically be associated with the correct **DropDown Object**.

## Forms

The Form object is the primary means by which users submit information to the system. Process Director Forms are web-based, electronic forms that can be designed online, published and made available to end users. Users fill out and submit Forms online from within their browser. In most cases, submitting a Form automatically starts a Process Timeline that models a process to determine how the form should be routed to different participants during the process. The Form serves as the primary storage object for all of the data that the process requires, and is usually displayed to all participants in the process, both to provide information for decision-making, and to record changes or additional data the process may require as it proceeds.

Forms are primarily created using the [Online Form Designer](#). Additionally, ASP.Net forms can be developed using a development tool like Visual Studio. A Visual Studio plugin is available for using Visual Studio to create forms as ASCX custom controls, with the appropriate license option.

**!** The Word Form Builder has reached the end of its life cycle. The BP Logix plug-in for Microsoft Word uses Microsoft's ActiveX technology, which Microsoft deprecated in 2015 with the release of the Microsoft Edge browser. With Internet Explorer's end of life on 15 June 2022, support for both Internet Explorer and ActiveX technology from Microsoft has ended. BP Logix **STRONGLY** recommends that ALL new forms be created with the Online Form Designer, and that customers convert existing Word forms to the OFD format using the conversion tool provided on the Edit tab of the form definition.

Each Form definition contains not only the properties that govern its operation, but the design template that defines how the form will appear to end users. Forms enable you to show or hide different parts of a form at different times, implement validation rules, display data from external sources, and perform many other operations. As such, the Form definition is the most complex Process Director object.

To view the documentation about the Form definition, you can use the Table of Contents displayed on the upper right corner of the page, or use one of the links below.

[Managing Forms](#): General information about creating and managing Form definitions in the [Content List](#), and other general features available in Forms.

[Configuring Forms](#): Detailed descriptions of the configuration tabs and properties available when configuring a Form definition.

[Online Form Designer](#): The design system used to specify the visual appearance of Forms, and the Form controls available for use.

[Email Templates](#): Using Forms to build customized email message templates to use in notifications.

## Form Definition

Form definitions are stored in the Process Director database. Form definitions have properties similar to other objects, such as documents.

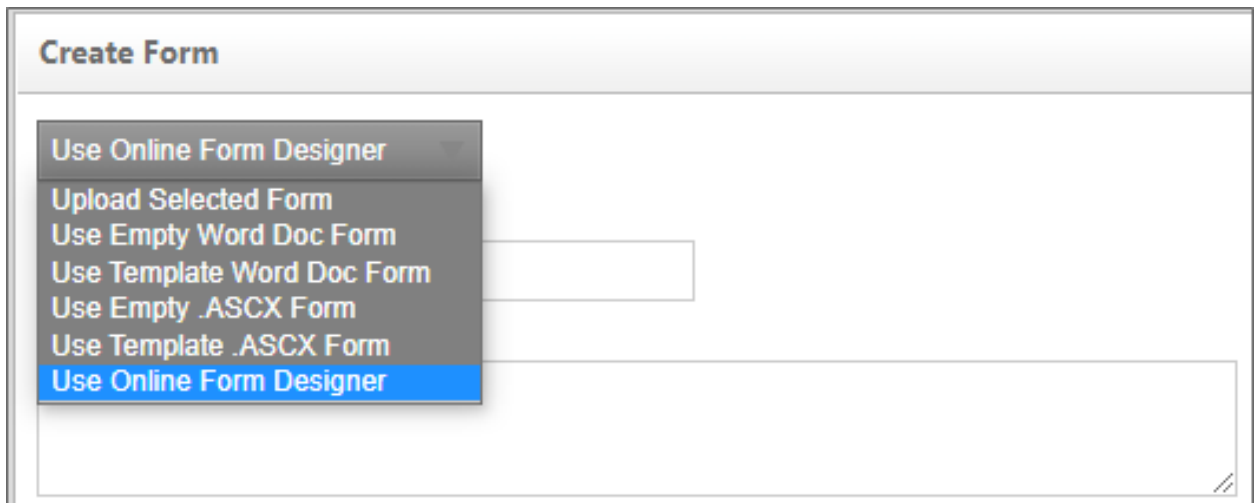
A Form definition is made available to users through their profile, the [Content List](#) or from within a Knowledge View. The Form definition enables a user to fill out and submit form based information directly from within their browser. When a Form is submitted, the form data entered by the user is saved as a form data object in a new Form instance. This form data is stored in the Process Director database and can be evaluated by any process model. When viewing the data for a completed form, the Form definition that was used to submit the information is used to display the data.

### Adding a New Form Definition #

To create a new Form definition, use the [Create New](#) dropdown in the [Content List](#) and choose the [Form Definition](#) menu item. A user must have Modify permission to the parent folder to create a Form definition. If a user doesn't have the appropriate permission, the hotlink won't appear.



The **Create Form** dialog will display, enabling you to select the Form template format you'd like to use from the dropdown. Enter a required name for the form and optionally give it a description. Click the **OK** button to create the form.



In nearly every case, you should use the default Online Form Designer format for creating new Forms. There are, however, several options you can select for creating your Form template:

- **Upload Selected Form:** This option will allow you to choose a Form template that you've already created. You can choose an existing word document template or a .NET form in ASPX format.

- **Use Empty .ASCX Form:** This option will create a blank ASP.NET web form that uses c# as the CodeBehind programming language. This form must be edited in a C# programming IDE, like Visual Studio 2013. The ".ASCX" designation refers to the file extension for ASP.NET web forms. This option is primarily for programmers, not regular implementers.
- **Use Template .ASCX form:** There is an existing ASP.NET web form template in Process Director that contains some basic fields and instructional text. You may choose this form to use for training, or as a basis for a new production form. This option is primarily for programmers, not regular implementers.
- **Use the Online Form Designer:** Process Director v4.06 and higher contains a built-in Online Form Designer that you can use to create Form Templates from within the Process Director interface. This is the default option for creating new Forms.

Prior to v5.44.1000, additional Form creation options for the Word Form Builder will be displayed:

- **Use Empty Word Doc Form:** This option will create a blank Word document template that you can then design manually. **END OF LIFE**
- **Use Template Word Doc Form:** There is an existing Word template in Process Director that contains some basic fields and instructional text. You may choose this form to use for training, or as a basis for a new production form. **END OF LIFE**



BP Logix strongly recommends that you do NOT use either of the Word document options for creating new forms, as Microsoft has sent these technologies to end of life, and they are no longer supported.

Once the form is created, the Online Form Designer will open automatically to enable editing immediately.


## Updating a Form Definition #

You can immediately access the online template for any Form that was built using the Online Form Designer by clicking the **Edit** tab of the Form definition. Process Director will automatically check out the Form template and display it in the designer.









The screenshot shows a software interface for defining a form. On the left is a vertical sidebar with three icons: a gear (Settings), a pencil (Edit), and a list (Properties). The 'Edit' icon is highlighted with a red square. The main area contains a 'Form Name' field with the value 'Documentation' and an 'Icon' field with a list icon. Below this is a 'PROPERTIES' section with a 'Description' field containing the placeholder text 'Enter a brief description of this Object'. At the bottom, there is a label for 'Instantiated Form Name'.



## Forms In the Content List #

When a Form is filled out and submitted, the data is stored in the [Content List](#) as a Form Instance under the Form definition. To view the Form instance, click on the Form instance's name or the  icon in the Content List.

Form instances are similar to other objects in the Process Director database, supporting permissions, categories, properties and processes. Users must be given the appropriate permissions to view Form instances in the [Content List](#). Users that are given View Children permission to a Form definition will automatically be given View permission to Form instances stored under that Form definition.

/Training/BP Logix Getting Started Sample Project	
<input type="checkbox"/>	<b>Name</b> ▲
<input type="checkbox"/>	 <b>Active Travel Requests</b> This Knowledge View shows active travel requests, i.e., those that are still in the approval process.
<input type="checkbox"/>	 <b>ChargeNumbers.xlsx</b> Excel spreadsheet containing a list of charge number to import into the internal database.
<input type="checkbox"/>	 <b>ERP Import</b> Imports Excel data into Internal Database
<input type="checkbox"/>	 <b>Is Finance Approval Needed?</b> Business rule that determines where finance approval is needed based on the request amount.
<input type="checkbox"/>	 <b>My Expense Requests</b> Airfare
<input type="checkbox"/>	 <b>ReasonForTravel</b> List of travel reasons
<input type="checkbox"/>	 <b>Travel Expense Approval Process</b> This timeline creates the approval process for the request.
<input type="checkbox"/>	 <b>Travel Expenses Request</b> This form creates the travel request.

## Form Definition URL

Forms can be linked and run from outside of Process Director using a manually-configured URL (for example, linked from your Intranet portal). The default URL can be found on at the bottom of the Form Definition's Properties page, and looks similar to this:

```
https://
ServerName.com/form.aspx?pid=PP&formid=NN&nohome=0&completepageprompt=0&completepage=&completetext=
```

Where [ServerName](#) is the hostname of BP Logix, [PP](#) is the internal ID of the Partition, and [NN](#) is the internal ID of the Form Definition. For users of Process Director v5.12 and higher, the URL is displayed on this tab as a hyperlink.

A number of URL parameters can be added to the URL's existing parameters to alter the way Process Director responds to the URL request. The URL parameters may be used in any order.

```
https://ServerName.com/form.aspx?pid=PP&formid=NN&nohome=1&completepageprompt=0
```

- The **nohome** parameter controls whether the home page should be opened when the link to the [Task List](#) or Form is clicked. Set **nohome** to 1 to ensure that the home page isn't opened.
- The **completepage** and **completetext** parameters specify the URL of the page that completing the Form should redirect to and the text displayed with the URL, respectively. When using **completepage** parameter, you must include the "**ServerName.com/**" or "https://" when specifying your URL. In other words, the URL must be a fully-qualified URL, and not just a relative path or other partial URL. Additionally, the **completepage** URL must, for security reasons, point to a page *inside* the Process Director domain. You can't redirect to an external URL. We recommend that, if you want to direct users to an external URL, you create a custom redirect page inside the /custom folder of your process director installation, and use that page as the **completepage** URL. That custom redirect page can then, in turn, redirect the users to an external URL.
- The **completepageprompt** parameter supplies the page completion prompt message. If **completepageprompt=0**, then the default "completed" page will NOT be displayed. If you are trying to redirect the user to a specific URL after the form is submitted, you'll most likely want to suppress the "completed" page.
- The **prinstid** parameter supplies a process instance ID to display the form instance that is associated with a specific process instance.
- The **findtask=1** parameter will display the form in task context. This will generally open the form in the context of the current task. Some form instances, of course, may be associated with multiple process instances, so you can give an additional hint about the process you are trying to use by adding the **linkprid** parameter to pass the ID of the specific process instance which you want to open in task context.

Process Director v5.39 and higher adds additional URL parameters to control how the form responds to the [Save and Close](#) button, when used.

- The **saveformpage** and **saveformtext** parameters specify the URL of the page that saving the Form should redirect to and the text displayed with the URL, respectively. The usage specifics of the **completepageprompt** and **completetext** parameters listed above also apply to this parameter.
- The **saveformpageprompt** parameter supplies the page save prompt message. If **saveformpageprompt=0**, then the default "completed" page will NOT be displayed. If you are trying to redirect the user to a specific URL after the form is saved, you'll most likely want to suppress the "completed" page.

You can also set the value of Form fields in the URL that will default or override the values for the specified fields. For security reasons, you'll need to specify the fields you want to set via URL parameter beforehand. There are two methods for setting the value via a URL parameter.

### Setting a default value

If you **only** need to set the default value of a field via URL for a new Form instance, you can use the following procedure to configure the field.

In the **Form Fields** tab of the Form definition, open the field properties for the field you wish to set. You must set a variable as the default value, and there are two ways to do this:

1. Set the **Default Value** as a Variable, then enter the name you'd like to use for the variable:

The screenshot shows a dialog box titled "Form Field Properties: ViaURL". At the top, there are three checkboxes: "Event Field", "Synchronize Field Within Process", and "Encrypt". Below these are input fields for "ToolTip" and "Friendly Name" (containing "Via URL"). The "Data Type" is set to "Text", with "Min" and "Max" value fields. The "Default Value" section is highlighted with a red box; it features a "Variable" button with a downward arrow and a text input field containing "Stuff". Below this are "Link to Attribute" and "Case Property" (set to "None") buttons. At the bottom, there are four dropdown menus: "Set Readonly Options", "Set Display Options", "Set Required Options", and "Set Style Options". "OK" and "Cancel" buttons are at the very bottom.

2. Set the default value to a string, and enter a generic variable into the field, using the syntax `{VAR:VarName, modifier=ModifierValue}`.

The screenshot shows a dialog box titled "Form Field Properties: ViaURL". It contains several configuration options:

- Event Field:
- Synchronize Field Within Process:
- Encrypt:
- ToolTip:
- Friendly Name: Via URL
- Data Type: Text (dropdown), Min: , Max:
- Default Value: String (dropdown), {VAR:Stuff} (text input, highlighted with a red box)
- Link to Attribute:  ...
- Case Property: None (dropdown)
- Set Readonly Options: (dropdown)
- Set Display Options: (dropdown)
- Set Required Options: (dropdown)
- Set Style Options: (dropdown)
- OK and Cancel buttons at the bottom.

Using the VAR object in a text string enables you to add additional modifiers to the variable. For instance, the syntax `{VAR:VarName, null="Value if null"}` specifies the value to use if the URL parameter isn't present.

The VAR (Variable) object is an *ad hoc* variable whose primary use is to serve as a placeholder for the value passed via the URL Parameter. This is the same implementation procedure used when [passing values to a Knowledge View in a URL](#).

No matter which of the two methods above that you implement, you can now use the Variable as a URL parameter. In the case of the above example, the Parameter would be

`https://ServerName.com/form.aspx?pid=PP&formid=NN&Stuff=value`

The default value will be filled with the value specified in the URL parameter.

This procedure is the simplest method for setting a field's value, and it doesn't require any further configuration. The drawback to this method is that it can only be used to set the default value of a field in a new Form instance, and so won't change a field value that has already been saved in an existing Form instance.

If you wish to set the value in an existing Form, you'll need to use the next method.

## Setting a Field Value Universally

This method enables you to set a field value via URL parameter at any time. For security reasons, however, you must first identify the field by setting the field name as configurable via URL in the Custom Vars file (vars.cs.ascx), using the [FormFieldsAllowDisabledURLUpdate](#) method.

Once you've specified the field name in the Custom vars file, you can now use a URL parameter to set the value any time you desire. In the URL parameter, the field is identified using the prefix "EXT\_", followed by the field name, to identify the field, e.g., "EXT\_FieldName".



Be sure to URL encode the values you specify in the URL Parameter.

### Example:

In the Custom Vars file, identify the fields to change via URL Parameter:

```
public override void PreSetSystemVars(BPLogix.WorkflowDirector.SDK.bp bp)
{
    //Fields to set via URL Parameter
    bp.Vars.FormFieldsAllowDisabledURLUpdate.Add("Field1");
    bp.Vars.FormFieldsAllowDisabledURLUpdate.Add("Field2");
}
```

Now, you can use the following URL parameters to set the field values:

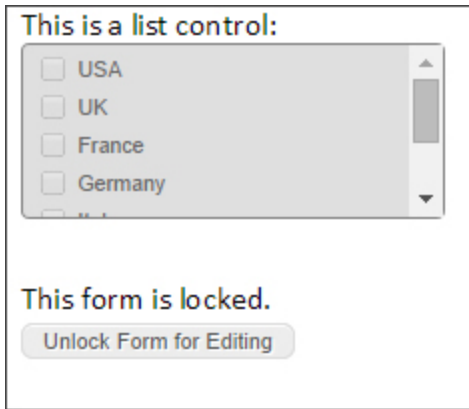
```
https://ServerName.com/form.aspx?pid=PP&formid=NN&EXT_Field1=Some+Value&EXT_
Field2=Another+Value
```

## Form Locking

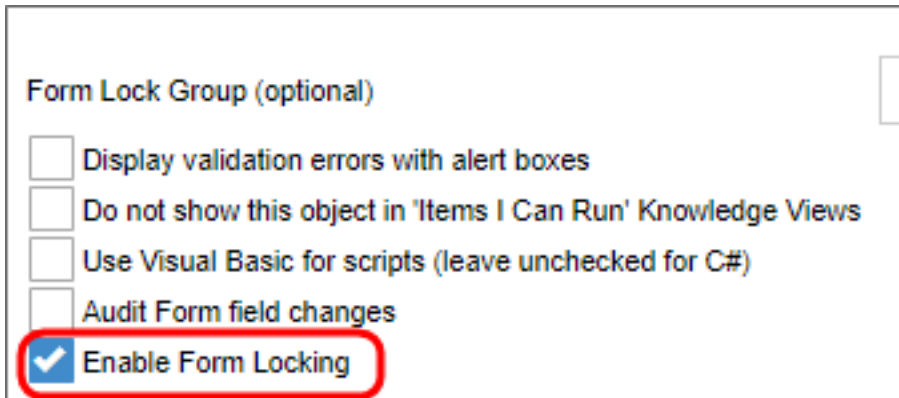


Form locking works only for authenticated users. Anonymous users cannot lock a form.

Forms can be locked by users, preventing other users from making changes to that instance of the form while the form is still locked. If a user views a form that is unlocked, but for which form locking is enabled, the form will be locked for all users except that user. The form remains locked until the user leaves the form. If another user views a locked form, all controls will appear disabled. You can display text for users viewing a locked form and optionally allow them to unlock a locked form using the [Lock Form control](#).



To enable form locking, select the **Enable Form Locking** checkbox in the **Options** section of the Form definition's **Properties** tab.



Form locking can be problematic in some cases. If one user needs to review the form, without making changes, while another user in the process is trying to complete a task, the reviewer would lock the form, preventing the task assignee from completing his work. If multiple users can access the form, then the user that opens the form first will lock the others out.

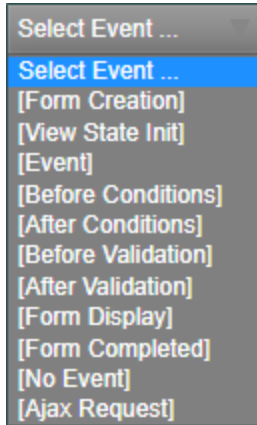
For this reason, the Form definition allows you to create conditions on the **Entire form is read-only when** option. This option allows you to determine conditions when the form can be made read-only. This prevents editing in specified circumstances, without locking the form for all users.

## Form Events

Form Events are actions that occur on a form, to which you can add custom actions of your own. Many events fire automatically as part of the form's natural life cycle. For instance, when a form is first opened, an event called Form Creation is fired. Some events, on the other hand, are manually fired. A **Button** control, for example, has an event that fires when the button is clicked, and this event is built into the **Button** control automatically. Additionally, form input controls can also be set as an **Event Field** in the properties you configure on the **Form Controls** tab of the Form definition. These controls will fire an event when the data in the control is changed. Any time an event fires, you can configure an action to run on the event, such as a Set Form Data action, or running a Custom Task. For example, you might place a **Button** control

on a form, and on the event for that button, run a Custom Task that converts the form to a PDF document.

Form events can be accessed from a Dropdown control that lists all of the available events on the form. This dropdown is primarily used on the [Set Form Data](#) and [Custom Task Event Mapping](#) tabs, though it appears anywhere that an event is applicable.



This dropdown displays all of the available events that can be used to attach actions to the Form. By default, the dropdown shows only the events built into the form's life cycle, but any buttons or other event controls you place on the form will be automatically added to the selections displayed in the dropdown. The table below describes the default events that occur as part of the form's life cycle.

Event	Description
Form Creation	Fired when a Form instance is called for display.
View State Init	Fired on the preparation of the Form for display.
Event	Fired on any Event
Before Conditions	Fired before any Form conditions, such as visibility or enabling conditions on controls, are called.
After Conditions	Fired after all Form conditions have been processed.
Before Validation	Fired when the Form has been submitted, but before any Form validation conditions are evaluated.
After Validation	Fired after all validation conditions have been processed.
Form Display	Fired when the form is actually displayed in the browser. All Fill Dropdown actions, form calculations for <a href="#">Calculation</a> and <a href="#">Sum</a> controls, and other automated form update events, are automatically processed on the Display event.

Event	Description
Form Completed	Fired after the form's display has been ended (i.e., when the form closes), and just prior to the form being disposed from memory.
No Event	Placeholder to prevent an action from firing on any event.
Ajax Request	Fired when a client-side JavaScript Ajax request is processed.

So far, we've just talked about Form events in the context of the user interface, but Process Director also enables custom scripts to be run, and the script actions also run on the same events that UI configuration uses. Additionally, Custom Tasks also run on the same events.

The order of operations for these different actions is:

- **Custom scripts:** Scripted actions always run first on any event.
- **UI and Datasets:** After the custom scripts complete, the UI or Dataset actions, such as Set Form Data or Fill Dropdowns, will run.
- **Custom Tasks:** Custom Task run after all other actions.

The order in which the actions run is very important. For instance, if you want to configure the **Set Form Data** tab to run an action that is based on some data pulled from a Custom Task, you can't run both the Set Form Data action and the Custom Task action on the same event. The Custom Task will **not** run before the Set Form Data action, only after it. In such a case, you might have to run two Custom Tasks, with the first Custom Task retrieving the data, then running the Set Form Data Custom Task second, to ensure that the operations happen in the order you desire.

Additional advanced information on Form events from a scripting perspective can be found in the [Custom Scripting](#) and [Form Class](#) documentation in the Developer's Guide.

## Form Data Caching

Users of Process Director v4.51 and higher may use a feature named Form Data Caching, which BP Logix has implemented to improve the speed of querying and searching for form data, and to improve performance of the Process Director server. This feature affects the performance of both Knowledge Views and SQL views that you create in the product.

### tblFormData

We often think of the data from a Form instance being stored in its own table, like a spreadsheet, where each data field is a column and each individual form instance is represented by a row in the table or spreadsheet. Process Director doesn't store form data in that way, however. Instead, there is a single database table named **tblFormData** that stores the data for every field from every form instance. As you might imagine, over time, this table gets quite large. Even a reasonably-sized installation of Process Director may contain millions of records in **tblFormData**. So, when you run any query or Knowledge View for form data, Process Director, by default, has to parse through all of the records in **tblFormData** to return the results you want, which may take an inconveniently long time and a large amount of system resources. The



more form instances that exist in your installation, the longer it takes to parse through this ever-growing recordset.

Form Data Caching helps to alleviate this problem. When you use this feature, Process Director makes a cache table that contains only the instance data for the Form definition for which you've configured the feature. You can then run your Knowledge Views or SQL Views on that cached table, instead of [tblFormData](#). Doing so may provide significant performance improvements, and less resource usage, since you are only having to parse through the relevant data from a single form definition.

## Implementing Form Data Caching

In the **Options** section of the **Properties** tab for each Form definition, there is check box property named **Enable Form Data Cache**. Checking the property will enable process Director to create the cache table in the database when you direct a cache to be created. The temporary cache table will be named with the naming convention, `tblFDC_GUID`, with the GUID portion being the actual GUID of the Form Definition.

To direct Process Director to create a Form Cache, you can call a page named "build\_form\_cache.aspx". This page is located in the root directory of the web site, and can be accessed with the URL, "http://localhost/build\_form\_cache.aspx", where `PDServerName` is the base URL of your process Director installation. This page will build the form caches for all of the forms that are configured to use Form Data Caching. The URL does, however, accept the URL parameter "`?FORMID=GUID`", where, again, the GUID would be the GUID of a specific Form definition that is configured to use Form Data Caching. When you use this URL parameter, Process Director will create the Form Cache only for the form you specify in the parameter, instead of creating the cache for all configured forms.

To keep the form cache relatively current, you can use the Windows Scheduler to call the `bputil.exe` utility to run this page on a scheduled basis, by using the commands (Where `PATH` is the installation directory for Process Director, e.g. `c:\Program Files\BP Logix\Process Director\`):

```
"PATH\bputil.exe" SU http://localhost/build_form_cache.aspx
```

...to create the whole Form Cache

or

```
"PATH\bputil.exe" SU http://localhost/build_form_cache.aspx?FORMID=GUID
```

...to create the cache table for a single Form.

For Process Director v5.39 and higher, an additional URL parameter, `forminstid`, will pass a Form Instance ID to update the form cache for a specified form instance.

```
"PATH\bputil.exe" SU http://localhost/build_form_cache.aspx?forminstid= FormInstanceGUID
```

Once the cache table has been created, the table will appear in any Data Source objects of the Internal Database **Datasource Type** as a table with the prefix "`tblFDC_`" followed by a GUID, e.g.: `tblFDC_04215238-517b-47b6-9`. The example below displays several Form Data Cache tables in a Process Director Data Source object.



## Querying the Cache

You can query the cache in one of two ways: Using a Knowledge View, or creating a SQL View.

### Using a Knowledge View

In the **Configure** tab of a Knowledge View definition, there is a check box property named **This Knowledge View will only use cached Form Instance Data**. Assuming that 1) the Knowledge View returns Form instance data, 2) the Form definition has been configured to use caching, and 3) a cache has been created for the Form, the Knowledge View will query the cache for the form, and not **tblFormData**. If the form definition does **not** have the cache check box set, the Knowledge View will ignore the cache configuration setting and query **tblFormData** instead. If the Form and Knowledge View are configured to use Form Data Caching, but the cache table doesn't exist, then the user will receive an error message when the Knowledge View runs.

### Using a SQL View

You can [create SQL Views](#) for any form in the **Form Data SQL View** tab of a Form Definition, which contains a check box property named **Use cached tblFormData table for SQL VIEW?**. Assuming that the Form is configured to use Form Data Caching, then, when this property is checked, Process Director will create a SQL View that queries the cached table rather than **tblFormData**. If the Form isn't configured to use Form Data Caching, or the cache doesn't exist, any attempt to create the view will fail immediately.

## Some Caveats

Form Data Caching may provide significant performance improvements, but the nature of your form data, and its distribution across forms, does play a large part in how much performance improvement you'll actually see. If you have a system with many Forms, and with the data more or less regularly distributed between them, then the performance improvement should be noticeable. On the other hand, if you have very few forms—especially if you have a single form that contains the majority of the data—the improvements will be much less noticeable.

When you are querying the cache, whether via a Knowledge View or SQL View, you should remember that the data you'll see is only as recent as when the cache was actually created. If you haven't created the cache for five days, then the most recent data will be five days old. As an implementer, when you create Knowledge Views that use cached data, you should consider letting your users know that the data may not be current. For instance, in the Knowledge View Definition, you may want to use the [HTML Code](#) property to display a message that informs the user that the data they are viewing is cached, rather than live.

Creating the cache queries [tblFormData](#) and creates a new temporary table for each form, so the cache creation process can impact performance while it's running, especially if you are creating a large number of caches at one time. So, you need to balance your need to have current data in each form cache with the need to create the cache during times of lower system use, to minimize the performance impact on the system.

## Visibility and Enabling Scenarios

Any time you create a Process Director Form, you should think carefully about when controls should be enabled or disabled, or when they should be shown or hidden. Most of the time, for instance, the majority of controls will need to be disabled after the initial user fills out the form. Or perhaps the form might contain controls that only managers can see, and which should be hidden from other users. In general, you should ask yourself some questions about visibility and enabling scenarios:

1. What fields should be edited by the initial user?
2. What fields should be edited at later stages of the process?
3. What fields should be shown or hidden based on the user?
4. What fields should be shown or hidden based on the process?

You should also remember how enabled and visibility settings are inherited in Process Director, which is that the setting at the lowest level wins. So, if a parent control, like a [Section](#), is disabled, but a control inside the [Section](#) is enabled, the control will be enabled. Similarly, if a parent control is visible, a child control can be hidden. One exception to the rule that the setting for the lowest control wins, however, is when a parent control is set as hidden. **If a parent control is hidden, Process Director won't paint either the parent or any child controls onto the form**, so it isn't possible to hide a parent control and show a child control, though the reverse can be done.

## When Controls Should be Edited

First, we can assume that most of the controls on a Form should be edited when the initial user fills out and submits the Form. The corollary to this is that, once a form has been submitted, most of the controls don't need to be edited. It follows then, that the default state for controls on the initial display to the first user should be enabled, while the default state for subsequent viewings of the Form should be to have controls disabled.

For the initial display of the Form, Process Director incorporates this assumption by providing a dropdown setting on the [Properties](#) tab of the Form definition that determines the default state for form fields. You can change the visible or enabled state of the controls at any later step in the process, based on whatever conditions you desire.

Instantiated Form Name  
{FORM\_DEF\_NAME} Submitted On {CREATE\_DATE}

Form Fields are Enabled Only When ... ▼ [\(New Form Instance? = Yes\)](#)

Entire form is read-only when [Click to create condition ...](#)

One of the settings in the dropdown is **Form Fields are Enabled Only when**, and this setting enables you to set a condition, which should almost always be "New Form Instance = Yes". This setting allows the initial user to fill out the form, but disables the fields for subsequent users. Because Process Director's inheritance model allows the lowest control setting to win, you can enable individual sections of the Form, so that users can fill them out when needed in the process, while the rest of the Form's controls are still disabled, preventing inadvertent editing of those controls.

By default, disabled Form control displays in Process Director as "grayed-out" controls, like the example below:

**Employee Section**

Employee Name: Diana Stuart

Department: Finance ▼

Position: AR Clerk

Another useful option in Process Director will change this display behavior to simply show the fields as text, without displaying the control itself. In the **Options** section of the **Properties** tab of the Form definition, you can check the **Show disabled fields as text** check box to enable this option.

Select a Content List background image for this form (optional)

Choose background

Cover ▼

Opacity Level 1-100 0

Select a background image from this list for this form (optional)

Form Lock Group (optional)

Display validation errors with alert boxes

Do not show this object in 'Items I Can Run' Knowledge Views

Use Visual Basic for scripts (leave unchecked for C#)

Audit Form field changes

Enable Form Locking

Display warning if user Cancels form changes

Show disabled fields as text

Show only the first validation error (instead of all)

Hide disabled buttons

Remove form from existing Case on submission

Enable Form Data Cache

When you check the **Show disabled fields as text** option, the data for disabled controls appears on the Form without the "grayed out" control being visible.

Employee Section
<p><b>Employee Name:</b> Diana Stuart</p> <p><b>Department:</b> Finance</p> <p><b>Position:</b> AR Clerk</p>

### Initial Display Conditions

When a form is first filled out prior to submission, it's quite possible that some fields are irrelevant, and don't need to be displayed. For instance, if there is a **Routing Slip**—and there should be—approver's comments, or fields that contain information that will be added at a future point in the process, they simply don't need to be displayed in new Form Instances. In fact, at every step in the process, fields that aren't relevant to that specific task shouldn't be displayed. After all, you want users to be focused on the assigned task, so there's no need to show them data that isn't relevant to that task.

This need to conditionally show or hide controls makes **Section** controls especially useful. The **Section** control enables you to group like controls together. By setting the visibility of a **Section** control essentially set the visibility for all the child controls, since, if a **Section** is hidden, all of the child controls are hidden as well. It's a good idea, then, to organize your form into sections that contain groups of controls that are relevant to the different portions of the process. As the process moves forward, you can then hide or show the relevant portions of the Form—or *all* of the Form—based on Activity results, new Form data, or any other evaluable condition.

### Sections with Approval Comments and Routing Slips

As mentioned previously, any approval comments or **Routing Slips** should be hidden on the initial display of the Form. Any approval comments and other approval-related controls should be placed in a **Section** control immediately above the Form buttons. Placing a **Routing Slip** inside a **Section** control is optional, but the **Routing Slip** should be placed below the Form buttons.

The screenshot shows a routing slip interface. At the top is a section titled 'Approver Comments' with a scrollable text area. Below this is a 'Button Area' containing 'OK' and 'Cancel' buttons. The main part of the form is a table with columns: Participants, Signature, Completed, Status, Result, and Comments. The table is organized into three approval stages: Supervisor Approval, VP Approval, and HR Review. Each stage has a header row with a date and time, followed by individual approver rows. Red callout boxes point to the 'Approver's Comments Section', the 'Button Area', and the 'Routing Slip' table.

Participants	Signature	Completed	Status	Result	Comments
<b>Supervisor Approval</b> 7/20/2010 9:39 AM					
Diana Stuart	<i>Diana Stuart</i>	7/20/2010	Completed	✓ Approve Request	looks fine
<b>VP Approval</b> 7/20/2010 9:39 AM					
Barb Stanley	<i>Barb Stanley</i>	7/20/2010	Completed	✓ Approve	ok
<b>HR Review</b> 7/20/2010 9:40 AM					
Rick Rob	<i>Rick Rob</i>	7/20/2010	Completed	Start Recruitment	done

To a certain extent, the **Routing Slip** control controls its own visibility. The **Routing Slip** is always hidden on the initial display of a Form, or when there are no **Routing Slip** entries to display, such as when a Form has been filled out and saved without submitting it.

### Setting Display Conditions

Setting display conditions is done on the **Form Controls** tab of the Form definition. Click on the **Edit** link for the control you wish to configure to open the **Field Properties** dialog box. Once the dialog box is open, set the enable/disable conditions using the **Set Readonly Options** dropdown, and visibility conditions using the **Set Display Options** dropdown.

#### Set Readonly Options

There are two selections in the **Set Readonly Options** dropdown that enable conditions for enabling the control.

OPTION	DESCRIPTION
Field is Enabled when...	Apply conditions that determine when the field is Enabled.
Field is Disabled when...	Apply conditions that determine when the field is Disabled.

In either case, selecting the option will automatically place a link to the **Condition Builder** on the dialog box, adjacent to the control, labeled "Click to create condition...".

Click on the [Condition Builder](#) link, [Click to create condition...](#), to create the condition that will enable the control. In addition, there is an [Otherwise Disabled](#) check box (or [Otherwise Enabled](#), depending on the option you choose). Checking this check box will override all other Readonly settings that might affect the control's Readonly state, such as the Readonly state of a parent control. Clicking this check box is usually not necessary, especially when you're applying the setting to a parent control like a [Section](#) control, but if there is a case where a settings conflict might occur, checking this box ensures that the control's Readonly setting always wins the conflict.

### *Set Display Options*

Like the Readonly options, there are two selections in the [Set Display Options](#) dropdown that enable conditions for the control's visibility.

OPTION	DESCRIPTION
Field is Visible when...	Apply conditions that determine when the field is Enabled.
Field is Hidden when...	Apply conditions that determine when the field is Disabled.

Again, selecting either option will automatically place a link to the [Condition Builder](#) on the dialog box, adjacent to the control, and you can click it to create the condition that will show or hide the control. Similarly, there is an [Otherwise Hidden](#) or [Otherwise Visible](#) check box to override other visibility settings that might conflict with the desired visibility condition for the control.

### *Consistency in Condition-Setting*

Note that both the disabling and display options provide two condition settings apiece. For instance, you can **show** a control when a condition applies, or you can **hide** a control when a condition applies. You may find it helpful to consistently use the same choice for every control that's part of your visibility/enabling scenario. If you start off using "Hide this control when", when configuring visibility scenarios, then you should use that setting for all controls you want to hide, when possible.

Mixing and matching show/hide or enable/disable conditions makes the form's configuration more confusing, and can often lead to unexpected clashes between conditions, thus unexpected Form behaviors. Be consistent in your method of applying visibility and enabling scenarios.

## Collaborative Document Markup

For users of Process Director v5.13 and higher, document attachments may be annotated using Collaborative Document Markup. This feature creates a PDF markup copy of the original document, which stores both the original document text and formatting, as well as the markup annotations placed into it. The original document is not edited directly. When the markup document is opened, it displays all previous markups from all users.

Multiple users can add markup comments and annotations to a document simultaneously, though the simultaneous changes made by other users are not displayed in real time. Instead, each simultaneous user works in a unique session while annotating the document, and see only the new markups they add during their session. Once the user saves and closes the markup document, the markups made during their session will be recorded and added to the existing markup annotations. Once you reopen the markup document, it will display the markups saved by all users during your editing session.

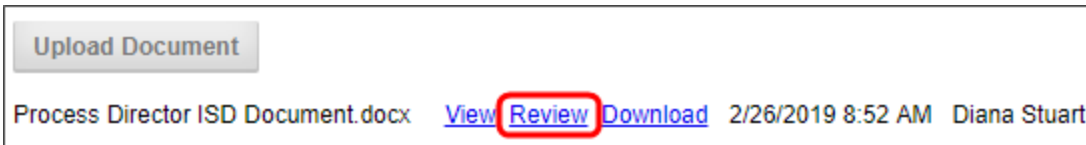


**i** Collaborative Document Markup is a separately licensed component that may not be available for all installations of Process Director. This feature also requires that the Process Director server has the 64-bit version of the Microsoft Visual C++ runtime library installed. The library can be obtained from [Microsoft's web site](#).

Please see the documentation for the [ShowAttach control](#) to see how to enable Collaborative Document Markup for attachments. Assuming that [Review Permissions](#) are enabled, Collaborative Document Markup will enable the user to add comments, callouts, and other annotations for the following document types:

- DOC/DOCX
- XLS/XLSX
- PPT/PPTX
- TXT/CSV
- PDF

Once enabled, when a user clicks the Review link for the attachment, the attachment will be opened in a new window for the Collaborative Document Markup interface for display and/or editing.



While the document itself can't be edited directly, a number of marking, annotation, and commenting features can be used and saved with the document.

Process Director ISD Document.docx Save Close

44. COGNITIVE REQUIREMENTS

45. Explain the concept of using Process Timelines to manage process performance

**DEFINE EDUCATION AND TRAINING REQUIREMENTS**

Students will be trained to perform specific tasks at each expertise level. The Core Proficiency level will require competency in creating the basic Process Director objects. The Advanced Proficiency level ....

**CORE PROFICIENCY**


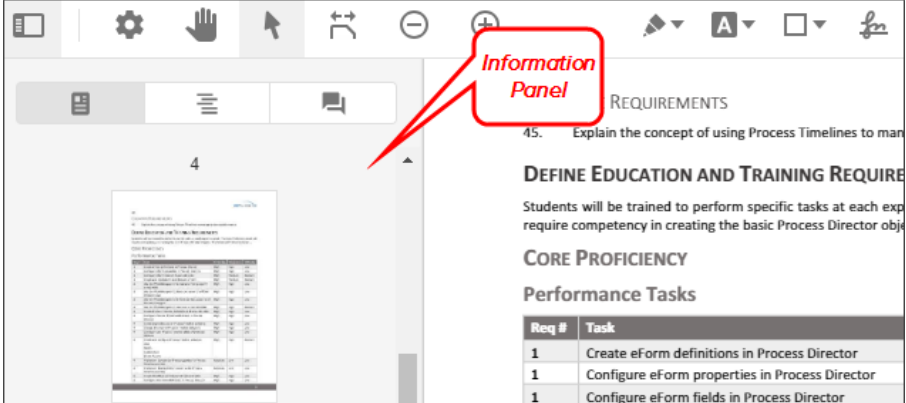


**Performance Tasks**

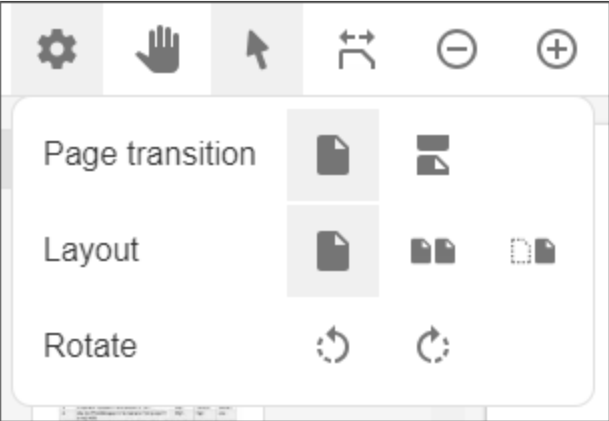






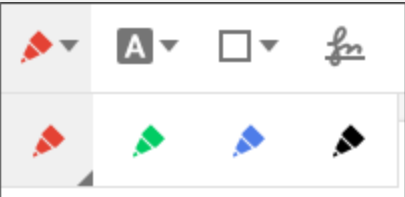

Req #	Task	Criticality	Frequency	Difficulty
1	Create eForm definitions in Process Director	High	High	Low
1	Configure eForm properties in Process Director	High	High	Low
1	Configure eForm fields in Process Director	High	Medium	Medium
1	Create and implement conditions in eForms	High	Medium	Medium
2	Use the Word Designer to Format an eForm properly using tables	High	High	Low
2	Use the Word Designer to place controls on the Word Designer page	High	High	Low
2	Use the Word Designer to Set form control properties in the word designer.	High	High	Low
2	Use the Word Designer to create an eForm template	High	High	Medium
3	Create Process Timeline definitions in Process Director	High	High	Low

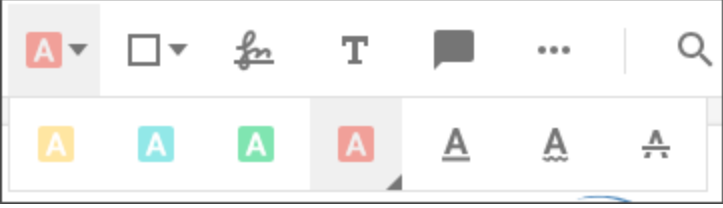



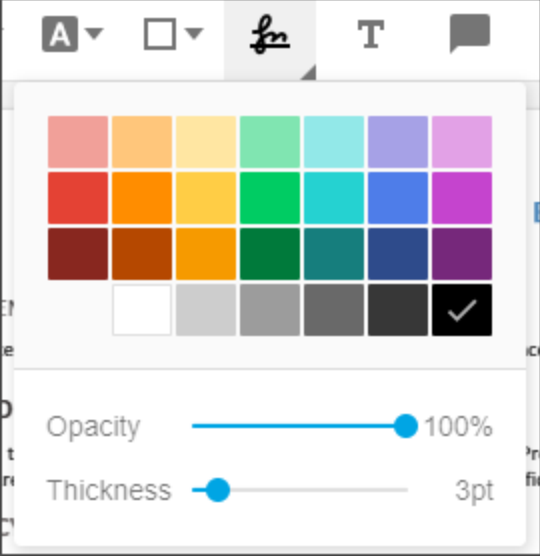



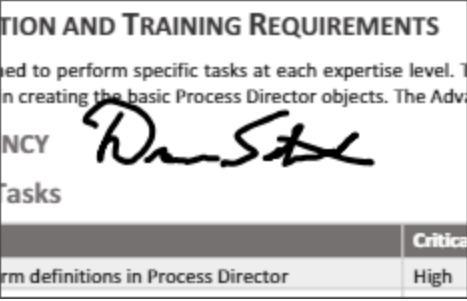

## Markup Tools #


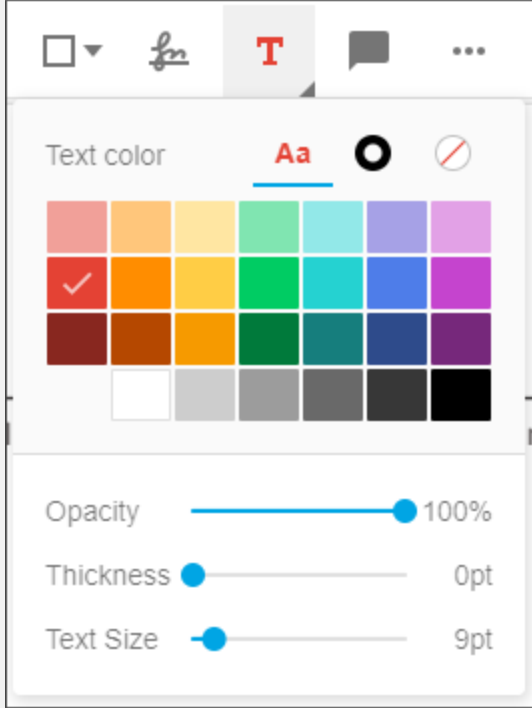
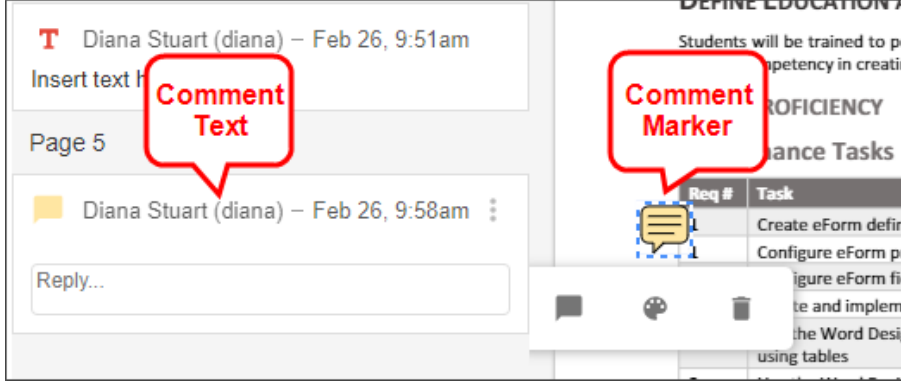
A number of tools are available to annotate the document, and these tools are displayed in the toolbar at the top of the page.

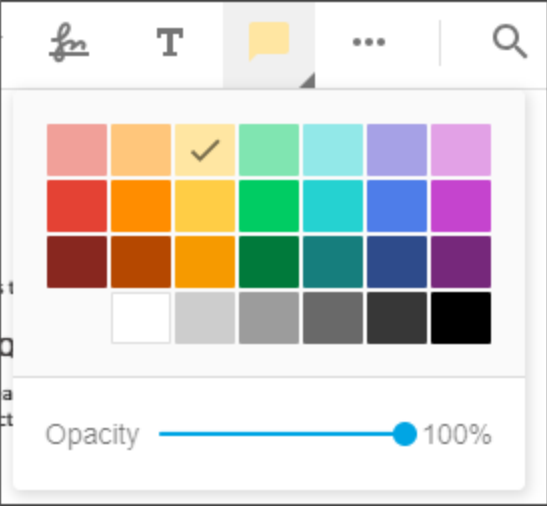

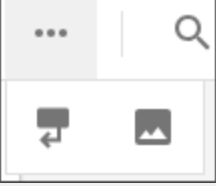

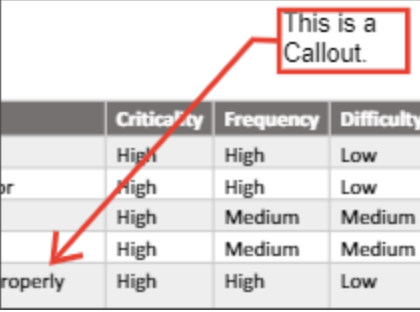
TOOL	NAME	DESCRIPTION
	Panel	<p>The panel tool shows or hides the information panel, which will appear on the left side of the document window.</p>  <p>The information panel has three different display modes, which can be selected from the buttons at the top of the panel.</p>  <ul style="list-style-type: none"> <li>• <b>Thumbnails</b> will display thumbnail images of each page of the document, which is useful for navigating quickly through long documents. In the example shown above, Thumbnails is the selected mode.</li> <li>• <b>Outlines</b> will display the PDF outline of the document, if applicable, which is, again, useful for navigating large documents.</li> <li>• <b>Annotations</b> will display any annotations that have been added to the document.</li> </ul>
	View Controls	<p>Clicking this tool displays a dropdown menu with some tools to determine how you'd like the document to display.</p>

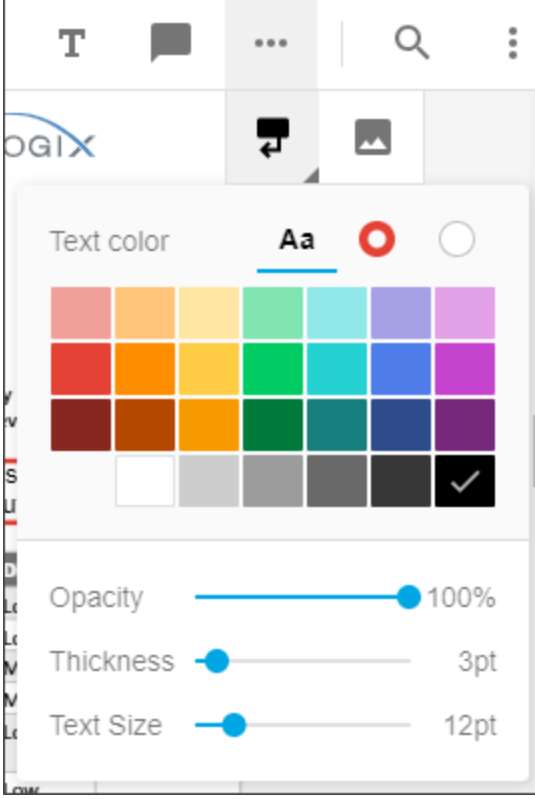

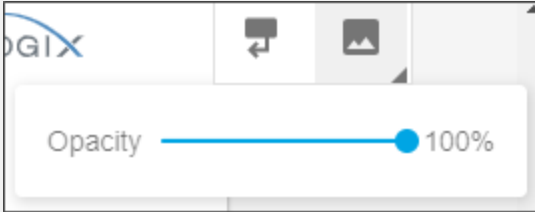

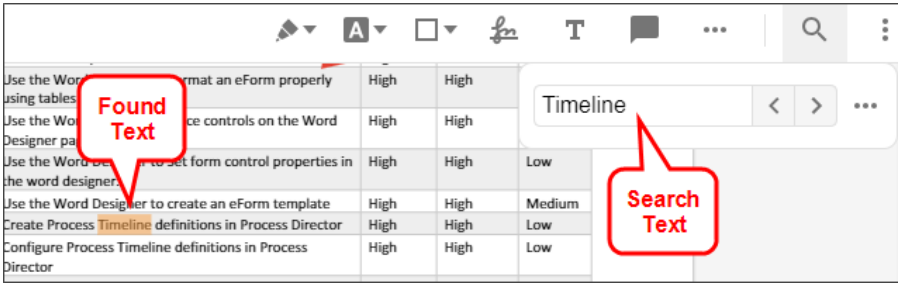
TOOL	NAME	DESCRIPTION
		 <p><b>Page Transition</b> enables you to select how to display the page transition for multi-page documents.</p> <p><b>Layout</b> enables you to select the reading layout for the document.</p> <p><b>Rotate</b> enables you to rotate the document clockwise or counter-clockwise.</p>
	Pan	Enables you to use the mouse to click and drag the displayed document when the view is zoomed in.
	Select	Switches to the standard mouse cursor, turning off all Markup controls.
	Fit	Changes the page display size to switch between fit to width or fit to page.
	Zoom Out	Zooms the page display out.
	Zoom In	Zooms the page display in.
	Freehand	<p>Enables you to choose a pen color to make freehand ink markings with a freehand pen, using the mouse.</p> 
	Text Tools	Enables you to select a number of text marking tools to highlight, underline, or strikeout text in the document.


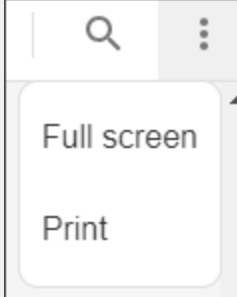
TOOL	NAME	DESCRIPTION
		
	Shape Tools	<p>Enables you to add different drawing shapes, lines, or arrows to the document.</p> 
	Signature	<p>Enables you to use a signature control that you can use to sign with the mouse to place the signature on the document. Clicking the lower right corner of the tool also opens a pen menu from which you can choose the pen color, opacity, and thickness.</p>  <p>Once you select the tool and its settings, simply click on the document where you'd like the signature placed. A signature control window will open that enables you to use your mouse to add a signature.</p>

TOOL	NAME	DESCRIPTION
		 <p>If you're dissatisfied with the signature, and want to start over, clear the signature by clicking on the <b>Delete</b> tool. When you're done, click the <b>Sign</b> tool to place the signature on the document.</p> 
	Free Text	<p>This tool enables you to place a free-standing text box on the document. Clicking the lower right corner of the tool also opens a menu from which you can choose the properties of the text box.</p>

TOOL	NAME	DESCRIPTION
	<p>Comment</p>	
		<p>Enables you place a comment annotation on the document. Once this tool is selected clicking anywhere on the document will place a comment marker at that location. You can add the text for the comment in the Comment box that will appear in the information panel.</p>  <p>Clicking the lower right corner of the tool also opens a menu from which you can choose the properties of the comment.</p>

TOOL	NAME	DESCRIPTION																								
																										
	Misc Tools	<p>Opens a submenu to display the Callout and Stamp tools.</p> 																								
	Callout	<p>Enables you to draw a callout box on the document.</p>  <table border="1" data-bbox="435 1207 852 1411"> <thead> <tr> <th></th> <th>Criticality</th> <th>Frequency</th> <th>Difficulty</th> </tr> </thead> <tbody> <tr> <td></td> <td>High</td> <td>High</td> <td>Low</td> </tr> <tr> <td>or</td> <td>High</td> <td>High</td> <td>Low</td> </tr> <tr> <td></td> <td>High</td> <td>Medium</td> <td>Medium</td> </tr> <tr> <td></td> <td>High</td> <td>Medium</td> <td>Medium</td> </tr> <tr> <td>properly</td> <td>High</td> <td>High</td> <td>Low</td> </tr> </tbody> </table> <p>Clicking the lower right corner of the tool also opens a menu from which you can choose the properties of the callout box.</p>		Criticality	Frequency	Difficulty		High	High	Low	or	High	High	Low		High	Medium	Medium		High	Medium	Medium	properly	High	High	Low
	Criticality	Frequency	Difficulty																							
	High	High	Low																							
or	High	High	Low																							
	High	Medium	Medium																							
	High	Medium	Medium																							
properly	High	High	Low																							

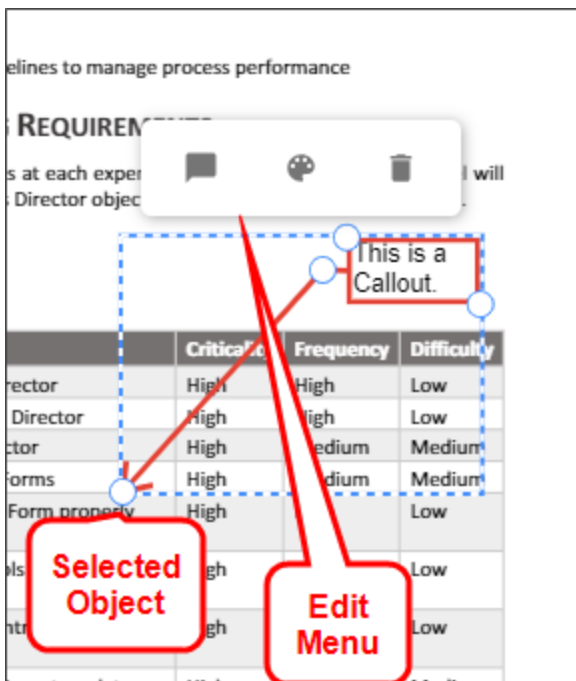
TOOL	NAME	DESCRIPTION
		
	Stamp	<p>Enables you to select an image to place on the document. Clicking on the document opens the Browse dialog box, from which you can navigate to and select the desired image. Clicking the lower right corner of the tool also opens a menu from which you can choose the opacity of the image.</p> 
	Search	<p>Enables you to search the document's text.</p> 

TOOL	NAME	DESCRIPTION
	Menu	Opens a menu to enable you to display in full-screen mode, or to print the document. 

### Using the Markup Tools #




To use any of the Markup tools, simply click on the tool you want to use. Many tools also have submenus where you can change specific properties of the tool, such as color, thickness, etc., as described above. Once you select the desired tool, the mouse cursor will show as a set of crosshairs to use in determining where to place the object. Simply, click on the place in the document where you'd like to place the tool. The appropriate object will be placed in the location where you click. You can then add text, or draw the desired item, as required.

Once an object has been placed in the document, you can go back to edit or delete the object by clicking on the **Select** tool to activate it., then clicking the object. When you click on the object using the **Select** tool, the object will become highlighted for editing, and a small edit menu will appear.



The edit menu provides three options for editing.



TOOL	NAME	DESCRIPTION
	Comment	Add a comment to the object in the Information Pane.
	Style	Open the style menu to change the style of the object.
	Delete	Delete the object.

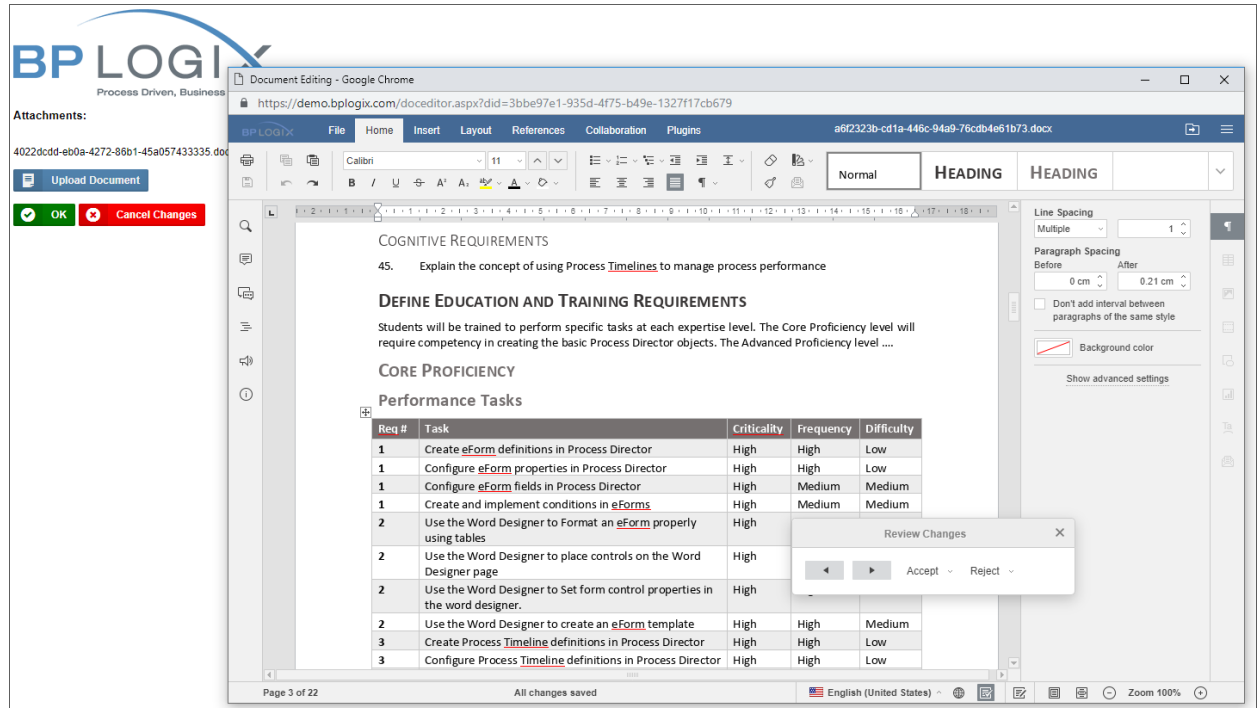
Text Markups can be edited directly by double-clicking on them, then editing the text.

Once you are done reviewing the document, click the **Save** button to save your changes. If you are making a lot of Markups, you may want to save periodically to ensure that you don't lose your changes should the browser window be inadvertently closed. To close without saving your changes, click the **Close** button, then confirm that you want to close without saving in the confirmation dialog box.

## Collaborative Document Authoring

For users of Process Director v5.13 and higher, document attachments may be annotated using Collaborative Document Authoring.

 Collaborative Document Authoring is a separately licensed component that is only available for Cloud installations.



Please see the documentation for the [ShowAttach\\_control](#) to see how to enable Collaborative Document Authoring for attachments.

Collaborative Document Authoring is a cloud-based service, OnlyOffice, that opens a number of file formats in an online editor for collaborative authoring and editing, change tracking, etc. The documentation for the Document, Spreadsheet and Presentation editors is available at [the OnlyOffice documentation web site](#).

The following file formats can be authored in the OnlyOffice editors:

- Document Editor: DOCX, TXT, ODT, or RTF files.
- Spreadsheet Editor: XLSX, ODS, or CSV files.
- Presentation Editor: PPTX or ODP files.

## Form Configuration

Form definitions are configured via a tabbed interface, like all other Process Director objects. Each interface tab enables you to configure different properties. You can view the documentation for each of the configuration tabs by using the Table of Contents displayed on the upper right corner of the page, or the links below.

**[Properties](#)**: General Form definition properties.

**[Edit](#)**: Opens the editor for the form's visual template, the [Online Form Designer](#).

**[Form Controls](#)**: Enables you to set individual properties for each field on the form.

**[Custom Task Event Mapping](#)**: Enables you to map Custom Tasks to run on specified form events.

**[Validation Rules](#)**: Enables you to add custom validation to the form.

**[Set Form Data](#)**: Enables you to set the value of Form fields automatically, based on events or conditions you specify.

**[Fill Dropdowns](#)**: Enables you to automatically set the options available in a dropdown control.

**[Other Tabs](#)**: Documentation for the [Form Data SQL View](#), [History](#), [Meta Data](#), [Versions](#), and [Permissions](#) tabs.

## Properties Tab

Form Name Icon

Documentation

---

### PROPERTIES

---

**Description**

Enter a brief description of this Object

---

**Instantiated Form Name**

{FORM\_DEF\_NAME} Submitted On {CREATE\_DATE}

---

Form Fields are Enabled by Default ▼

---

**Entire form is read-only when**

▲ **Options**

Automatically start this process after Form is submitted (optional)		<span>...</span>
Workflows Associated with Form (optional)		<span>...</span>
Timelines Associated with Form (optional)		<span>...</span>
Default Groupname for Form Instances		
Form Script (optional)		<span>...</span>
Automatically create an instance of this Case (optional)		<span>...</span>
Select a Content List background image for this form (optional)		<span>...</span>
Select a background image from this list for this form (optional)	Choose background ▼ Cover ▼ Opacity Level 1-100 0	
Form Lock Group (optional)		

<input type="checkbox"/> Display validation errors with alert boxes <input type="checkbox"/> Do not show this object in 'Items I Can Run' Knowledge Views <input type="checkbox"/> Use Visual Basic for scripts (leave unchecked for C#) <input type="checkbox"/> Audit Form field changes <input type="checkbox"/> Enable Form Locking <input type="checkbox"/> Disable automatic Friendly Name generation for controls	<input type="checkbox"/> Display warning if user Cancels form changes <input type="checkbox"/> Show disabled fields as text <input type="checkbox"/> Show only the first validation error (instead of all) <input type="checkbox"/> Hide disabled buttons <input type="checkbox"/> Remove form from existing Case on submission <input type="checkbox"/> Enable Form Data Cache
---	--

The **Properties** tab enables you to change the default behavior of the form. We'll discuss all of the existing properties, but keep in mind that some properties are only visible to installations that have the appropriate licensing. If your license doesn't cover a property, such as the auditing or scripting properties, you won't see them in the product UI.

The options on this tab include the ability to change default styles and how to handle validation errors. Associating processes with Forms allows the Form to reference steps and other information in that process. The **Default Styles** control the stylized borders and colors around Form controls, depending on the controls' status (enabled, required, disabled, or error).

### Instantiated Form Name

The **Instantiated Form Name** property determines the name that will be given to each Form instance. One should usually use System Variables here, so that the Form's instance's name is relevant to the specific instance. As a best practice, every form should be given an **Instantiated Form Name** that is unique to that instance, for easier searching in Knowledge Views.

### Form Fields Enabled/Disabled

You have the option to determine the default setting for whether form fields are enabled or disabled, using this dropdown control. The control offers four settings:

- **Form Fields are enabled by default:** All form fields are enabled by default.
- **Form Fields are disabled by default:** All form fields are disabled by default.
- **Form Fields are enabled only when...:** All form fields are enabled only when a user-defined condition applies. Selecting this option enables you to define a condition or set of conditions that, when true, enable the form fields.
- **Form Fields are disabled only when...:** All form fields are disabled only when a user-defined condition applies. Selecting this option enables you to define a condition or set of conditions that, when true, disable the form fields.

These form level-settings can be overridden in the properties for individual controls. For example, if you select "Form Fields are disabled by default", but set an individual control to be enabled, the control setting will override the Form option setting you choose here. The general rule to remember when setting an enabling option is that the setting at the lowest level always wins.

### Entire form is read-only when

Form definitions can be configured as "read-only". Selecting this option enables you to define a condition or set of conditions that, when true, will cause the form will act as though the user doesn't have modify permission on the form (whether or not that would otherwise be the case). An open form in "read-only" state won't trigger form locking. Unlike the option to enable form fields, this option overrides any "enabled" settings at the control level. The data contained in a read-only form can't be edited, irrespective of the enabled/disabled settings that might be set elsewhere.

### Options Section #

This section of the tab contains the general properties of the form. This section enables you to specify the Process Timeline to invoke when the form is submitted, and to associate different objects to the form to

make configuring conditions based on those objects easier. Additionally, this section contains properties that enable you to configure the overall operation of the form, and specify general form behaviors.

### **Automatically start this process after Form is submitted**

The primary method for starting a process in Process Director is to call the process from a Form when the Form is submitted. This option defines the process that will be started, and the **Object Picker** control allows you to choose a Process Timeline. Once you've selected the process you desire, then every time a new instance of the Form is submitted, the selected process will start, and the Form and its data will be attached to that process.

BP Logix recommends that you use a Process Timeline as the process to start after the form is submitted.

In most cases, you'll also want to set the **Timelines Associated With Form** property to the same process you set here, so that any Form conditions you create that evaluate process data will use this process by default. Setting the association will make configuring Timeline-based conditions much easier and quicker.

### **Workflows Associated with Form (optional) / Timelines Associated with Form (optional)**

In many cases, you'll need to create conditions that evaluate process data to control the operation of a form. For example, you might wish to show a section of the form only when a specific Timeline Activity is running. To do so, you need to create a Form condition that returns true or false, based on whether the specified Activity is currently running. When you associate a process via this property, that process will become the default process to use when configuring process-based conditions. If you don't set this property, then you'll have to specify the process every time you configure a condition that evaluates the process. Setting this property, therefore, makes configuring process-based conditions much simpler, and quicker.

Setting the **Automatically start this process...** property doesn't create this association. In most cases, you should set both this property and the **Automatically start this process...** property to the same Process Timeline.

### **Default Groupname for Form Instances**

You can set a custom Group Name for each instance of the Form. This enables you to group Forms by department, process type, etc. When a process is started that automatically adds a Form, it will use the **Default Groupname...** property configured on the Form definition.

### **Form Script (optional)**

You may associate a custom script with a Form to perform specialized, programmatic functions. This feature requires the SDK license.

### **Automatically create an instance of this case (optional)**

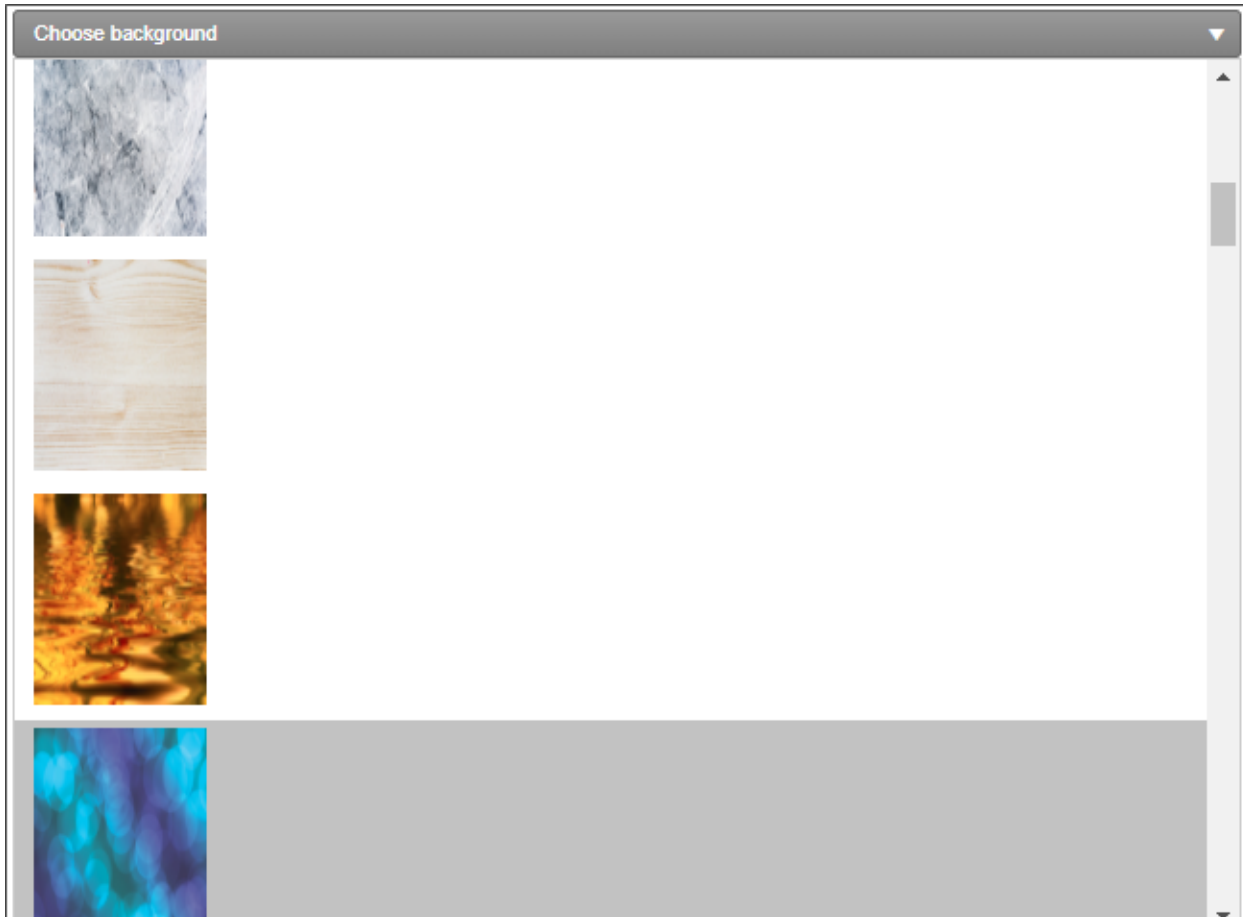
For [Case Management applications](#), this option enables you to select the Case Definition for which a new instance will be created when the Form is submitted. When a Case definition is selected, the Form won't only create the Case instance, but will also set some or all of the Case properties from controls on the Form.

### **Select a Content List background image for this form (optional)**

This option enables you to select an image you've uploaded to the [Content List](#), and will use that image as a background image for the Form.

### Select a background image from this list for this form (optional)

This option enables you to select an image from a dropdown control that contains a number of stock background images, and will use that image as a background image for the Form.



Below the image dropdown, an additional Background Image Sizing dropdown enables you to specify how the background image will appear on the form. The following options are available:

- Cover: The image will expand to fill the entire page background.
- Center: The image will appear in its actual size, centered on the page vertically and horizontally.
- Tile: The image will appear in its actual size, aligned to the top left of the window, and will tile vertically and horizontally.
- Tile X: The image will appear in its actual size, aligned to the top left of the window, and will tile along the X-axis only.
- Tile Y: The image will appear in its actual size, aligned to the top left of the window, and will tile along the Y-axis only.

Finally, an [Opacity Level](#) property enables you to set the opacity percentage by changing a slider value from 0 to 100.

Select a Content List background image for this form (optional)

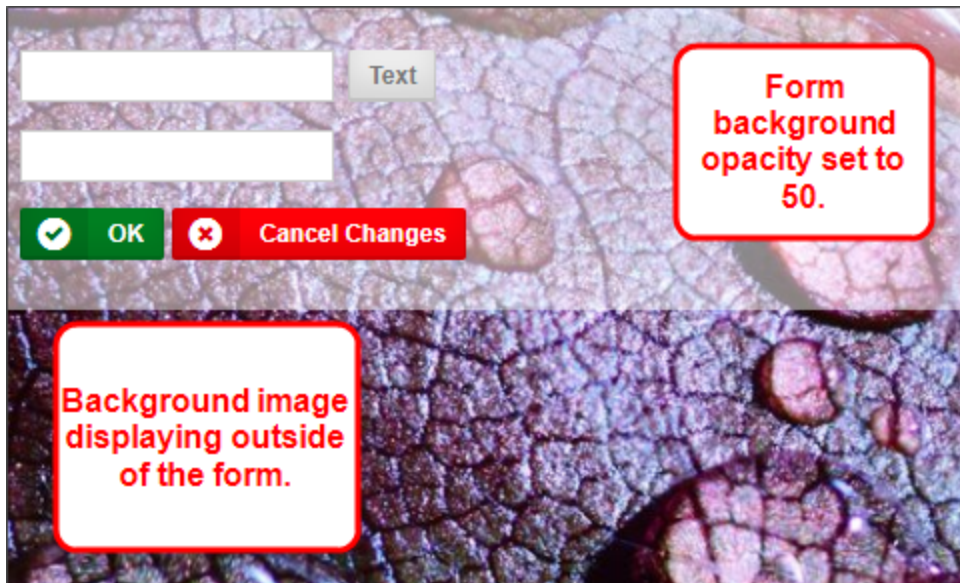
Select a background image from this list for this form (optional)

abstract-art-artistic.jpg

Cover

Opacity Level 1-100 10

This slider will set the opacity of the form, not the background image. The form background displays above the page background that contains the image, so, the higher the opacity level, the more opaque the form background becomes, and the more the background image is obscured. In the example below, the opacity level is set to 50, i.e., 50% opaque.



### Form Lock Group (Optional)

This is an advanced locking feature. Entering a value in this box will create a form lock group for the form, so that multiple locks can exist at once on a form instance. This feature is an expansion of the existing Form Locking functionality. Rather than locking the form any time a user is viewing it, the Lock Group allows the administrator to add additional conditions that will lock the form only when those conditions are met. This enables concurrent tasks or users assigned in parallel to a task to access the form simultaneously without locking it. This does have the inherent risk of one user saving over another user's changes if multiple people have it open and don't meet the conditions of the Form Lock Group, similar to the existing risks when locking isn't being used.

So, let's look at an example. Let's say that we set the optional Form Lock Group to `{TASK_USER, FORMAT=T=user-id}`. In that case, the form will be locked only when a user *in the same task* accesses the form. Users who are in other tasks would still be able to edit the form, because the lock group wouldn't apply to them.

### Display validation errors with alert boxes



The default method of displaying validation errors on a Form is to display a text message at the top and bottom of the Form, which will appear as a yellow error banner. Selecting this option also causes the validation error to appear in an alert dialog box.

### Do not show this object in 'Items I Can Run' Knowledge Views

Selecting this option removes the Form from the [Items I can Run](#) Global Knowledge View.

### Use Visual Basic for scripts (leave unchecked for C#)

The default programming language for scripts in Process Director is the C# programming language. Selecting this option enables the use of Visual Basic.NET as a scripting language for this Form.

### Audit Form field changes

Users of Cloud Installations, on-premise installation with the Subscription license, or on-premise installations with the Compliance Option, have the ability to maintain a record of every change made to Form fields during the process. Selecting this option enables field-level auditing for the Form. This auditing information can be reviewed in the properties of the Form Instance, by viewing the [Audit Viewer tab](#). Fully-detailed auditing information can also be accessed in the Audit logs, or, if configured for use, the [IT Admin](#) area's [Audit Logs page](#).

### Enable Form Locking

Form locking prevents other users from making changes to a Form instance once a user opens it. This prevents multiple users from overwriting each other's changes. Selecting this option enables form locking.

### Disable automatic Friendly Name generation for controls

For Process Director v6.0 and higher, this property enables you to turn off the default behavior that automatically generates Friendly Names for Form controls. When selected, all Friendly Names will have to be supplied manually for each control.

### Display warning if user Cancels form changes

Selecting this option will display a warning before closing the Form when users cancel their changes.

### Show disabled fields as text

Selecting this option will display only the text in disabled fields, rather than displaying the actual control. The value in the control will be output to the Form as simple text, as if the control did not exist.



BP Logix recommends that you use this setting.

### Show only the first validation error (instead of all)

In a Form contains multiple validation errors (i.e., field values that don't meet the criteria for acceptable data), only the first validation error will be shown when the validation fails if this option is checked.

### Hide disabled buttons

Selecting this option will keep disabled buttons from appearing on the Form when it is displayed.



## Remove form from existing Case on submission

This option tells Process Director to remove a new form instance from an existing case instance during submission if it is already in a case instance.

When a new Form instance is created from inside an existing case, the default behavior is to add the new Form to that case. Checking this option changes the default behavior so that the new form Instance is automatically transferred to a *new case instance*. Any case properties from the existing case instance that have been applied to form fields will remain applied. This enables the new form to be displayed with the case properties from the existing case instance, but when the form instance is submitted, it will only set properties in a *new case instance* that is created on submission, without altering the properties of the existing case instance.

## Enable Form Data Cache

This option implements form data caching for the form. For an explanation of what Form Data Caching is, and how it works, please see the [Form Data Caching](#) topic.

## URL Customization

For users of Process Director v5.13 and higher, this section enables you to customize the Form URL to use to access the form. This section assists you with constructing the URL to use for users who access the form **via URL only**. *It doesn't change the operation of the form when accessed through the Process Director UI.* You can then copy the **Form URL** and paste it into a hyperlink from an HTML document, such as an intranet portal page, outside of Process Director. This section of the form **only** changes the **Form URL** that is displayed. It has no other effect of the form's operation. The sole purpose of this section is to edit the URL so that you can copy and paste it for use outside of Process Director. Changing the options in this section will, when the Form definition is updated, update the **Form URL** displayed at the bottom of the tab.

**Form Complete Options**

Do Not Show Home Page  Skip Complete Page

Complete Page URL

Complete Page Text

The properties available in this section correspond to some of the commonly used [Form Definition URL](#) parameters documented in the Forms topic, and which are included in the default **Form URL** displayed at the bottom of the tab.

- **Do Not Show Home Page:** Controls whether the home page should be opened when the link is clicked.
- **Skip Complete Page:** Controls whether the complete page should be displayed when the form is submitted.
- **Complete Page URL:** Specifies the URL of the page that should serve as the complete page when the form is submitted.
- **Complete Page Text:** Specifies a text message that should be displayed when the form is submitted, rather than redirecting to a complete page.

## Default Styles Section #

Default Styles

Enabled  Disabled

Required  Error

Additional Body CSS Styling

Enter additional CSS to include on your form.

This section enables you to quickly change the default styling of various fields on the Form, based on the status of the field. For each field status, you can choose a default style from the dropdown, or type in any desired CSS styles manually. You can alter the styling for the following field status types:

- Enabled
- Disabled
- Required
- Error

Additionally, the Additional Form Body CSS Styling property will accept CSS Stylesheet Class syntax that you can apply to any element on the Form. You would simply enter the CSS styling in the same way you'd in a CSS stylesheet, e.g.:

```
body
{
  color: #FF0000;
  font-family: Arial, Geneva, Helvetica, sans-serif;
}
```

For Process Director v5.31 and higher, a custom variable, [EnableFormFieldDownload](#), will display a link at the bottom of this tab labeled Download Field List, to enable users to download an Excel Spreadsheet containing the list of fields. This feature is primarily relevant to the use of the separately-licensed [Mobile Application Component](#).

### More Configuration Tabs

You can view the documentation for each of the other configuration tabs by using the Table of Contents displayed on the upper right corner of the page, or use one of the links below.

**Edit:** Opens the editor for the form's visual template, the Online Form Designer.

**Form Controls:** Enables you to set individual properties for each field on the form.

**Custom Task Event Mapping:** Enables you to map Custom Tasks to run on specified form events.

**Validation Rules:** Enables you to add custom validation to the form.

**Set Form Data:** Enables you to set the value of Form fields automatically, based on events or conditions you specify.

**Fill Dropdowns:** Enables you to automatically set the options available in a dropdown control.

[Other Tabs](#): Documentation for the [Form Data SQL View](#), [History](#), [Meta Data](#), [Versions](#), and [Permissions](#) tabs.

## Edit Tab

Clicking on the [Edit](#) tab will, depending on the format of the form template, open one of two different screens. Form templates created in ASCX format will present the following screen. Use of this screen is documented in the [Creating ASP.NET Forms](#) topic.

Please note that Forms created as a custom ASCX control can't be converted to an Online Form Designer Form, and the [Convert to an HTML Form](#) button will *not* be displayed in the [Edit](#) tab.

Forms created with the Online Form Designer will automatically open the Online Form Designer's editing screen, which is [documented in its own topic](#).

**i** Forms created using the legacy Word Form Builder will display a similar screen that also provides an option to convert the Form from Word to the Online Form designer, via a [Convert to an HTML Form](#) button. Clicking this button will create a new Form version for the Word Form prior to creating the Online Form Designer version. You can, if you desire, revert to the Word Form from the [Versions](#) tab.

**!** BP Logix strongly recommends that Forms built using the legacy Word Form builder be converted to the Online Form Designer format, as the technologies that support the Word Form Builder were eliminated by Microsoft on 15 June 2022.

## More Configuration Tabs

You can view the documentation for each of the other configuration tabs by using the Table of Contents displayed on the upper right corner of the page, or use one of the links below.

[Properties](#): General Form definition properties.

[Form Controls](#): Enables you to set individual properties for each field on the form.

[Custom Task Event Mapping](#): Enables you to map Custom Tasks to run on specified form events.

[Validation Rules](#): Enables you to add custom validation to the form.

**Set Form Data:** Enables you to set the value of Form fields automatically, based on events or conditions you specify.

**Fill Dropdowns:** Enables you to automatically set the options available in a dropdown control.

**Other Tabs:** Documentation for the [Form Data SQL View](#), [History](#), [Meta Data](#), [Versions](#), and [Permissions](#) tabs.

## Form Controls Tab

This tab displays a list of all the controls (i.e. form fields) on the Form definition. When you add a control to the Form Template, the corresponding Form data field will automatically be created, and the control will appear in the list of controls on the **Form Controls** tab after the Form Template is checked in.

Control Name	Control Type	Properties	Commands
ApprovalSection	Section		<a href="#">Edit</a>
ApproverComments	Workflow Signature Comments		<a href="#">Edit</a>
ApproverCommentsLabel	Label		<a href="#">Edit</a>
Buttonarea1	Button Area		<a href="#">Edit</a>
Buttonarea2	Button Area		<a href="#">Edit</a>
CommentLog	Activity Journal		<a href="#">Edit</a>
ComplaintSection	Section		<a href="#">Edit</a>
DiscussionSection	Section		<a href="#">Edit</a>
EndDate	Date		<a href="#">Edit</a>
EndDateLabel	Label		<a href="#">Edit</a>
Justification	Text Editor		<a href="#">Edit</a>

Properties can be configured for these form controls (fields) by clicking on the **Edit** link, which will open the **Form Field Properties** dialog box for the field.

### **Form Field Properties #**

For Process Director v6.0 and higher, the Form field properties can also be configured in the **Design Console**, a persistent **Properties** pane that appears on the **Edit** tab, when the Online Form Designer is displayed. The Field properties can thus be configured on both the **Edit** tab, as well as the **Form Controls** tab of the Form definition. In the Online Form Designer, the Form field properties will appear on the **Field Properties** tab of the designer's persistent **Properties** pane, as shown below. This feature enables you to set the properties without having to navigate to the **Form Controls** tab, simplifying the process of fully configuring a control's properties. This feature doesn't necessarily replace the properties accessible from the **Form Controls** tab, but does give you an additional method for accessing the same properties from within the Form designer.

For all versions of Process Director prior to v6.0, Form field properties must be configured on the **Form Controls** tab of the Form definition.

On the **Form Controls** tab, the **Form Field Properties** dialog box for each form control can be opened by clicking the **Edit** link for each control. The dialog box enables you to configure the properties of each form field. You can configure the form field's tooltip, its friendly name, the minimum and maximum values it can hold, as well as under what conditions the form field will be enabled, viewable, or required. You can also configure the form field's default value, data type, and attribute to which it links. Different form field types may have some subset of the properties below, as not all field types have the full set of properties. In general, however, the full set of form field properties are listed below.

**Form Field Properties: Keywords**

Event Field     Synchronize Field Within Process     Encrypt

ToolTip

Friendly Name

Data Type  ▼    Min     Max

Default Value  ▼

Link to Attribute  ...

Case Property  ▼

▼

▼

▼

▼

The form field properties include:

### Event Field

If checked off, this form field will trigger an event when interacted with. Events can be used to update the form by triggering Custom Tasks or custom form scripts.

### Synchronize Field Within Process

If checked, this field will synchronize values with all other form fields that have the same names but are on different forms with different definitions in the same Process. When the value of a synced form field is updated, the value will be synchronized with all other form fields with the same names in the Process, whether or not those form fields are also marked as synchronized. However, if a form field's value is updated and that form field isn't marked as a synchronized field, it won't update the values of other fields with the same names in the process, even if those fields are marked as synchronized.

If a column in an array is marked as synchronized, the number of rows and the content of that column will be synced with all other arrays in the process containing a column of the same name. This means that adding and removing rows in an array with a synced column will also add and remove rows in other arrays in the process with columns of the same name. Because synchronized array fields are so powerful, caution

is recommended when using synchronized fields with arrays. Instead, we recommend alternatives, like the Copy Form Data Custom Task.

Prior to Process Director v4.04, field synchronization would only occur on form submission, but not when the Form was saved via a **Save** button, and not submitted. From v4.04, synchronization will occur any time the Form is saved, even if the Form isn't submitted.

### ToolTip

When moused over, the form field will display the text entered into this text box as a tooltip.

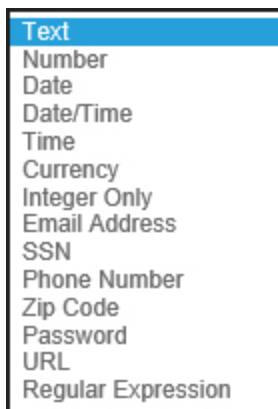
### Friendly Name

Because Form Field Names have some unavoidable constraints in how they are named, the field names may not be particularly friendly for users. The **Friendly Name** property enables you to configure a different name that will be displayed to end users in places like Knowledge Views, without worrying about the constraints that you have to adhere to when you place a field on the form template. For users of Process Director v5.12 or higher, using a Camel Case value (e.g., MyField) for the field's **Name** will automatically create a **Friendly Name** based on the Camel Case **Name** (e.g., My Field).

### Link to Dropdown Object

This enables you to fill a Dropdown field with the content from a [Dropdown Object](#). You can link multiple dropdowns to Dropdown Objects, which allows you to reuse Dropdown Objects in multiple forms.

### Data Type



Setting the data type will enforce that the form field accepts data formatted as the appropriate type. The Data Type can be any of the following:

- **Text:** this option allows for arbitrary text to be entered into the form field
  - Examples: “foo”, “bar”
- **Number:** this option ensures that the form field contains a number (either with or without a decimal)
  - Examples: “10”, “9.75”

- **Date:** this option ensures that the field contains a date. Dates must be in a [format](#) that can be interpreted by .NET.
- **Date/Time:** this option ensures that the field contains a date and time. DateTimes must be in a [format](#) that can be interpreted by .NET.
- **Time:** this option ensures that the field contains a time, which can be in any of the following formats. Times must be in a [format](#) that can be interpreted by .NET.
- **Currency:** this option ensures that the contents of the field can be interpreted as an amount of currency
  - Examples: “22”, “22.32”
- **Integer Only:** this option ensures that the field contains an integer. It also allows you to specify a minimum and maximum value that the integer can be.
  - Examples: “1”, “394221”
- **Email Address:** this option ensures that the field contains an email address. As a user warning, email addresses shouldn't have any trailing spaces, or they'll be rejected as invalid. Process Director can only validate the format of an email address, meaning that it will ensure the email address has a format that includes an "@" and "." characters in approximately the correct locations in a text string, e.g., `text@text.text`. Process Director cannot otherwise determine the validity of an email address.
- **SSN:** This option ensures that the field contains an American Social Security Number, with or without dashes.
  - Examples: “123456789”, “123-45-6789”
- **Phone Number:** This option ensures that the field contains a phone number. Most formats people usually use to type phone numbers will be accepted, as this data type will ignore most special characters, but match on specific digits and patterns.
  - Examples: “(123) 456-7890”, “+1 123-456-7890”, “123-4FO-OBAR x 4321”
- **ZIP Code:** This option ensures that the field contains an American ZIP Code. The ZIP code can contain only five digits or all nine digits.
  - Examples: “12345”, “12345-6789”, “123456789”
- **Regular Expression:** This option enables you to set a format for data entry for the field, e.g., a Canadian Postal Code format like "M5W 1E6".
- **Concealed Text:** Introduced in Process Director v5.12, this field type replaces the "Password" data type, and will hide the text in the field. Additionally, the behavior of the "Password" data type was changed to check the password strength according to the system settings and return an error if the password strength of the entered value isn't strong enough. This change will automatically convert all form fields that were previously configured as a Password data type to Concealed Text. Fields set to this type will display values with a set number of asterisks (\*\*\*\*) when displayed in a Knowledge View, in a disabled form field, etc.

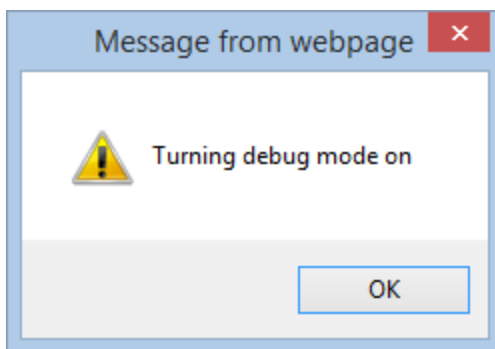


## Changing Data Types

Occasionally, it may be necessary to change the data type of a Form Field. For instance, one common mistake implementers make is setting the data type of a field like a Zip Code to "Number", because the field contains only digits. The natural assumption implementers make is that, since the field contains only digits, it must be a number field. This is generally incorrect, since the best practice is that if you aren't going to do math on a field value it isn't a number. Moreover, treating fields like Zip Codes as numbers will cause them to sort differently (in numerical order) than they would sort if they were text (alphabetical order).

Often, problems don't become apparent until a number of processes have already been started, so you not only wish to change the data type for future instances of the Form, you also want to change the data type for past instances as well. Happily, Process Director enables you to make the change retroactive by following the procedure below:

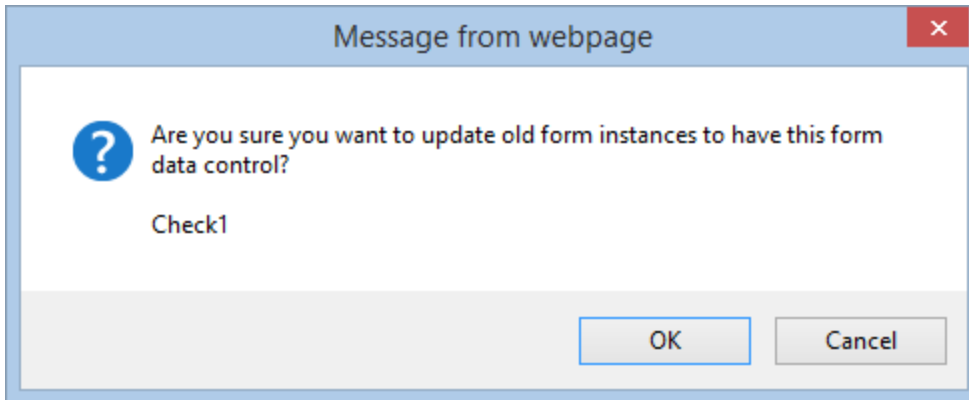
1. Place the mouse pointer in the workspace tab area at the top of the page, and click the right mouse button. This will place the page in Debug mode, and a dialog box will appear notifying you that the mode has been changed. Click the **OK** button to close the dialog box.



2. Open Form Definition and click the the **Form Controls** tab. In Debug mode, the **Form Controls** Tab will look slightly different.

Control Name	Control Type	FCID	DataType	ControlType	Properties	Commands
ApprovalSection	Section	27619c06-b70b-4b2a-b643-ef84822167ca	Number	Section		<a href="#">Edit</a>
ApproverComments	Workflow Signature Comments	9552e27d-043f-4873-805f-fd18ff53adee	String	SignatureComments		<a href="#">Edit</a>
ApproverCommentsLabel	Label	ff29afb1-8115-4ff6-a399-df440cb6fdb7	String	Label		<a href="#">Edit</a>
Buttonarea1	Button Area	d2675079-6bd6-468c-a939-f95ebfc678b6	String	ButtonArea		<a href="#">Edit</a>
Buttonarea2	Button Area	ee649c92-4a2f-4327-bc44-907679e45bec	String	ButtonArea		<a href="#">Edit</a>
CommentLog	Activity Journal	772f6362-5c05-4b74-bf83-1c5f423cfe1	None	ActivityLog		<a href="#">Edit</a>
ComplaintSection	Section	530b13dd-2209-4573-b4ef-5f52ab030264	Number	Section		<a href="#">Edit</a>

3. Click the **Edit** link to open the field properties dialog box for the field you'd like to change.
4. Change the data type of the field to the desired data type and click the **OK** button to close the field properties dialog box.
5. Click the Form definition's **Update** button to save the change to the Form Definition.
6. In the **Commands** column of the **Form Properties** tab, click the **Update Old Instances** link. A confirmation dialog box will appear.



7. Click the **OK** button to confirm the change, close the dialog box, and update all previous Form Instances to the new data type for the field.

**i** An additional debugging option exists to convert existing form data from a text field into a number or date field, as appropriate. This is available in the troubleshooting tab in the IT Admin while in "debug mode" and is called "Update All Form Data to match Controls". This will fix an issue that arises when trying to generate SQL Views that include text fields that stored integer or date data. Earlier version of Process Director stored this data as a string. This option will transfer the data from a string field in the database into the appropriate number or date fields. This doesn't affect the usual operation of Process Director, and is relevant only when trying to include these fields in a SQL View.

**i** For v5.32 and higher, an additional debugging option, which will appear next to form fields that are setting meta data attributes, will cause the system to update the category and attribute data on all old form instances using the form data on the instance.

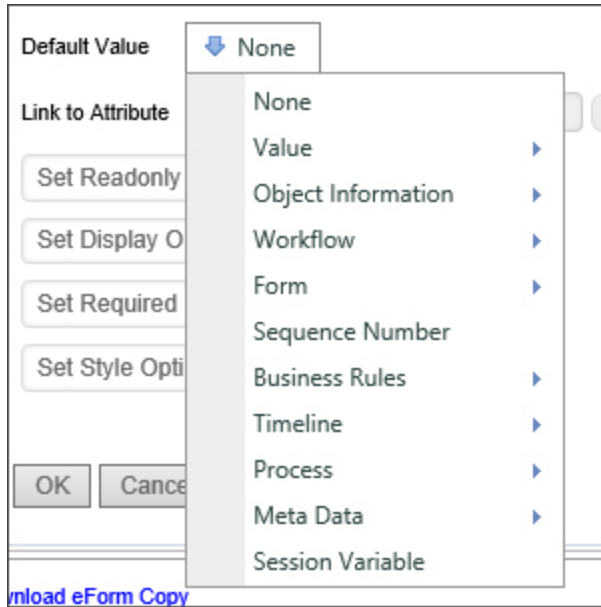
## Min/Max

Certain data types enable you to set a minimum or maximum value for the data the field will accept. This behavior will be different for text or numeric fields.

1. For numeric fields, setting these properties will limit data entry to the **values** that fall within the numeric range specified in the **Min/Max** properties, e.g., a **Max** value of 10 will only allow a numeric value smaller than or equal to 10 to be entered.

- For text fields, the **length** of the text data will be limited to the values specified in the **Min/Max** properties, e.g., a **Max** value 10 will only allow users to enter a text string that is 10 characters long or less.

## Default Value



You can specify the default value of a form field. The Default Value dropdown allows you to select from a list of common default values, like Timeline Instance Names or the results of Business Rules. You can also specify the default be set to a custom value or result from a System Variable.

## Link to Attribute

You can use this picker control to set a Meta Data Attribute from the Form Field.

## Case Property

For [Case Management applications](#), setting the property will save the field as a Case Property, rather than a Form field. Case Management Forms often contain a mix of Form Fields and Case Property fields. Case property fields don't need to be replicated between forms. The Case Property is part of the shared data context that links all Forms to the Case instance. Once a case property is set, that property is available for any other Form in the case to get or set the property.

Some controls, such as radio buttons and dropdowns, can have both a value and a display string. Process Director will save **both** values in the case of those controls, and, by default, the case property returned will be *the value of the control*. The display string, however, can be returned via a Case system variable, using the syntax:

```
{CASE:PropertyName, format=string}
```

So, in any instance where you'll want to evaluate the display string, rather than a value, such as in a condition for a Business Rule, you'll need to evaluate the System Variable as a string, using the syntax above.

If you select the case property from the dropdown menu in the [Choose System Variable](#) dialog, the case property will return the value, and not the display string.

### Set Readonly Options

This option gives you the ability to Enable or Disable a form control including the sections and collapsible sections. You may also Enable or Disable based on a condition using the [Condition Builder](#).

### Set Display Options

This option gives you the ability to Show or Hide a form control including the sections and collapsible sections. You may also Show or Hide based on a condition using the [Condition Builder](#).

### Set Required Options

This option gives you the ability to set the form control as required or not required including the sections and collapsible sections.

You may also set the form control as required or not required based on a condition using the [Condition Builder](#).



Validation and required fields work together in an important way, because Process Director will only validate a field if it contains a value. Process Director will not validate a blank field, even a field that you want specifically formatted using a Regular Expression. Setting a field as required tells Process Director that the field must contain a value, and will impose validation on a blank field. *The fact that you specify a data type or regular expression for a field doesn't indicate to Process Director that the field must contain a value. So, if you want to ensure that Process Director validates a field, even if it's blank, then the field must be a required field.*

## Deleting Controls and Form Fields <#>

Deleting a Form field requires two different actions to remove them from the Form's design.

1. First, you must delete the Form control from the design surface of the Online Form Designer, then save your changes. The field data for deleted controls will, by default, remain as data in the Form instances that were created when the Form originally existed. This data is orphaned, because the original Form control is no longer present on the form.
2. The second step, therefore, is to navigate to the [Form Controls](#) tab of the Form definition to specify how to handle this orphaned Form field data. Controls that have been removed from the design surface of the Form are shown as being marked for deletion on the [Form Controls](#) tab.

Control Name	Control Type	Properties	Commands
DeletedField1 (Deleted Field 1)	TextBox		Edit Delete Remap [Select Form Control] ▼
DeletedField2 (Deleted Field 2)	TextBox		Edit Delete Remap NewField (New Field) ▼
NewField (New Field)	TextBox		Edit

You have two option links for handling this orphaned Form data: **Delete** or **Remap**.

If you click the **Delete** option, Process Director will first display a warning dialog box that requires you to click **OK** to confirm that you wish to delete the field, or **Cancel** to cancel the action. If you click the **OK** button, Process Director will completely erase the field and all of its stored data for all Form instances. **This action cannot be undone**, and the Form field data is completely and permanently erased.

If you need the data in the deleted field to remain accessible on the Form, you can choose to move the field data to a different Form field. To do so, first select the Form field into which you wish to move the data from the provided dropdown, then click the **Remap** link. The existing data from the old Form instances will be remapped to the field you selected for all existing instances. It's best to remap the data to a **new** Form field. If you remap the data to an existing field, any Form data in the existing field will be overwritten by the remapped data. Like deletion, remapping cannot be undone.

In the example shown in the screen shot above, the **NewField** has been selected for remapping the data from **DeletedField2**. Clicking the **Remap** link will complete the remapping operation, and the existing data will be transferred into **NewField**.

### More Configuration Tabs

You can view the documentation for each of the other configuration tabs by using the Table of Contents displayed on the upper right corner of the page, or use one of the links below.

**Properties:** General Form definition properties.

**Edit:** Opens the editor for the form's visual template, the Online Form Designer.

**Custom Task Event Mapping:** Enables you to map Custom Tasks to run on specified form events.

**Validation Rules:** Enables you to add custom validation to the form.

**Set Form Data:** Enables you to set the value of Form fields automatically, based on events or conditions you specify.

**Fill Dropdowns:** Enables you to automatically set the options available in a dropdown control.

**Other Tabs:** Documentation for the **Form Data SQL View**, **History**, **Meta Data**, **Versions**, and **Permissions** tabs.

### Custom Task Event Mapping Tab

This tab specifies Form Custom Tasks to be mapped to different events that occur when the Form is run. The Form Custom Tasks provide enhanced functions to perform a variety of actions, such as to connect to

external data sources or to manipulate data on the form.

The screenshot displays the 'CUSTOM TASK EVENT MAPPING' configuration screen. At the top, there's a form header for 'Purchase Request'. Below it, the title 'CUSTOM TASK EVENT MAPPING' is centered. A 'Select a Custom Task' dropdown is followed by an 'Add Custom Task' button. The main area contains a table with the following data:

Event Name	Custom eForm Task	Options	Condition
AddVendorButton	Execute SQL Command (Insert Vendor) 94052d4f-7e4b-4054-b876-31060795a26a	Configure	Click to create condition ...
AskAQuestionButton	Run Process Configured On 3/3/2015 14c686e0-5b20-4174-9d7d-8c5664cc2a04	Configure	Click to create condition ...
Quantity	Set Focus Configured On 12/6/2016 12:05 PM 0500f047-bffc-4f80-a39f-fde50ad4f342	Configure	Click to create condition ...

Each Custom Task that you map, is mapped to an event that is raised by the Form. The events that are available will appear in the dropdown control that appears in the **Event Name** column. The Form events that can be mapped to a Custom Task, and the order in which the events will be called, is displayed in the table below.

EVENT	DESCRIPTION
specific form event	This allows you to tie a Custom Task to a specific form control that is marked as an event field. Every control that is marked as an event field will appear in the dropdown.
[Event]	Any event from the form (a button, an event field being changed, etc.) will call this event.
[Form Creation]	This event is only called once per form instance to initialize the form data.
[View State Init]	This event is called every time a form is opened. It won't be called for postbacks on the current view.
[Before Conditions]	This event is called prior to evaluating the form conditions (which control the visibility, the required setting, the enabled setting, etc.).
[After Conditions]	This event is called after evaluating the form conditions.
[Before Validation]	This event is called if the user hits a complete button, prior to the

EVENT	DESCRIPTION
	form validation.
[After Validation]	This event is called if the user hits a complete button, after the form validation.
[Form Completed]	This event is called if the user hits a complete button, and the validation was successful.
[Form Display]	This event is called prior to displaying the form. This will be called on every postback, and when the form is first displayed.

The order of the events is very specific. For example, If the user changes a dropdown form field which is an event field, the Custom Tasks mapped to events in the following order will be called if configured: specific form event, [Event], [Before Conditions], [After Conditions], [Form Display].

If a user hits the **OK** button on a form, and the validation succeeds, the events will occur in this order: [Event], [Before Conditions], [After Conditions], [Before Validation], [After Validation], [Form Completed].

If a user hits the **OK** button on a form, and the validation fails, the events will occur in this order: [Event], [Before Conditions], [After Conditions], [Before Validation], [After Validation], [Form Display].

### Mapping an event to a Custom Task

Click on the **Build** button for the **Custom Task Picker** to open the **Choose Custom Task** dialog box.



From the **Choose Custom Task** dialog box, select the Custom Task you want to map. When you select the Custom Task, the **Choose Custom Task** dialog box will close and the name of the Custom Task will appear in the text box of the **Custom Task Picker**. Click the **Add Custom Task** button to add the Custom Task to the Form.

The **Custom Task Event Mapping** tab will now display a new line containing the Custom Task to be configured.

From the dropdown in the **Event Name** column, select the desired event to tell the Form definition to run the selected Custom Task when the event runs.

We now need to configure the Custom Task by clicking on the **Configure** button to open the configuration dialog box for the Custom Task.





By clicking on the **Add Validation Rule** button you can add your own validation rule. You can create a condition under which each validation rule will run. If that condition is met—or, not met, as you desire—you can display a custom validation error message.

Conditions	Rule	Message	Form Field
<a href="#">Click to create condition...</a>	Validation Fails if Condition is Met <input type="checkbox"/>		Choose form control <input type="checkbox"/>

Some special considerations apply when using validation rules to validate fields in an array. Process Director uses slightly different logic for array validation, and that logic may change depending upon the operator used to validate the array values. In the examples below, there are a number of possible validation rules to assist you in understanding the logic used when validating arrays. In these examples, an array field will be designated as an array column with the name of **Col1**, **Col2** etc., while non-array fields will be designated as **Field1**, **Field2**, etc.

Some general rules need to be addressed first, though. Any time you create a validation rule, you are evaluating a value on the left side of the operator with a value on the right side of the operator. Process Director will respond differently depending on whether you are trying to compare a non-array field with an array field, on what side of the operator the array field is used, or whether you are comparing two array fields.

## Comparing Non-Array and Array fields

The way process Director compares array fields with non-array fields is different, based on whether the array is on the left or right side of the operator.

- If the right side of the operator is an array column, the evaluation will occur against a comma-separated list of all values in the column, e.g., "1,2,3".
- If the left side is an array column field and the right side is a non-array field, the evaluation will occur for the column field for EACH row, and each condition is tested against the field value.

Let's assume we have an array field, named **Col1**, containing three rows, with the following values:

- Row 1: a
- Row 2: b
- Row 3: c

Let's also assume that we have a non-array field called **Field1**, with a value of "b".

If we want to do a comparison with **Field1** on the left side of the operator and **Col1** on the right side, Process Director will return the value of **Field1** as "b", and **Col1** as "a, b, c". As a result, some operators will generally not work. In general, **Field1 = Col1** (b = a, b, c), will generally return "false", while **Field1 <> Col1** (b <> a, b, c) will generally return "true". Similarly, **Field1 Contains Col1** will generally return "false". On the other hand, **Field1 In Col1** will validate properly, because "b" is in "a, b, c".

Notice the liberal use of the word "generally" in the preceding paragraph. You *can* make comparisons using the "=" or "<>" operators if the non-array field contains comma-separated values. For instance, if

you put the value "a, b, c" in **Field1**, then **Field1 = Col1** will return true, because "a, b, c" = "a, b, c". You're essentially comparing two comma-separated lists that both have the same value. (Actually, you need to be sure to put spaces between the values in **Field1**, because "a,b,c" won't validate, though "a, b, c" will.)

The logic for comparisons works differently if the array field is on the left side of the operator, and the non-array field is on the right. In that case, process Director will parse each array row for the comparison, which will provide the following results:

VALIDATION RULE	FIELD VALUES	DESCRIPTION
Col1 <b>In</b> Field1	Col1: X 2 Y Field1: 2	Is true if ANY row in Col1 is IN Field1. For example, this is "true" because col1 (row2) is in Field1
Col1 <b>Not In</b> Field1	Col1: 2 3 2 Field1 = 2	Is true if ANY row in col1 is IN NOT IN FIELD1. For example, this is "true" because col1 (row2) is NOT in Field1
Col1 <b>Contains</b> Field1	Col1: 1 22 3 Field1 = 2	Is true if ANY row in col1 contains FIELD1. For example, this is "true" because col1 (row2) contains Field1. In this case "22" contain "2"  NOTE: This is a <i>text</i> comparison NOT a <i>numeric</i> comparison.
Col1 <b>Does Not Contain</b> Field1	Col1: 2 3 2 Field1 = 2	Is true if ANY row in col1 doesn't contain FIELD1. For example, this is "true" because col1 (row2) doesn't contain Field1

## Comparing Array Fields

Comparing two array columns works quite differently. In general, Process Director will compare the values in each row in the array. The operators for array comparisons will work in the following manner.

OPERATOR	COLUMN VALUES	DESCRIPTION								
Col1 <b>In</b> Col2	<table border="1"> <thead> <tr> <th>Col1</th> <th>Col2</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>2</td> </tr> <tr> <td>2</td> <td>X</td> </tr> <tr> <td>3</td> <td>Y</td> </tr> </tbody> </table>	Col1	Col2	1	2	2	X	3	Y	Is true if ANY value in Col1 is in ANY row of Col2. For example, this is "true" because the value in Col1/Row2 is in Col2/Row1
Col1	Col2									
1	2									
2	X									
3	Y									

OPERATOR	COLUMN VALUES	DESCRIPTION								
Col1 <b>Not In</b> Col2	<table border="1"> <thead> <tr> <th>Col1</th> <th>Col2</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>3</td> </tr> <tr> <td>X</td> <td>2</td> </tr> <tr> <td>3</td> <td>1</td> </tr> </tbody> </table>	Col1	Col2	1	3	X	2	3	1	Is true if ANY value in Col1 is NOT IN in ANY row of Col2. For example, this is “true” because the value in Col1/Row2 isn't in Col2.
Col1	Col2									
1	3									
X	2									
3	1									
Col1 <b>=</b> Col2	<table border="1"> <thead> <tr> <th>Col1</th> <th>Col2</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> </tr> <tr> <td>2</td> <td>X</td> </tr> <tr> <td>3</td> <td>Y</td> </tr> </tbody> </table>	Col1	Col2	1	1	2	X	3	Y	Is true if ANY value in Col1 equals the value in Col2 in the SAME Row. For example, this is “true” because Col1=Col2 in Row 1.
Col1	Col2									
1	1									
2	X									
3	Y									
Col1 <b>&lt;&gt;</b> Col2	<table border="1"> <thead> <tr> <th>Col1</th> <th>Col2</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> </tr> <tr> <td>2</td> <td>X</td> </tr> <tr> <td>3</td> <td>3</td> </tr> </tbody> </table>	Col1	Col2	1	1	2	X	3	3	Is true if any value in Col1 isn't the same as the value in Col2 in the SAME Row. For example, this is “true” because Col1 <> Col2 in Row 2.
Col1	Col2									
1	1									
2	X									
3	3									
Col1 <b>Contains</b> Col2	<table border="1"> <thead> <tr> <th>Col1</th> <th>Col2</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>X</td> </tr> <tr> <td>X2</td> <td>2</td> </tr> <tr> <td>3</td> <td>Y</td> </tr> </tbody> </table>	Col1	Col2	1	X	X2	2	3	Y	Is true if any value in Col1 contains the value in Col2 in the SAME Row. For example, this is “true” because Col1 contains Col2 in Row 2. Again, keep in mind that this is a <i>text</i> comparison.
Col1	Col2									
1	X									
X2	2									
3	Y									
Col1 <b>Does Not Contain</b> Col2	<table border="1"> <thead> <tr> <th>Col1</th> <th>Col2</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> </tr> <tr> <td>2</td> <td>X</td> </tr> <tr> <td>3</td> <td>3</td> </tr> </tbody> </table>	Col1	Col2	1	1	2	X	3	3	Is true if any value in Col1 doesn't contain the value in Col2 in the SAME Row. For example, this is “true” because Col1 doesn't contain Col2 in Row 2
Col1	Col2									
1	1									
2	X									
3	3									

### More Configuration Tabs

You can view the documentation for each of the other configuration tabs by using the Table of Contents displayed on the upper right corner of the page, or use one of the links below.

[Properties](#): General Form definition properties.

[Edit](#): Opens the editor for the form's visual template, the Online Form Designer.

[Form Controls](#): Enables you to set individual properties for each field on the form.

[Custom Task Event Mapping](#): Enables you to map Custom Tasks to run on specified form events.

[Set Form Data](#): Enables you to set the value of Form fields automatically, based on events or conditions you specify.

[Fill Dropdowns](#): Enables you to automatically set the options available in a dropdown control.

[Other Tabs](#): Documentation for the [Form Data SQL View](#), [History](#), [Meta Data](#), [Versions](#), and [Permissions](#) tabs.

## Set Form Data Tab

Prior to Process Director v3.53, by far the most common Custom Task used on a Form was the [Set Form Data](#) Custom Task, which was configured on the [Custom Task Event Mapping](#) tab. For Process Director v3.53 and above, the [Set Form Data](#) tab will replace the Custom Task for setting the value of Form Fields. The [Set Form Data](#) Custom Task is still available if required, and will still work properly if you have existing projects that use it, but the vast majority of use cases will be satisfied by using the [Set Form Data](#) tab. The [Set Form Data](#) tab is easier to configure, will reduce the number of Custom Tasks required to configure a Form, and is easier to manage.

In a sample form, the tab will look similar to the one below.

Business Value: Selected Vendor		Add Form Field	
Vendor	Vendor	...	
VendorName (Vendor Name)	VendorName	VendorName	↑ ↓ ✕
VendorAddress1 (Vendor Address first line)	Vendor Address 1	Vendor Address 1	↑ ↓ ✕
VendorAddress2 (Vendor Address Second Line)	VendorAddress2	VendorAddress2	↑ ↓ ✕
VendorCity (Vendor City)	VendorCity	VendorCity	↑ ↓ ✕
VendorContact (Vendor Contact)	VendorPOContact	VendorPOContact	↑ ↓ ✕
VendorState (Vendor State)	VendorState	VendorState	↑ ↓ ✕
VendorZip (Vendor Zip)	VendorZip	VendorZip	↑ ↓ ✕

The [Set Form Data](#) tab is divided into a number of sections, with each section corresponding to a form event. The order in which the events appear on the tab is *not* indicative of when, or in what order, the events will fire. When form events fire—and in what order—are determined by where you are in the form's life cycle.

Both the standard form events (i.e., View State Init, Before Validation, etc.) and the event fields have a configuration section. For example, in the sample above, the [ExistingVendor \(Vendor\)](#) field is an event field that, when fired, will extract data from a Business Value. The value of the [ExistingVendor \(Vendor\)](#)

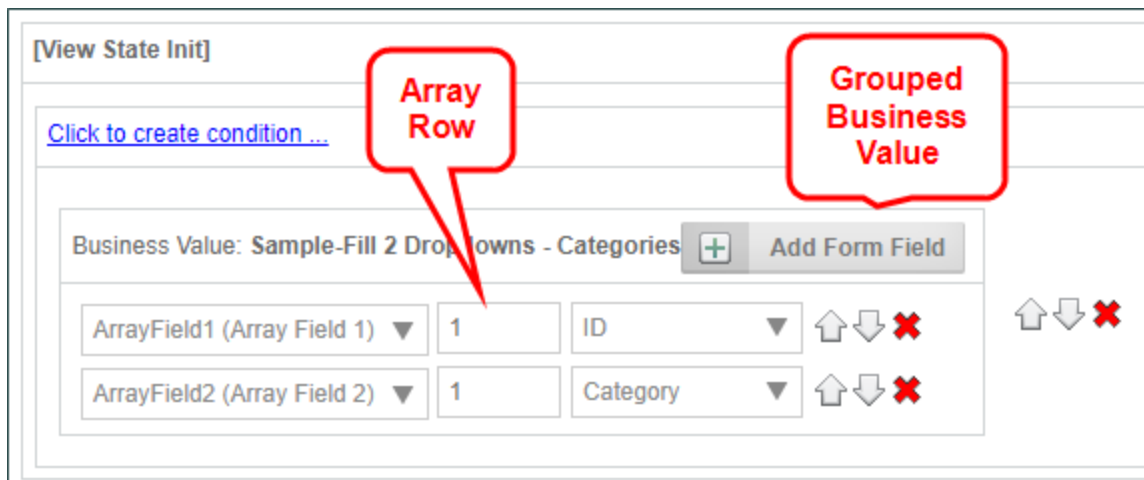
field is passed to the Business Value as a parameter, in order to return the specific data for the selected vendor.

### Adding a Set Form Data Field

1. Find the section that corresponds to the event you wish to configure. For this example, we will use the ServiceLevel field from our sample form, above.
2. Click the **Add Condition** button. A new mapping section for the condition will appear in the tab. You may optionally add a condition by clicking on the **Click to create condition** link, then using the **Condition Builder** to define the condition. If you don't define a condition, the Set Form Data action will occur every time the event fires.
3. Click the **Add Form Field** button. A new mapping row will appear in the condition section.
4. In the **Form Field** dropdown, select the Form field whose data will be set.
5. Repeat steps 3 and 4 as desired.
6. Select the new value for the Form control by using the **Value Object Picker** control.
7. Each row can be ordered in the list of mapping rows by clicking one of the arrow icons to move the row up or down. The row can be deleted by clicking the red "X" icon. Similarly, if you have multiple conditions for an event, the conditions can be ordered as well, so the condition evaluations will occur in the order you desire. Items can only be ordered inside their container section.

### Working with Arrays and Business Values

When using the **Set Form Data** tab to assign form field values from a Business Value. Process Director will automatically group all field changes from a single Business Value into a single row.



When setting the value, you can specify the row number if the field value to be set is contained in an array. If you don't specify an array row when setting data, Process Director will handle the **Set Form Data** action in different ways, depending on the data you are using to modify the array,

When using the **Set Form Data** tab with Business Value properties, the key thing to remember is that the default behavior when adding Business Value data to an existing array is to append new rows when a Row Number isn't specified. In other words, if the array has two existing rows, then, when the event fires that retrieves Business Value data returns four rows, the array will expand to six rows when the four new rows are appended. If you wish to overwrite the existing array data, then the first thing you must do in the

event is to set the array's value to "0", which will clear the array. The new data will then create the appropriate number of rows for the data returned.

It is possible to return data from multiple Business Values in the same event, and Process Director will concatenate the values from multiple Business Values into a single Array change. Process Director will append the number of rows that corresponds to the largest number of results from any Business Value Property in the same Condition Set. With Business Value Properties that return different numbers of results, the Set Form Data will set no data in fields in the appended Array rows that have no corresponding result. In other words, if one Business Value property set returns five rows, and a second returns four rows, then Process Director will append five rows to the array, but the fifth row will have no value from the second Business Value.



When returning data from two different Business Values, you have no way to ensure that the values from each Business Value corresponds with the data from other Business Values. In such a case, you are essentially retrieving two unrelated sets of data. So, if possible, you should retrieve data with multiple columns from a single Business Value.

The behavior with any other type of System Variable is quite different from the Business Value behavior. Instead of appending new rows, the Set Form Data event will apply the new System variable value to each row that is currently in the array, *overwriting the existing data on every row with the new value*.

## More Configuration Tabs

You can view the documentation for each of the other configuration tabs by using the Table of Contents displayed on the upper right corner of the page, or use one of the links below.

[Properties](#): General Form definition properties.

[Edit](#): Opens the editor for the form's visual template, the Online Form Designer.

[Form Controls](#): Enables you to set individual properties for each field on the form.

[Custom Task Event Mapping](#): Enables you to map Custom Tasks to run on specified form events.

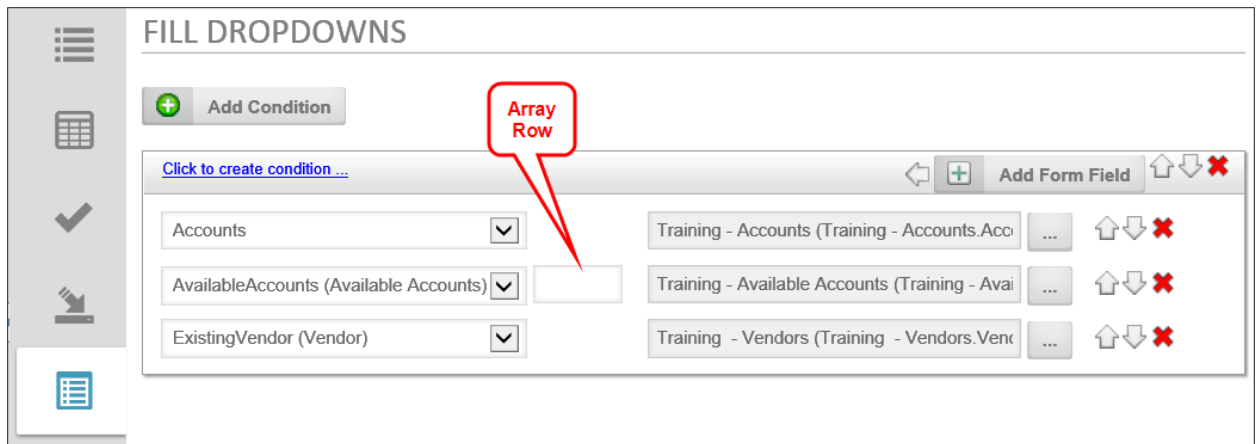
[Validation Rules](#): Enables you to add custom validation to the form.

[Fill Dropdowns](#): Enables you to automatically set the options available in a dropdown control.

[Other Tabs](#): Documentation for the [Form Data SQL View](#), [History](#), [Meta Data](#), [Versions](#), and [Permissions](#) tabs.

## Fill Dropdowns Tab

Another common requirement for Forms is the need to fill dropdown controls with options from a database. With Process Director v3.75 and higher, a [Fill Dropdowns](#) Tab has been added to make it easier to fill dropdown controls with options from the database.

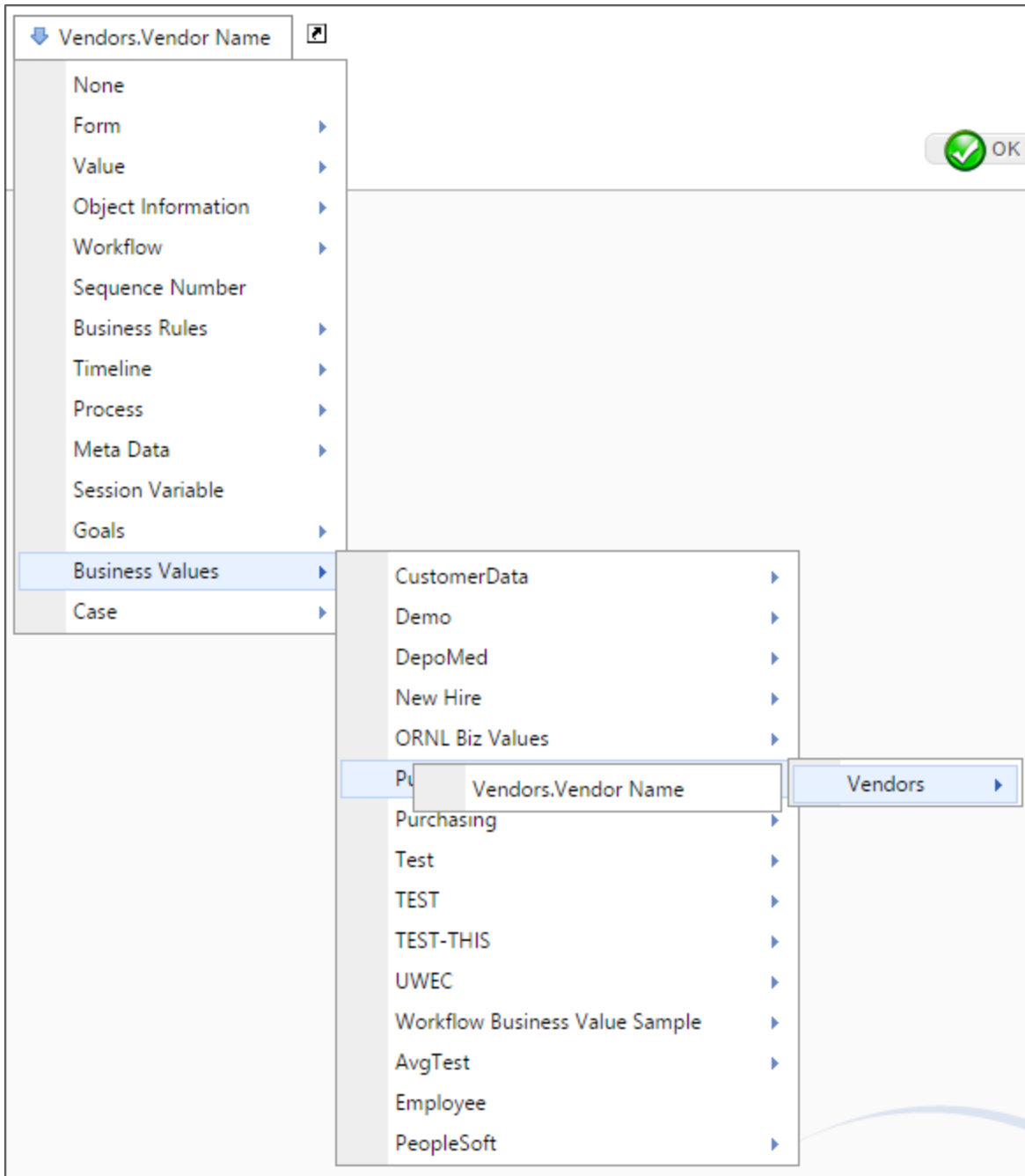


In previous versions of Process Director, the primary method of filling a dropdown was to use a [Fill Dropdown from DB](#) Custom Task on the [Custom Task Event Mapping](#) tab. While this method is still available, the addition of [Business Values](#) greatly simplifies the process, as Form Designers no longer need to know how to access the database, internal or external, from where the dropdown options are obtained. Designers now need only to select the appropriate Business Value to fill the dropdown. In the example shown in the screenshot above, the [ExistingVendor](#) dropdown is populated by the [Vendors.Vendor Name](#) Business Value that returns a list of existing vendor names.

Commonly, some dropdown controls are filled with data based on the value of other Form controls. To ensure that these data dependencies are properly implemented, all other Form Custom Tasks will run prior to the [Fill Dropdowns](#) tab elements being filled. Running the [Fill Dropdowns](#) items last enables all other possible data changes to be processed, so that dependent dropdown controls are filled with the correct options, based on the changed data.

### Filling a Dropdown from a Business Value

1. Click the [Add Condition](#) button to add the event condition to the Form.
2. From the controls dropdown, select the dropdown control to fill.
3. From the object picker, click the [Build](#) button to open the [Choose System Variable](#) dialog box.
4. From the [Choose System Variable](#) dialog box, select the Business Rule that returns the list of items you'd like to use to fill the dropdown, then click the [OK](#) button to close the dialog.



5. If you have multiple dropdowns to fill in the same event, the order in which you'd like the dropdowns to be filled can be changed by clicking the up or down arrow icons to re-order the dropdowns.

Just as with the **Set Form Data** tab, you can apply multiple conditions to multiple events for a complicated set of fill conditions.

When using a Business Value or Business Rule to fill a dropdown, duplicate values are automatically removed from the list of values used to fill the dropdown control's options.



## Filling a Dropdown from an Array

In the Forms section of the [Choose Custom Task](#) dialog box, the [Fill Dropdown From Form Data](#) Custom Task enables you to fill a dropdown with array fields located on the current Form. Use of this Custom Task is configured on the Form's [Custom Task Event Mapping](#) tab, not on the [Fill Dropdowns](#) tab. Please refer to the documentation for [the Fill Dropdown From Form Data Custom Task](#) for information on how to configure it.

## More Configuration Tabs

You can view the documentation for each of the other configuration tabs by using the Table of Contents displayed on the upper right corner of the page, or use one of the links below.

[Properties](#): General Form definition properties.

[Edit](#): Opens the editor for the form's visual template, the Online Form Designer.

[Form Controls](#): Enables you to set individual properties for each field on the form.

[Custom Task Event Mapping](#): Enables you to map Custom Tasks to run on specified form events.

[Validation Rules](#): Enables you to add custom validation to the form.

[Set Form Data](#): Enables you to set the value of Form fields automatically, based on events or conditions you specify.

[Other Tabs](#): Documentation for the [Form Data SQL View](#), [History](#), [Meta Data](#), [Versions](#), and [Permissions](#) tabs.

## Form Data SQL View Tab

You have the ability to create other database views from the Form Definition's [Form Data SQL View](#) tab.

When you create a Form definition in Process Director, there is a tendency to think of the Form instances as being stored in a table, with each Form field as a column, and each row containing a Form instance. In reality, though, the database schema is far more complicated than that. Yet, there are many times when you wish to use tabular data from a Form to use elsewhere in Process Director. Creating a View from the Form definition will return the Form's data in a tabular format, and enables you to choose the specific fields you wish to return in a dataset.

## FORM DATA SQL VIEW

A SQL View can simplify external access to the database tables. It will present a form or case in the database as a single table with columns that represent the form fields or case properties. You must be a System Administrator to be able to create the VIEW from this page. The VIEW is designed to give easy external access, not for performance.

▲ Create new or modify an existing SQL view

Create new SQL view

WebinarForm **✘**

▲ Fields to use in the SQL View

Checkbox2  Dropdown3  Input1

Select All Select None

Re-generate View

SQL Create View Command

View Name  
WebinarForm

View Description

Use NO LOCK on the SQL VIEW generated

Only include forms that have been submitted

```
CREATE VIEW [WebinarForm] AS SELECT tblFormInstance.oFORMINSTID BP_oFORMINSTID, tblContent.sName BP_sName, tblContent.tCreate BP_tCreate, tblContent.tUpdate BP_tUpdate, tblContent.oCreateUser BP_oCreateUser, tblContent.oUpdateUser BP_oUpdateUser, max(case oFCID when '1c405652-a3dc-41ae-a945-f6200a44af56' then CAST(nValue as int) else null end) AS bp_Checkbox2, max(case oFCID when '90d7b6eb-b6be-431a-97d4-6b58f02e213b' then sValue else null end) AS bp_Dropdown3, max(case oFCID when 'a8d3f862-e82f-487c-b869-3ebd614de842' then sValue else null end) AS bp_Input1 FROM tblFormInstance LEFT OUTER JOIN tblContent ON (tblContent.oID = tblFormInstance.oFORMINSTID) LEFT OUTER JOIN tblFormData ON (tblContent.oID=tblFormData.oFORMINSTID) WHERE (tblFormInstance.oFORMID='46fde18f-f909-4ba2-8610-ec30fd198af6') GROUP BY tblFormInstance.oFORMINSTID, tblContent.sName, tblContent.tCreate, tblContent.tUpdate, tblContent.oCreateUser, tblContent.oUpdateUser
```

Create View Copy View to Clipboard

SQL Delete View Command

DROP VIEW [WebinarForm]

Remove View

The **Create new or modify an existing SQL view** section enables you to select whether to create a new view, or to select an existing view to modify or, by clicking the red "X" icon, delete. This section was added in Process Director v5.44.900, to enable you to more easily create and manage multiple views from the same form. For Process Director v6.0.100 and higher, views listed in this section are presented alphabetically, by name.

Views that you created in Process Director prior to upgrading to v5.44.900 will appear in a separate section, labeled as **Legacy SQL Views**, and will display those views separately. If no views were created, this section won't appear at all.

You can create the view you desire by checking the field names in the **Fields to use in view** section of the screen, specifying the **View Name**, then clicking the **Re-Generate View** button. to create the appropriate SQL syntax for the view you desire.

The **Use NO LOCK on the SQL View Generated** property, when checked, will prevent locking on the form or Case instances. This may result in partially saved form or case data being returned when selecting data from the view.

The **Only Include Forms that have been submitted** checkbox will, when checked, return only data for forms that have been submitted. If this property is unchecked, **all** form instance data, including data contained in forms that have been saved, but not yet submitted, will be returned.

Once you click the **Re-generate View** button, the appropriate SQL syntax that creates the view is automatically written into the **SQL Create View Command** section of the screen.

You can customize the SQL syntax as you desire, or you can simply accept the default SQL command that was created when you clicked the **Re-generate View** button. When the SQL syntax creates the desired view you wish to create, click the **Create View** button to automatically create the view in the Process Director internal database.

When the view is created, an editable SQL delete command is created in the **SQL Delete View Command** section. You can run this command by clicking the **Remove View** button, which will delete it from the Process Director internal database.

Once you've created the view, it is accessible in Process Director through the database connector. This means that you can use the views to create dropdown fills from the database, or use the view in reports.

For users of Process Director v4.51 and higher, you have the ability to use the cached data for a form, if the form is configured to use the Form Data Caching feature. The View will return only the cached data, not the most recent data, but may run significantly faster than running it against the tblFormData table. To implement this, check the **Use cached tblFormData table for SQL VIEW?** check box. For more information, please see the [Form Data Caching](#) topic.

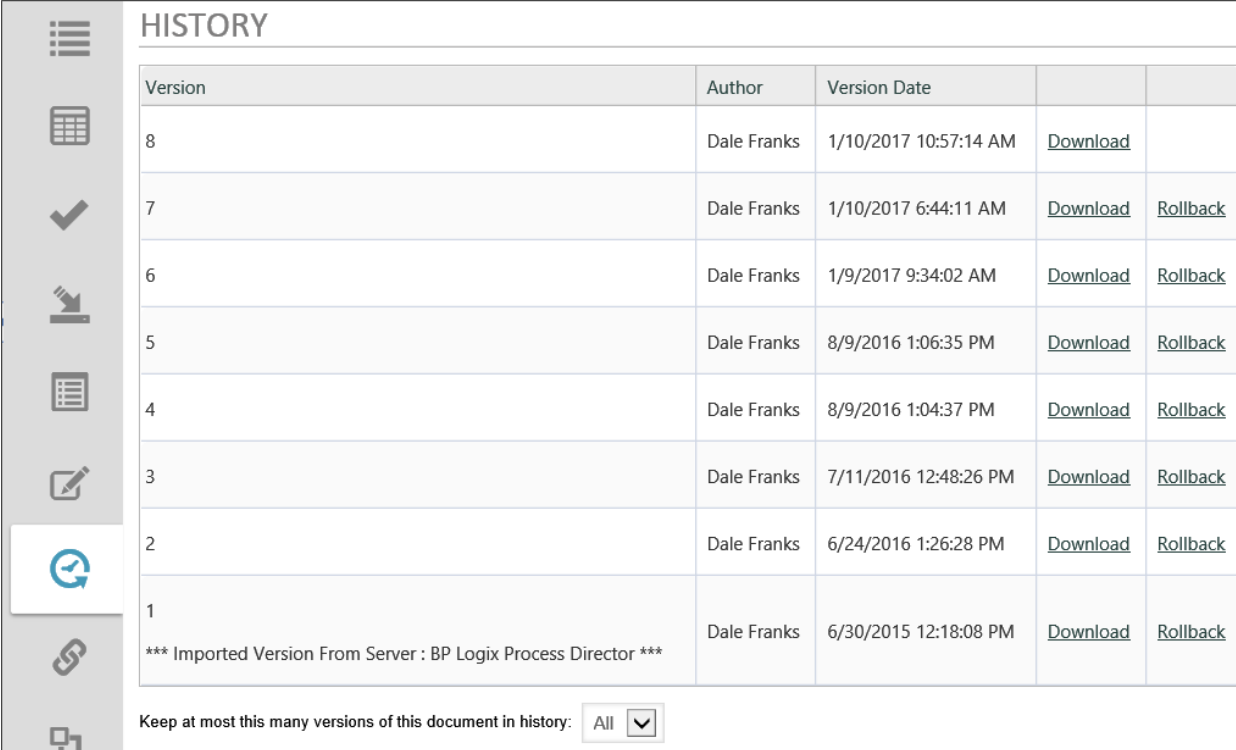


As of Process Director v4.45, the SQL view that is generated for a Form definition casts the field type for integer or checkbox fields to the "int" data type.

## Other Form Configuration Tabs

In addition to the primary configuration tabs for a Form definition, the remaining tabs are largely administrative, and are documented below.

## History Tab #

The History Tab interface features a vertical sidebar on the left with icons for menu, calendar, checkmark, download, list, edit, refresh, link, and print. The main area is titled "HISTORY" and contains a table with columns for Version, Author, Version Date, Download, and Rollback. The table lists eight versions, with the most recent being version 8. A note at the bottom of the table states: "\*\*\* Imported Version From Server : BP Logix Process Director \*\*\*". Below the table is a control for the number of versions to keep in history, currently set to "All".

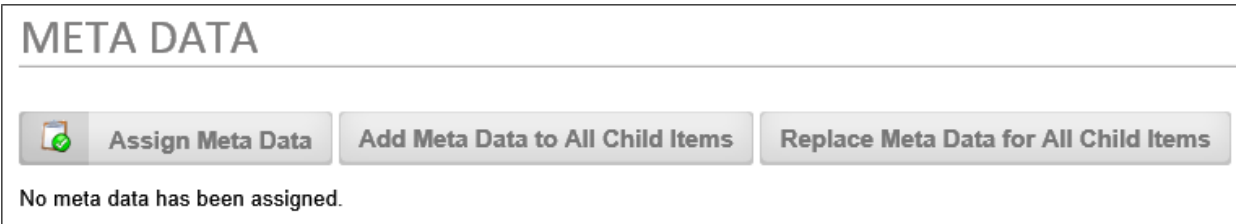
Version	Author	Version Date	Download	Rollback
8	Dale Franks	1/10/2017 10:57:14 AM	<a href="#">Download</a>	
7	Dale Franks	1/10/2017 6:44:11 AM	<a href="#">Download</a>	<a href="#">Rollback</a>
6	Dale Franks	1/9/2017 9:34:02 AM	<a href="#">Download</a>	<a href="#">Rollback</a>
5	Dale Franks	8/9/2016 1:06:35 PM	<a href="#">Download</a>	<a href="#">Rollback</a>
4	Dale Franks	8/9/2016 1:04:37 PM	<a href="#">Download</a>	<a href="#">Rollback</a>
3	Dale Franks	7/11/2016 12:48:26 PM	<a href="#">Download</a>	<a href="#">Rollback</a>
2	Dale Franks	6/24/2016 1:26:28 PM	<a href="#">Download</a>	<a href="#">Rollback</a>
1	Dale Franks	6/30/2015 12:18:08 PM	<a href="#">Download</a>	<a href="#">Rollback</a>

Keep at most this many versions of this document in history:

Process Director automatically keeps a history of Form edits. Each version can be downloaded, deleted, or applied to the Form to replace the current version.

Click the [Download](#) link to download the version, or the [Rollback](#) link to replace the current version with the version to which you'd like to roll back.

## Meta Data Tab #

The Meta Data Tab interface has a title "META DATA" and three buttons: "Assign Meta Data", "Add Meta Data to All Child Items", and "Replace Meta Data for All Child Items". Below the buttons, a message states: "No meta data has been assigned."

[Assign Meta Data](#) [Add Meta Data to All Child Items](#) [Replace Meta Data for All Child Items](#)

No meta data has been assigned.

Meta Data categories and attributes can be assigned to each Form.

When you click the [Assign Meta Data](#) button, it opens a dialog box from which you can choose the Meta Data categories you wish to assign.

When you click the [Add Meta Data to All Child Items](#) button, all of the Meta Data assigned to the Form definition will be copied to all of the Form instances *in addition to* any Meta Data that is already assigned to the instances.

When you click the [Replace Meta Data for All Child Items](#) button, all of the Meta Data assigned to the Form definition will be copied to all of the Form instances *and replace* any Meta Data that is already assigned to the instances.

For more detailed information about implementing and using Meta Data, please see the [Meta Data topic](#).

## Versions Tab #

☰

### VERSIONS

Version snapshots of this eForm are listed below. Most recently applied version: None.

Label:

Object Version Description:

[Create New Version](#)

Label	Saved By	Version Number	Version Date			
Initial Version <i>First pass at the form</i>	Dale Franks	1	1/30/2017 11:07:18 A	<a href="#">Download</a>	<a href="#">Apply This Version</a>	<a href="#">Delete</a>

Process Director implements versioning for Forms, and each version consists of a complete snapshot of the Form. Each version can be downloaded, deleted, or applied to the Form to replace the current version.

To create a new version of the Form, first fill out the version properties.

### Label

The name of the version object.

### Object Version Description

A description of the versions purpose, changes made from previous versions, or other relevant information.

Once you've filled out the properties, click the **Create New Version** button. Process Director will record a snapshot of the version, and a new version row will appear at the bottom of the screen.

Links on each version row give you the option to apply the version as the current version in use, or delete it. If you choose to apply a version, Process Director will display a status screen to notify you of the success or failure of the attempt to apply it.

## Permissions Tab #

Process Director implements fine-grained permissions for all objects. For detailed information on how to set permissions, please see the [Permissions](#) topic.

## More Configuration Tabs

You can view the documentation for each of the other configuration tabs by using the Table of Contents displayed on the upper right corner of the page, or use one of the links below.

[Properties](#): General Form definition properties.

[Edit](#): Opens the editor for the form's visual template, the Online Form Designer.

[Form Controls](#): Enables you to set individual properties for each field on the form.

[Custom Task Event Mapping](#): Enables you to map Custom Tasks to run on specified form events.

[Validation Rules](#): Enables you to add custom validation to the form.

[Set Form Data](#): Enables you to set the value of Form fields automatically, based on events or conditions you specify.

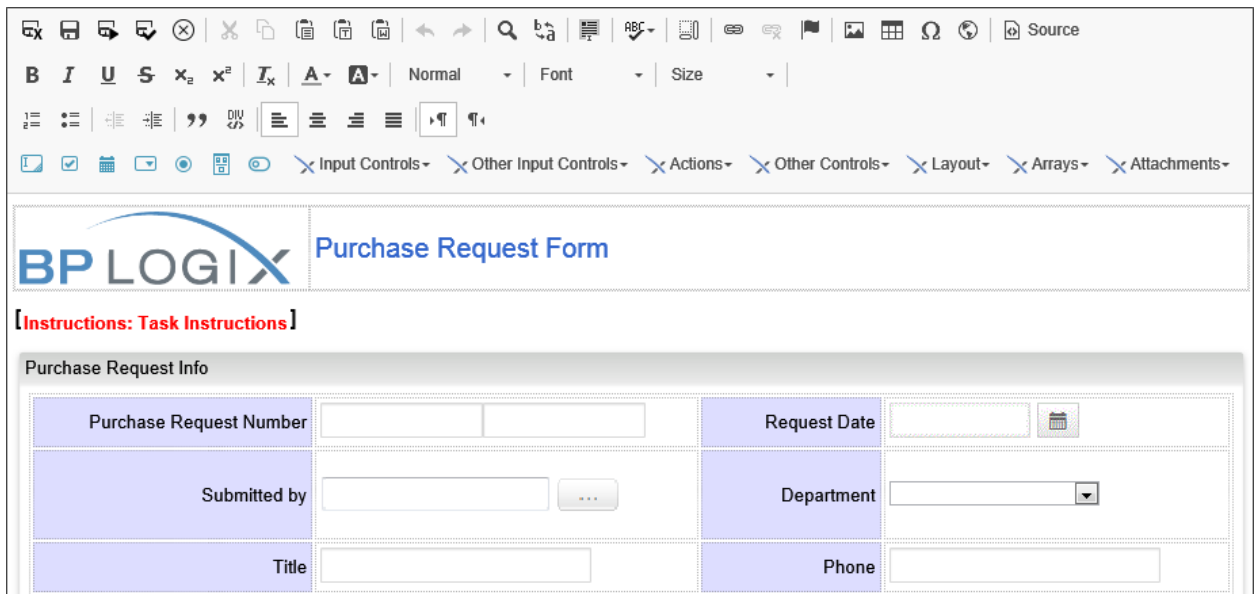
[Fill Dropdowns](#): Enables you to automatically set the options available in a dropdown control.

## Online Form Designer

Form definitions are stored in the Process Director database. The Form's template, which controls the look and feel of the form, available controls, and other physical aspects of the Form that will be presented to end users, can be created in a couple of ways. The primary method for creating and editing Form templates is through the Online Form Designer. You do, however, have the ability to use Visual Studio and the Process Director SDK to create the form as a custom ASCX control. Regardless of the method used to create the Form, end-users are presented with an HTML rendition of the electronic form. This enables your users to fill out and submit forms from within their browser.

The Online Form Designer is the built-in method for creating the template for your Form, and defines where the controls appear on the page. The Form Control Properties of the Form definition on the server control the default values, validation and pre-population of fields that you place on the Template with the Designer.

The Online Form Designer will check out the Form template and display it automatically whenever you click the [Edit](#) tab of the Form definition.



## Browser Extension Conflicts

The Online Form Designer is entirely browser based, and works with any modern browser, though you should be aware of a possible issue. One of the nice things about modern browsers is the wide availability of extensions/plug-ins to customize the browsing experience. Browser extensions offer many useful features that are not available in the stock installation of Chrome, Brave, Edge, Safari, etc. Unfortunately, extensions can also interfere with designing Forms in the Online Form Designer. Specifically, there may be issues with grammar and language checking extensions like LanguageTool or Grammarly.

BP Logix recommends that you turn off browser extensions like these if you design any Forms in the Online Form Designer. Indeed, you should probably, as a best practice, turn off any unnecessary browser extensions when designing Forms. The Online Form Designer has advanced HTML and JavaScript features. Browser extensions that alter the normal operation of the browser, such as modifying the HTML or JavaScript that runs in the browser window, are more likely to interfere with these features, and produce unpredictable results.

For instance, some extensions can interfere with the ability to add and save controls in the Online Form Designer, which can result in some or all of the controls being marked for deletion when the Form's design is saved, even though the controls still exist on the Form. Once these fields are marked for deletion, the Form becomes unrecoverable, except by rolling back to a previous version (if one exists) from the [Versions](#) tab of the Form definition.














Browser extensions can usually be disabled on a site-by-site basis, in the extension's settings.

## User Interface #











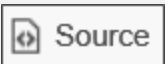






The User Interface for the Online Form Designer consists of a series of toolbars that enable you to add and format content and controls to the Form template. Though you can, if you understand HTML code, perform some more advanced formatting or other functions, ***knowledge of HTML isn't required to create a form using the Online Form Designer.*** No external tools or other software applications are required to

















create Form templates when you use the Online Form Builder. All of the functionality you need to create a template is provided in the toolbars that are arrayed at the top of the screen.




The online Form Designer uses the CKEditor, a third-party component, as the basis of the Form Designer. This component provides a number of built-in tools, in addition to the [Process Director controls](#) which are detailed in this documentation. Please refer to the [CKEditor documentation](#) for the use of these built-in tools. The following built-in tools are available for your use.

TOOL	NAME	
	Save and Close Form	Saves the Form template, closes the Online Form Designer, and returns you to the Form Definition properties.
	Save Form	Saves the current Form's template without closing.
	Save and Run Form	Saves the form and immediately runs it for submission. <b>(Process Director-specific functionality)</b>
	Save and Test Form	Saves the form and runs it in test mode. <b>(Process Director-specific functionality)</b>
	Discard Changes	Cancels your changes and closes the form without saving.
	Cut	Cut content from the template.
	Copy	Copy content from the template
	Paste	Paste content into the template.
	Past as Plain Text	Paste only plain text content into the template.
	Paste from Word	When you copy content from Microsoft Word, this function will clean the Word HTML and paste it into the template in standard HTML.
	Undo	Undoes the last action you took.
	Redo	Redoes the last action you took.
	Find	Find Content in the template.




TOOL	NAME	
	Find and Replace	Finds content in the template and replaces it with content you specify.
	Select All	Selects (highlights) all content in the template.
	Show Blocks	Displays the HTML container blocks that exist in the template.
	Create Link	Adds a hyperlink to text.
	Remove Link	Removes a hyperlink from text.
	Anchor	Adds a named anchor to the page.
	Image	Inserts an image.
	Table	Inserts a table.
	Insert Special Character	Inserts a special character or symbol into the text.
	iFrame	Inserts an IFRAME tag into the page.
 Source	Source	Shows the HTML source code for your template.
	Bold	Applies bold face to text.
	Italic	Italicizes text.
	Underline	Underlines text.
	Strikethrough	Places a strikethrough line in the horizontal center of text.
	Subscript	Formats text as subscript/
	Superscript	Formats text as superscript.

TOOL	NAME	
	Copy Formatting	Copies the formatting of a highlighted section of text. After clicking the tool, highlighting a second section of text will automatically copy the formatting of the initially highlighted text to the second section.
	Remove Formatting	Clears all text formatting.
	Text Color	Changes the color of text.
	Background color	Changes the background color of text.
	Styles	Applies some predefined text styles.
	Font	Sets the font of text.
	Font Size	Sets the font size of text.
	Ordered List	Creates a numbered list for text.
	Unordered List	Creates a bulleted list for text.
	Decrease Indent	Decreases the indentation of text.
	Increase Indent	Increases the indentation of text.
	Blockquote	Applies a predefined style to quoted content.
	Div Container	Places a DIV container onto the page.
	Align Left	Aligns the content to the left side of the page.
	Align Center	Aligns the content to the center of the page.
	Align Right	Aligns content to the right side of the page.

TOOL	NAME	
	Justify	Creates straight, even margins on both sides of the page.
	Left to Right	Sets the text to flow for reading from left to right. This is the default setting.
	Right to Left	Sets the text to flow for reading from right to left. This is useful for languages like Japanese or Hebrew, that are read from right to left.

The remaining toolbar presents you with the BP Logix controls you'll use to place controls on the Form, and, as mentioned, these tools will be discussed in detail in the [Form controls](#) documentation.

 Prior to 15 Jun 2022, a Word-based Form designer was provided with Process Director, but this feature was rendered non-functional by Microsoft moving both ActiveX and Internet Explorer to end-of-life status, and removing all support for these technologies. This feature is, therefore, no longer supported in Process Director. For legacy customers who wish to learn more, please see the [Microsoft Word Form Builder](#) topic.

## Control Tools

You can view the documentation for all of the Form control tools available in the Online Form Designer by using the Table of Contents on the upper right corner of the page, or by clicking one of the links below.

[Form Controls](#): Overview of the available form controls.

[Adding Controls to the Form](#): General information about adding controls the Form's design surface in the Online Form Designer.

[Basic Controls](#): The most commonly-used form design tools.

[Input Controls](#): Controls that are commonly used to collect data, but are a bit less widely used than the basic controls.

[Other Input Controls](#): Additional Input controls, consisting mainly of the different content picker controls.

[Actions](#): These controls enable you to control form actions, like placing buttons, or choosing objects via a picker,

[Other Controls](#): Controls that perform miscellaneous tasks like adding HTML content, or labels.

[Layout](#): Controls that are used to govern the control layout for the template, such as tabs and sections.

[Responsive Layout](#): Controls that implement Bootstrap form layout objects.

[Arrays](#): Controls that enable to you create and control arrays on the Form.

[Attachments](#): Controls that enable you to add and show attachments, such as documents or images, to the Form.

## Form Controls

The Online Form Designer has a control toolbar that contains all of the BP Logix controls you need to build complex Forms. The generic process for adding controls to a Form is addressed in [the Adding Form Controls topic](#), but, for now, let's address the different controls that are available in the Online Form Designer.



The first seven controls on the toolbar are the Basic Controls. These seven controls are the most commonly-used controls on a Form, and their configuration properties are listed in the [Basic Controls](#) topic.

To the right of the basic controls, there are dropdown menus that contain the remainder of the available form controls. Each dropdown menu groups a specific category of control tools for inserting controls onto the template. You can view the documentation for all of the tools available in each dropdown menu by using the Table of Contents on the upper right corner of the page, or by clicking one of the links below.

**Input Controls:** Controls that are commonly used to collect data, but are a bit less widely used than the basic controls.

**Other Input Controls:** Additional Input controls, consisting mainly of the different content picker controls.

**Actions:** These controls enable you to control form actions, like placing buttons, or choosing objects via a picker,

**Other Controls:** Controls that perform miscellaneous tasks like adding HTML content, or labels.

**Layout:** Controls that are used to govern the control layout for the template, such as tabs and sections.

**Responsive Layout:** Controls that implement Bootstrap form layout objects.

**Arrays:** Controls that enable you to create and control arrays on the Form.

**Attachments:** Controls that enable you to add and show attachments, such as documents or images, to the Form.

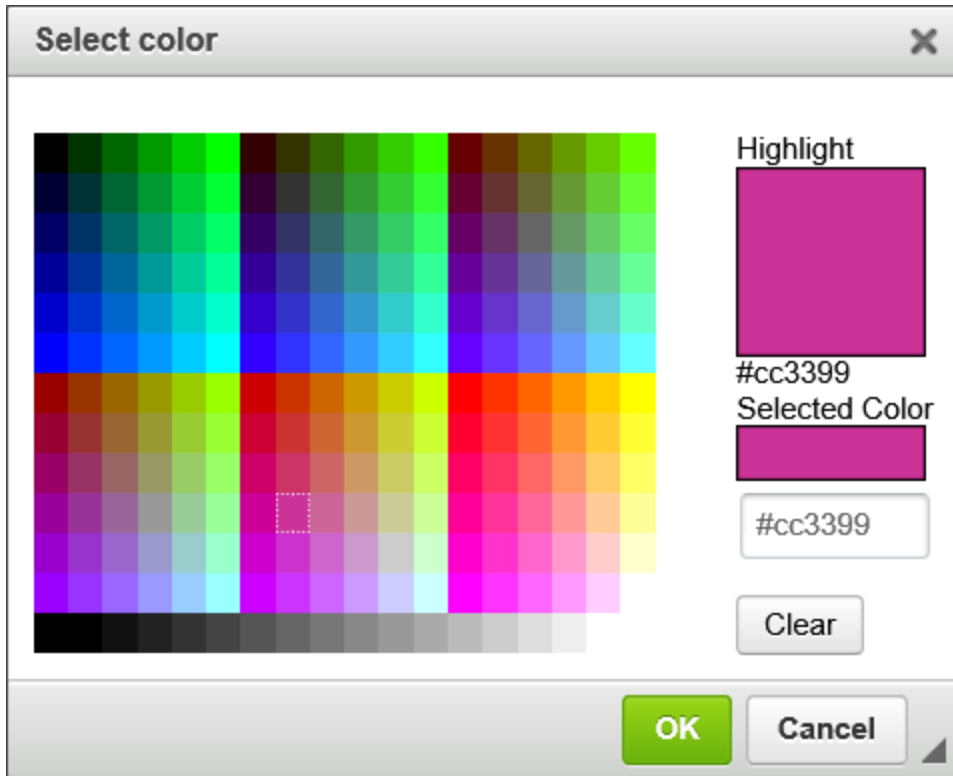
**Form Control Tags:** In addition to the predefined controls that are presented in the Online Form Designer, you can use System Variable syntax to add controls to a form. While the Form Control system variables are only shown as text in the Form's design view, and do not have a visual placeholder like the Form control tools that are available in the UI, these control tags will be converted to the actual controls when they are presented to the user at run-time.

### Color Properties #

Several of the controls in the Online Form Designer have color properties for setting text colors, background colors, etc. A color picker is provided to enable you to configure these color properties.

Text Color	<input type="text" value="#ffffff"/>	<input type="button" value="Choose"/>
Button Color	<input type="text" value="#000099"/>	<input type="button" value="Choose"/>

If you know the hexadecimal HTML color, you can type it directly into the property box, but in most cases, you'll click the **Choose** button to invoke the color picker.



As you move your mouse over the colored tiles, the **Highlight** box will display the color over which your mouse is currently hovering. When you click on a colored tile, the **Selected Color** box will display the color you select, while the text box below the **Selected Color** box will display the Hexadecimal HTML color you've selected. Once you've selected a color, you can save the color and close the dialog box by clicking the **OK** button. When the dialog box closes, the color you've selected will appear in the appropriate color property.

Text Color	<input type="text" value="#cc0099"/>	<input type="button" value="Choose"/>
Button Color	<input type="text"/>	<input type="button" value="Choose"/>

### Icons

Many Form controls have an **Icons** tab as part of the control's configuration. Icon usage can be ubiquitous in a Process Director Form, and can make your Forms look more attractive.

**Input Control** [Close]

Input Control | **Icons** | Formatting | Comments

Color [Text Box] [Choose]

Back Color [Text Box] [Choose]

Icon Number [Text Box] [Choose]

Tooltip [Text Box]

Size [Text Box]

Right

The following properties are configurable from a control's **Icons** tab.

### Color

The foreground color of the icon. You can choose the color from the Color Picker, or type in a HTML Hexadecimal color manually, e.g., "#FFFFFF".

### Back Color

The background color of the icon. You can choose the color from the Color Picker, or type in a HTML Hexadecimal color manually, e.g., "#FFFFFF".

### Icon Number

Every icon in Process Director has an ID number that can be accessed from the [Icon Chooser](#). If you know that ID number, you can enter it here. If you don't know the ID number, you can find it using the [Icon Chooser](#). Picking the icon from the [Icon Chooser](#) will automatically enter the Icon Number in the text box.

### Tooltip

The tooltip to display when a user mouses over the icon.

### Size

The size, in pixels, to use for the icon. This is an optional property, as the icon will show in a standard size if this property is left blank.

### Right

The default display for the icon is to show the icon on the left side of the control. Checking this box will display the icon on the right side of the control.

By way of example, a sample Icons tab configuration is shown below.

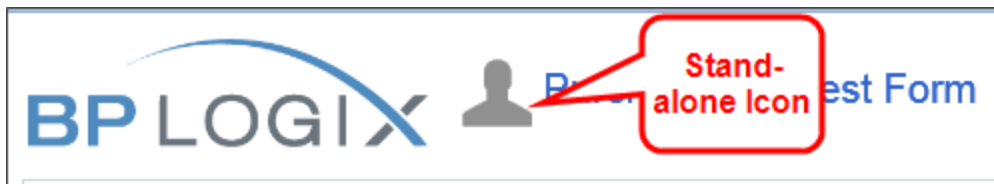
The screenshot shows a configuration window titled "Input Control" with a close button (X) in the top right. Below the title bar are four tabs: "Input Control", "Icons", "Formatting", and "Comments". The "Icons" tab is active. It contains the following fields and controls:

- Color:** A text input field containing "#ffffff" and a "Choose" button to its right.
- Background Color:** A text input field containing "#ff9966" and a "Choose" button to its right.
- Number:** A text input field containing "1004".
- Tooltip:** A text input field containing "This is the icon tooltip."
- Size:** An empty text input field.
- Right:** A checkbox located at the bottom left of the dialog.

Using this configuration, the field will look like the example below when the Form runs.



In addition to using the **Icons** tab to apply icons to a Form control, you can insert standalone icons onto the Form page by using the [Icon Control](#) or [Icon System Variable](#).



### Copying Control Properties #

For Process Director v6.0.300 and higher, Field properties can be copied between controls in the Online Form Designer. This feature makes it possible to apply settings for visibility, enabling, etc., to be copied from a field to another, to help ensure that the fields that require the same property settings can be configured more easily. Two new dropdown menu items, **Copy Field Properties** and **Paste Field Properties** menu items are available from the context menu that appears when you right click a control on the design surface.

In this example, a Form has two controls. This first control is a **Checkbox** that has been set as an **Event Field**, and which has an enabling condition applied.

This is a checkbox

**Check Box Control**

CheckBox   Formatting   **Field Properties**   Comments

Event Field    Synchronize Field Within Process    Encrypt

ToolTip

Friendly Name

Default Value

Link to Attribute

Case Property

Field is Disabled When ...

(New Form Instance? = Yes)

Otherwise Enabled

Immediately below it is an **Input** control, which has no Field Properties set.

This is a checkbox

**Input Control**

Input Control   Icons   Formatting   **Field Properties**

Event Field    Synchronize Field Within Process    Encrypt

ToolTip

Friendly Name

Data Type   Min  Max

Default Value

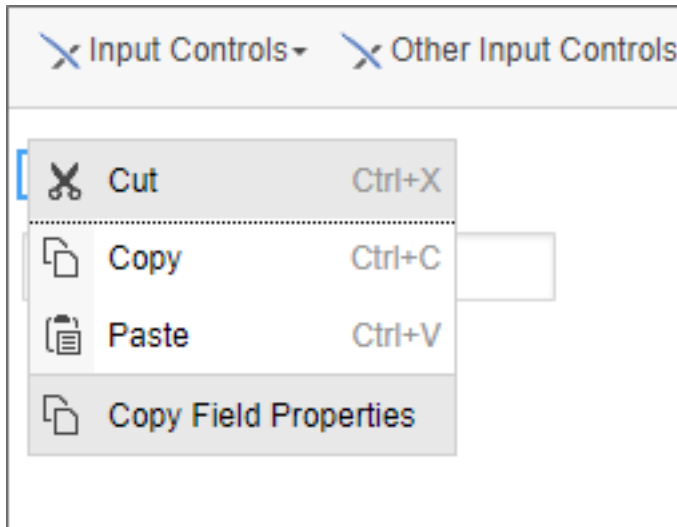
Link to Attribute

Case Property

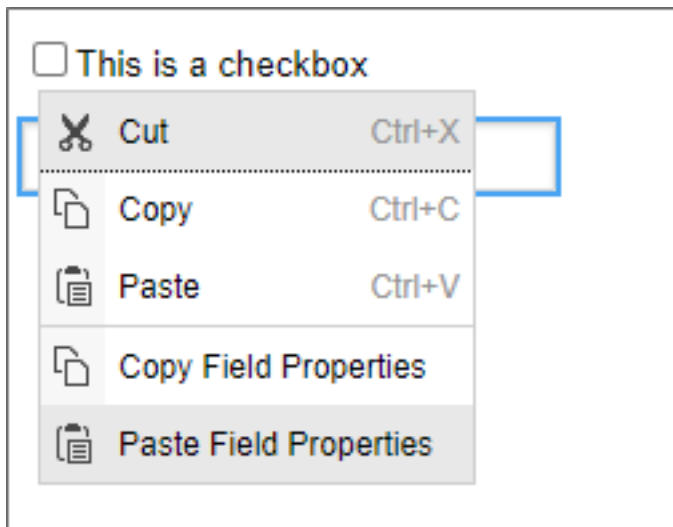
Set Readonly Options

To copy the properties from the **Checkbox** control to the **Input** control, first **right-click** on the "plus" icon for the checkbox control. A context menu will appear that has the **Copy Field Properties** menu item displayed.





Select the **Copy Field Properties** menu item. Once you've done so, you can select the **Input** control and, once again, **right-click** the control's "plus" icon. This time, an additional menu item, **Paste Field Properties**, will appear.



Click the **Paste Field Properties** menu item. When you do so, the properties will be copied from the **Check-box** to the **Input** control's **Field Properties** tab.

This is a checkbox

Event Field     Synchronize Field Within Process     Encrypt

ToolTip:

Friendly Name:

Data Type:  Min:  Max:

Default Value:

Link to Attribute:

Case Property:

Field is Disabled When ...

[Click to create condition ...](#)

Otherwise Enabled

Please be aware that, once the properties are pasted, the field properties are erased from memory. To copy the same properties from the **Checkbox** to another control, you'll need to copy the control properties again, before pasting them to another control. You cannot copy once and then paste the properties multiple times. Only one paste operation per copy is allowed.

### Special Text Formatting #

By default, the Online Form Designer does not enable you to apply different formats to ordered or unordered lists using the text formatting tools built into the toolbar. You can, however, insert a DIV into the form by clicking the **Div** tool on the Online Form Designer's toolbar:



When clicked this tool will present a **Properties** dialog box for configuration.

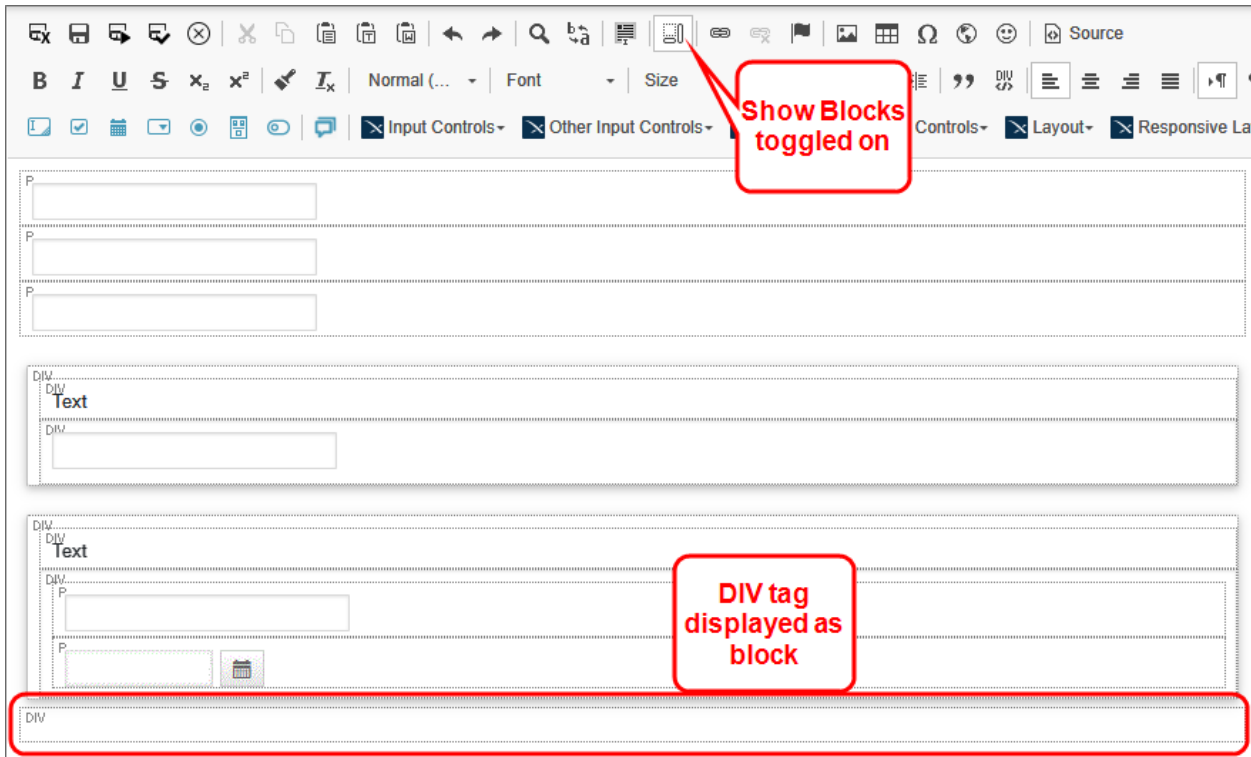
The screenshot shows a dialog box titled "Create Div Container". It has two tabs: "General" and "Advanced". The "Advanced" tab is selected and highlighted with a blue border. The dialog contains the following fields and controls:

- Id**: A text input field.
- Language Code**: A text input field.
- Style**: A text input field.
- Advisory Title**: A text input field.
- Language Direction**: A dropdown menu with the current selection "<not set>" and a downward arrow.
- Buttons**: "OK" (green) and "Cancel" (grey) buttons at the bottom right.

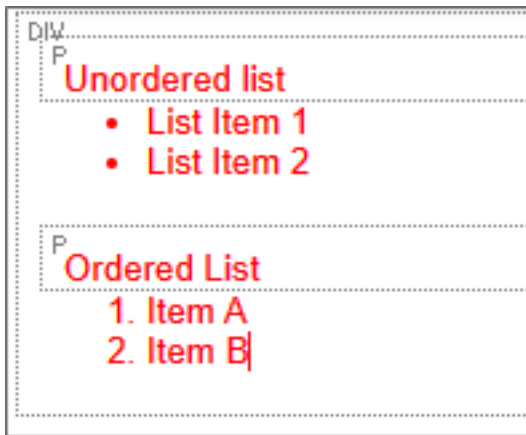
On the **Advanced** tab of the dialog box, you can use the **Style** property to specify the desired CSS style for the contents of the Div tag. You can then click the **OK** button to create the DIV tag on the form. By default dig tags do not appear in the UI. In order to see them, you'll need to click on the **Show Blocks** tool.



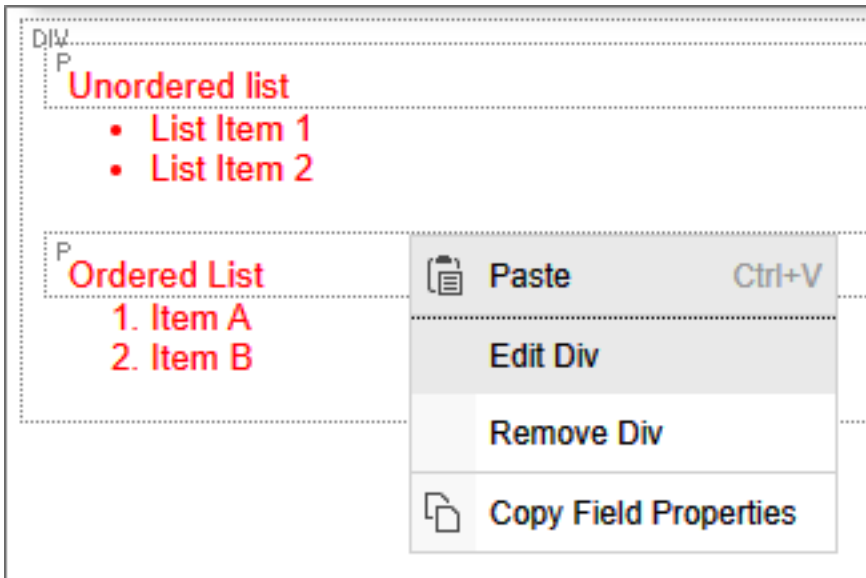
This is a toggle control that turns the visibility for DIV elements on and off. Once toggled on, the Div tag will appear on the Form.



The style you specified for the DIV tag will be displayed in all text, including ordered or unordered lists, that you create inside the DIV tag. For example, setting the `Style` property to `color:red`; will result in the following display for the text:



To open the Properties dialog box for the DIV tag to change its properties, simply right-click inside the Div Tag, and select Edit DIV from the pop-up menu.



Once the **Properties** dialog box reopens, you can alter the **Style** property as desired.

Beyond the use of the DIV tag for lists as shown here, you can insert DIV tags to perform many other special formatting actions for specific parts of a Form.

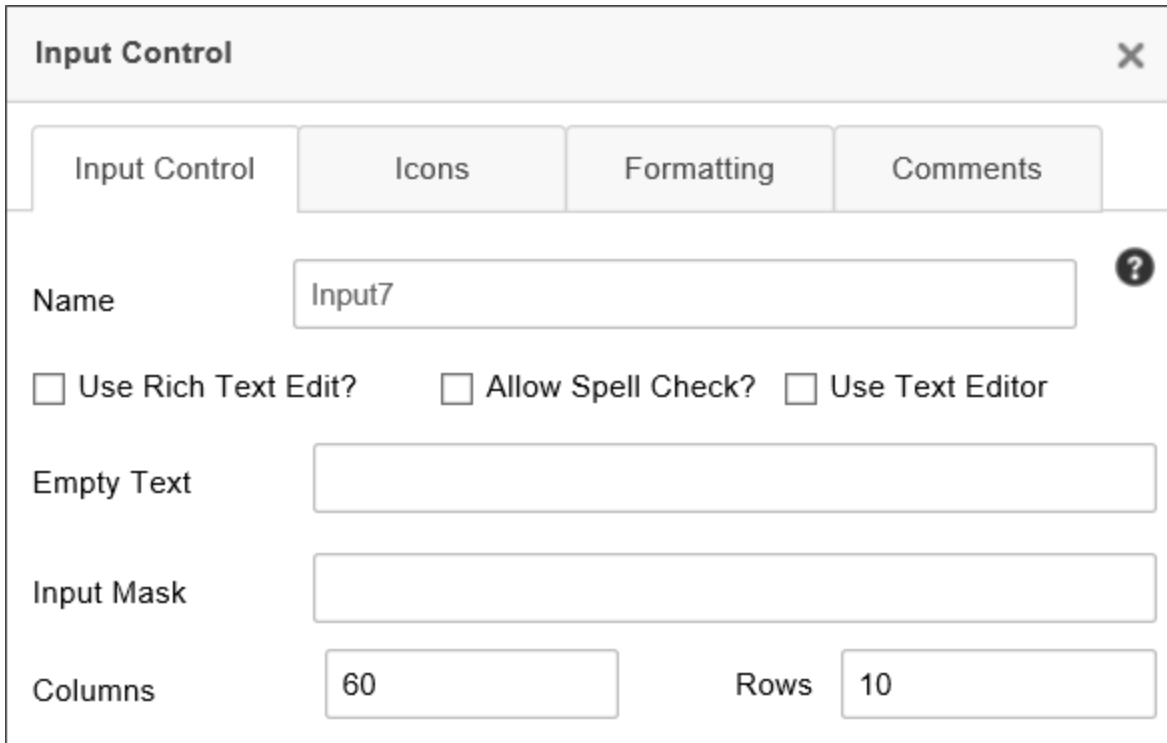
### Basic Form Controls



The first seven tool icons Form Controls toolbar displays the most commonly-used controls for the Online Form Designer.

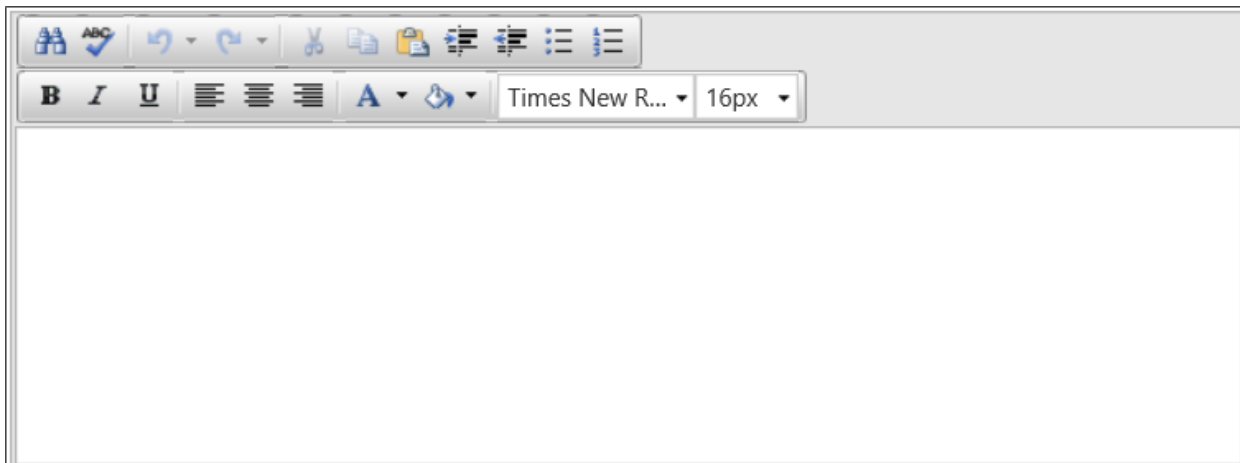
### Input Control

Selecting this item places an **Input** box in the document where the cursor is located. The name of the **Input** control can be changed in the **Properties** dialog box.



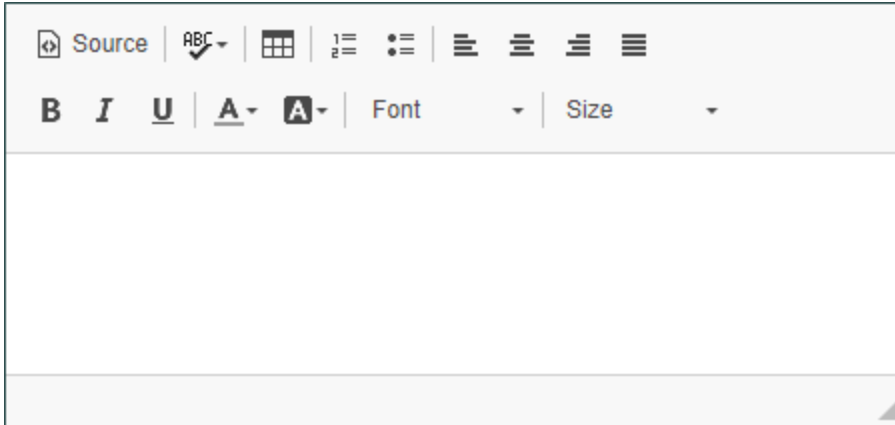
Set the **Name** to the name of the field you want to use on the Form. The fields name and description can be changed in the **Input Control** tab of the properties, while the formatting can be changed in what we conveniently call the **Formatting** tab of the properties.

The input control's default display mode is plain text; however, by checking the **Use Rich Text?** check box, the field will display as a Rich Text field, and allow basic formatting via a simple rich text editor. Additionally, pasting formatted text will carry their formatting over into the Rich Text box, including HTML formatting. (any JavaScript in pasted HTML will be stripped.)



The **Allow Spell Check** property turns on the browser's integrated spell checking for the contents of the **Input** box.

The **Use Text Editor** property enables a more modern text editor than the legacy Rich Text editor.



The **Empty Text** property enables you to enter the text you'd like to display in the input control when it is empty. This is useful for displaying brief instructions for the field.

The **Input Mask** property enables you to specify an input mask that will appear to users when they fill out the form. Input masks can be constructed using the following syntax.

MASK CHARACTER	DESCRIPTION
#	Digit or space (optional). If this position is blank in the mask, it is rendered as a prompt character.
L	Uppercase letter (required). Restricts input to the ASCII letters A-Z.
l	Lowercase letter (required). Restricts input to the ASCII letters a-z.
a	Accepts any character. If this position is blank in the mask, it is rendered as a prompt character.

By way of example, you can, using the above mask characters, set up an input mask for a phone number, using the syntax:

`(###) ###-####`

At run-time, the user will see the input mask:

`(___) ___-_____`

Input masks can't be used if the **Rich Text** checkbox is checked, and the **Input Mask** property will be disabled.

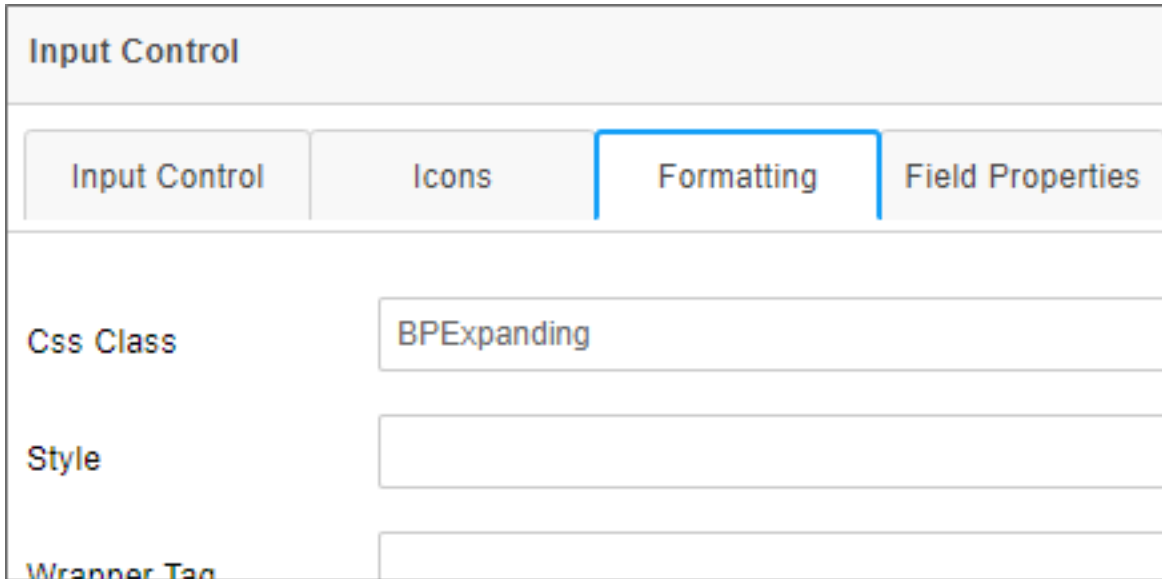
The **Columns** and **Rows** properties enable you to set the horizontal and vertical sizes, respectively, of the control.

For users of Process Director v4.5 and higher, input controls that implement numeric data types right-justify the text by default.

For users of v4.52 and higher, the [AutoMultilineTextBoxResize\\_Custom\\_Variable](#) enables resizing of multi-line **Input** controls.

Additionally, in conjunction with the [AutoMultilineTextBoxResizeClass\\_Custom\\_Variable](#), you can also enable automatic resizing as you type, to automatically expand the text box as you type, to fit all of the text without needing to scroll through it. To enable the automatic resizing feature, you must go to

the **Formatting** tab of the Input control, and set the **CSS Class** property to the class name specified in the `AutoMultilineTextBoxResizeClass` Custom Variable.

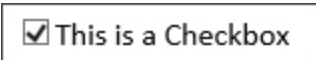


The screenshot shows a configuration window titled "Input Control" with four tabs: "Input Control", "Icons", "Formatting" (which is selected and highlighted with a blue border), and "Field Properties". Below the tabs, there are three input fields. The first field is labeled "Css Class" and contains the text "BPExpanding". The second field is labeled "Style" and is empty. The third field is labeled "Wrapper Tag" and is also empty.

The default class name for this variable is named **BPExpanding**. Please refer to the Custom Variable topics linked above to see more detail about how to implement these features.

### Check Box

Selecting this item places a **Checkbox** control in the document where the cursor is located.



The screenshot shows a single checkbox control. The checkbox is checked, and the text "This is a Checkbox" is displayed to its right.

The name and settings of the **Checkbox** can be modified by right clicking on the field and selecting the **Properties** menu item. Set the **Name** to the name of the field you want to use on the Form.



**Check Box Control**
✕

CheckBox
Formatting
Comments






Name  ?

Text

Check Type  ▼

Vertical

The Online Form Designer enables you to specify the type of control you'd like to use to represent the check box. Using the **Check Type** property, you can set the visual appearance of the check box as one of the following options:

OPTION	VISUAL REPRESENTATION
Checkbox	<input checked="" type="checkbox"/>
iOS	
Light	
Flat	
Skewed	
Flip	

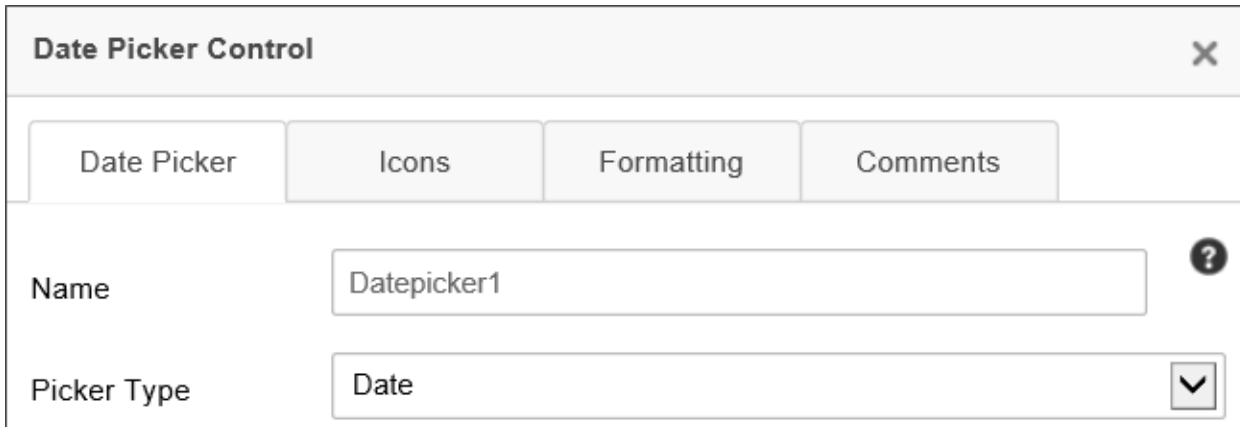
### Date Field

Selecting this item inserts a form control that will display a **Date Picker** control where the cursor is located. The name and size of the input box can be modified by right clicking on the input field and selecting

the [Properties](#) menu item.



When the date control is clicked on in the Form, a popup date picker is displayed making the selection of a date easy for the user. In the [Properties](#) dialog, you can configure whether the picker should be just a date picker, just a time picker, or a datetime picker.



When setting the value of a [Date Picker](#) control using the [Picker Type](#) of "Date" programmatically (for example, using a Custom Task), any time information included will be discarded. To retain the time information, you must set the [Picker Type](#) property to "Date/Time".

While the [Date Picker](#) can be displayed as a Date/Time picker, the times shown are only displayed in hourly increments, and display all 24 hours in a day. The display of the UI control cannot be customized. To customize the time intervals and/or to specify the start and end times to display, you will have to use the [DateTime\\_control\\_tag](#). The control tag enables you to display times in any desired increment of minutes, and to set the start times and end times for limiting the of times displayed to a desired time span.


## DropDown

Selecting this menu item places a [Dropdown](#) control where the cursor is located.

The screenshot shows a configuration window titled "Dropdown Control". It has three tabs: "Dropdown", "Formatting", and "Comments". The "Dropdown" tab is selected. Inside the window, there is a "Name" field containing the text "Dropdown8". Below the name field is a checkbox labeled "Use AutoComplete" which is unchecked. To the right of this checkbox is a sub-panel containing three more checkboxes: "Min Characters" (unchecked), "Max Results" (unchecked), and "Allow Event to fire on selection change" (unchecked). Below these checkboxes is a large empty rectangular area labeled "Items".

In addition to the standard [Name](#) property, the following properties are configurable.

The [Use Auto Complete](#) property, when checked, transforms the [Dropdown](#) control from a standard, selectable dropdown to a type-ahead dropdown. When you check this property, two additional properties become enabled. The first of these is the [Min Characters](#) property, which enables you to specify the number of characters a user must type before the type-ahead dropdown returns matching items. The higher the number, the fewer queries that will be required to return items that match the text you've typed. The second associated property is the [Max Results](#) property, which specifies the maximum number of matching items that will be returned in the type-ahead dropdown. Again, this enables you to minimize the number of items each query returns. Additionally, if you want the field to be an event field, you must check the [Allow event to fire on selection change](#) property, *in addition to* setting the field as an event field on the [Form Controls](#) tab of the form definition.

 To enable Type-Ahead to display properly, the [EnableFormThemes](#) Custom Variable must be set to "false".

For Process Director v5.31 and higher, dropdown Controls that use the TypeAhead feature with a Business Value will invoke a dynamic evaluation of that Business Value if one of the parameters matches the configured Event Control.

## Setting the Items in the Dropdown List

You can fill the dropdown manually by typing list items in the **Items** property box, with one item for each row. When an item is selected in the dropdown list the text of the item will be stored in the form field on the server. If you choose to associate a value that is different than the text, it can be accomplished by adding a colon (":") followed by the desired value. For instance if you want the user to see "August" in the dropdown, but you want to save the month number, "8", in the database, add the following text to the **Items** property box:

August:8

The text to the Left of the colon will be displayed in the dropdown, while the text to the right of the colon will be saved as the value of the field in the database. Similarly, you can save a null value to the database using the colon followed by quotation marks, as demonstrated in the following example:

[Select One]:""

This method also enables you to set a dropdown as a required field, since the NULL value will automatically be considered by the system as having not been filled out, and will cause a validation error.

ITEM VALUE	USER WILL SEE	VALUE STORED IN DATABASE
August	August	August
August:8	August	8
[Select One]:""	[Select One]	NULL

Instead of manually setting the list items for the dropdown using the **Items** property, you can assign a **Dropdown Object** to provide the values for the dropdown control. In the Form definition, in the field properties of the dropdown control, you can set the **Link to Dropdown Object** field to populate the dropdown control with the values in the selected Dropdown content object. See the section of this document entitled [Dropdown List Content Object](#) for more information.

The screenshot shows a dialog box titled "Form Field Properties: Department". At the top, there are three checkboxes: "Event Field", "Synchronize Field Within Process", and "Encrypted Field". Below these are text input fields for "ToolTip" and "Friendly Name". The "Default Value" is set to "None" with a dropdown arrow. The "Link to DropDown Object" field is highlighted with a red rectangle and contains an empty text box with a "..." button to its right. Below this is the "Link to Attribute" field, which is also empty with a "\*" icon and a "..." button. There are four more dropdown menus: "Set Readonly Options", "Set Display Options", "Set Required Options", and "Set Style Options". At the bottom are "OK" and "Cancel" buttons.

You can fill the dropdown list from a database. You can either use the [Fill Dropdowns tab of the Form definition](#) to specify a Business Value to use to fill the dropdown list, which is our primary recommendation, or you can use one of the [Fill Dropdown Custom Tasks](#).

Finally, you can use a Business Rule that returns a List of Users or a List of Groups to fill the dropdown.

**i** Irrespective of the manner you use to fill the options for the Dropdown control, the Best Practice is to ensure that a default blank option is set for the control. Otherwise, the control's value will default to the first item in the options list. In that case, setting the control as required will have no effect, as the control will ALWAYS have a valid value, and thus will always pass validation checks.

## Other Configurations

To set the default value for the dropdown list, specify the value of the item in the Default Value field of the Form definition on the server (see the section named [Form Definition Properties](#) in this document for more information). Dropdowns (and related controls) will display all entries, even those with duplicate values. So, for example, if a dropdown populated with country names has two entries with the display strings "United States" and "USA", each with the value "US", both will be displayed. Using the "None" option in the [System Variable Chooser](#) dialog box to set a [DropDown](#) or [Listbox](#) control in the Form Definition's [Set Form Data](#) tab will now set the value of the control to an empty string.

To configure the dropdown field as a required field in the Form Properties, which will force the user to choose an item in the list, ensure the first entry contains a NULL value. Process Director won't save the form until the NULL value is replaced with a valid value.

## Radio Button

Selecting this menu item places a single **Radio Button** in the document where the cursor is located. You can add multiple radio buttons to the form.

This is a radio button

The properties below are configurable from the **Properties** dialog box.

**Radio Button Control** [X]

Radio button | Formatting | Comments

Name / Group: Radio8 [?]

ID: [ ]

Text: Text

Value: [ ]

Radio buttons as blocks

**Name/Group:** Each radio button you want added to the same group of controls should be given the same **Name/Group** value. To add multiple radio button groups, use different **Name/Group** value for each collection of radio buttons.

**ID:** For Accessibility reasons, **Label** controls must be associated with the Control ID of the control the **Label** is associated with. This property enables you to specify the ID of the **Radio Button** that can be used, in turn, as the **Associated Control ID** for the **Label** control.

**Text:** The text value that you want displayed on the Form.

**Value:** To specify a value for a radio button that is *different* than the text, enter the radio button's value in the **Value** field.

**Radio Buttons as Blocks:** Checking this box will display all of the radio buttons included in the same group as a block.

The radio button input controls can be sized on the page by clicking and dragging the control to the desired size.

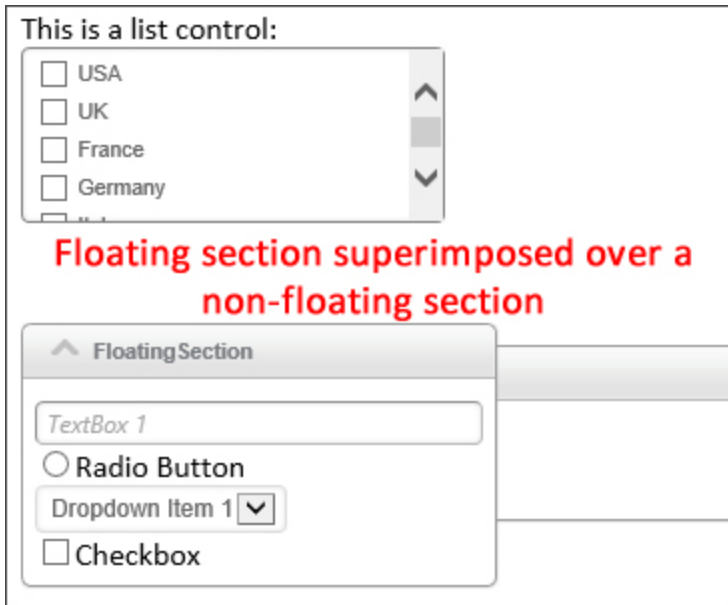
**Radio Button** controls are not list controls, and thus cannot be filled or created dynamically by associating them with a data source like a [Business Value](#). To create radio buttons dynamically, please refer to the [Radio Button List control](#).

## Section

**Sections** with Titles will look like this in the edit mode of the Online Form Designer:

In the **Section**'s properties, you can edit the **Name** of the section, the **Text** of the section (which is visible to the user), whether the section can be collapsed or expanded, whether it is expanded by default or not, and the image URLs for the collapse / expand buttons.

Checking the **Floating?** property will make the section, when expanded, display on top of other sections.



## Expanding/Contracting Sections

In addition to the regular field settings for visibility or enabling the control, Sections can also expand or collapse. The **Section** control's **Value** property determines whether control is expanded or collapsed. Setting the value to "0" will expand the section, while setting the value to "1" will collapse the section. The default value is "0".

You can set the value in the Form control's at design time by either:

1. Checking or unchecking the **Expanded?** property's check box in the **Section** control's **Properties** dialog box in the Online Form Designer, or
2. Setting the **Default Value** property in for the Form field's **Properties** dialog box in the **Form Controls** tab of the Form Definition.

You can also set the value at run time by using the **Set Form Data** tab of the Form definition (the recommended method for most use cases) or using the **Set Form Data** Custom Task in the **Custom Task Event Mapping** Tab of the Form Definition to change the control's value, based on some condition you desire.

A section can also be displayed as **Client Section** by checking the property. Client sections can be expanded or collapsed without a postback to server occurring to show or hide the section. This may be useful for large forms with many sections that need to be expanded or collapsed at run-time by the end user. However, unlike a regular section, the all of the section data for the client section is delivered to the browser, so, even if the section is collapsed, all of the data in the section is discoverable by the end user.

## Switch

Selecting this menu item places a **Switch** control in the document where the cursor is located. The **Switch** control is similar to a **Checkbox**, in that it shows two states, but with a more modern appearance. Like a **Checkbox**, **Switch** controls **always** have a value, irrespective of their selected state, so setting a **Switch** as a required field has no effect.





The name and settings of the **Switch** can be modified by double-clicking on the field and modifying the settings in the **Properties** dialog box. Set the **Name** to the name of the field you want to use on the Form.

**BP Logix Switch Control**
✕

Switch

Comments

Name  ?

Switch Type  ▼

On Text

Off Text




Vertical

The **On Text** property enables you to specify the text that the switch will display when the switch is selected, using the Skewed or Flip **Switch Type**.

The **Off Text** property enables you to specify the text that the switch will display when the switch isn't selected, using the Skewed or Flip **Switch Type**.

The **Switch Type** property enables you to specify the type of control you'd like to use to represent the check box. You can set the visual appearance of the check box as one of the following options:

OPTION	VISUAL REPRESENTATION
iOS	
Light	

OPTION	VISUAL REPRESENTATION
Flat	
Skewed	
Flip	

The default appearance for a switch is an appearance similar to the switches used in iOS.

If the **Vertical** property is selected, the iOS, Light, and Flat **Switch Type** will display as a vertical, rather than a horizontal, **Switch**.

### ***Other Control Tools***

You can view the documentation for all tools available in the Online Form Designer by using the Table of Contents on the upper right corner of the page, or by clicking one of the links below.

**Input Controls:** Controls that are commonly used to collect data, but are a bit less widely used than the basic controls.

**Other Input Controls:** Additional Input controls, consisting mainly of the different content picker controls.

**Actions:** These controls enable you to control form actions, like placing buttons, or choosing objects via a picker,

**Other Controls:** Controls that perform miscellaneous tasks like adding HTML content, or labels.

**Layout:** Controls that are used to govern the control layout for the template, such as tabs and sections.

**Responsive Layout:** Controls that implement Bootstrap form layout objects.

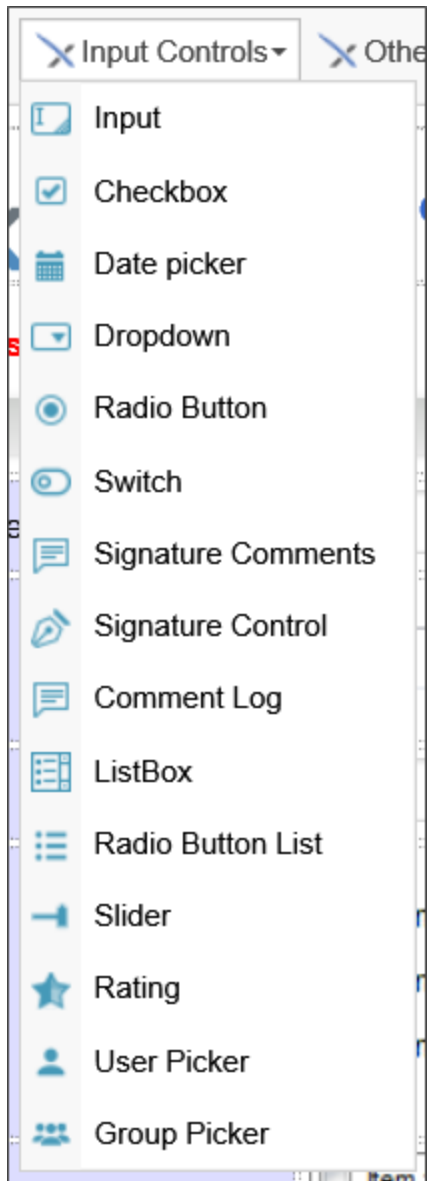
**Arrays:** Controls that enable to you create and control arrays on the Form.

**Attachments:** Controls that enable you to add and show attachments, such as documents or images, to the Form.

**Data List (v6.0.100 and higher):** Controls that enable the display of tabular data on a Form.

**Form Control Tags:** System Variables used to add controls to a Form, instead of using the UI controls.

## Input Controls



The documentation for the [Input](#), [Checkbox](#), [Date Picker](#), [Dropdown](#), [Radio Button](#), and [Switch](#) controls can be found in the [Basic Controls](#) topic. The following additional controls are available from the [Input Controls](#) menu dropdown in the online Form Designer.

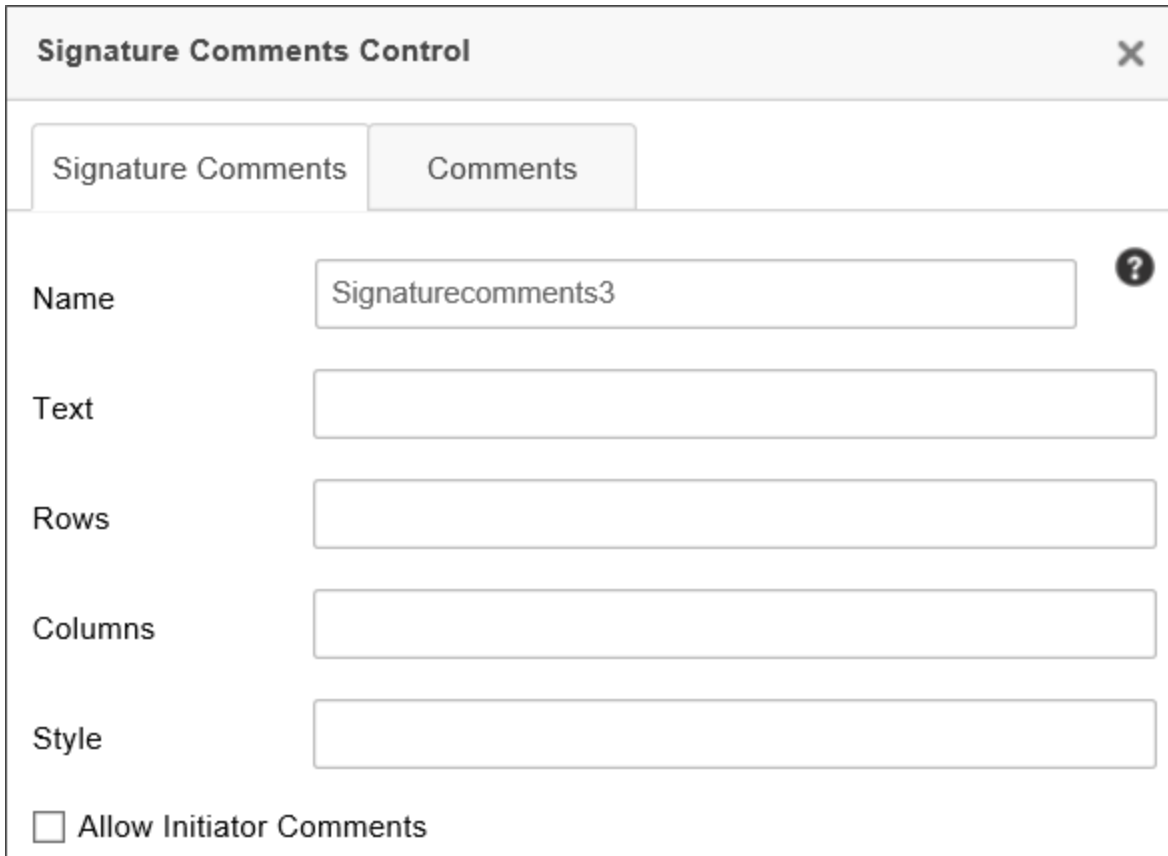
### Signature Comments

This field enables task participants to enter task comments when completing their tasks. Comments aren't required for task completion, by default.

[Signature Comments](#) have a major difference from other Form controls. They are **not** stored as Form data, and are thus not technically Form fields. Instead, comments placed in this control are saved as part of the Timeline Activity instance for which they are made. As such, they can't be displayed or retrieved via the use of Form System Variables. They can only be displayed via the [Activity Users Complete](#) system

variable, using the `format=comments` modifier, e.g., `{ACTIVITY_USERS_COMPLETE:ActivityName, format=comments}`. Comments are also displayed in the **Routing Slip** for the Form, and in the Timeline Instance properties for the Timeline Instance.

Similarly, setting this control as required is performed on the **Results** tab of the Timeline Activity, and is set individually for each configured Activity Result. So, for a given Timeline Activity, signature comments may be required for one result, and not required for other results.



The image shows a configuration dialog titled "Signature Comments Control" with a close button (X) in the top right corner. The dialog has two tabs: "Signature Comments" (selected) and "Comments". Below the tabs are several input fields and a checkbox:

- Name:** A text input field containing "Signaturecomments3" and a help icon (question mark) to its right.
- Text:** An empty text input field.
- Rows:** An empty text input field.
- Columns:** An empty text input field.
- Style:** An empty text input field.
- Allow Initiator Comments:** A checkbox that is currently unchecked.

**Text:** The default text to display in the comments box.

**Rows:** Sets the height of the control.

**Columns:** Sets the width of the control.

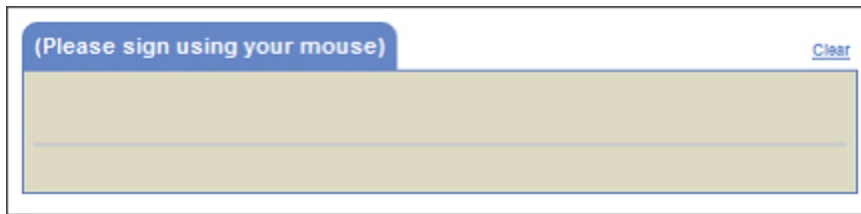
**Style:** CSS Style commands to apply to the control.

**Allow Initiator Comments:** When checked the control will accept signature comments from the initiator/submitter.

**i** In order to enable the initiator to enter comments, you must set the visibility properties for the control to display to the initiator by applying the visibility condition, "New Form Instance = True". By default, this control is hidden in new form instances.

## Signature Control

The **Signature** control enables the user to draw a signature on a Form using a mouse or stylus.



The ESIGN Act of 2000 guarantees that a signature stored via this control is legally binding:

“ [A] contract...can't be denied legal effect, validity, or enforceability solely because an electronic signature or electronic record was used in its formation. [ESIGN Act Section 106(2)]

The following properties are configurable:


The **Prompt** property enables you to configure a prompt message displayed to the user prior to signing.

The **Height** and **Width** properties enable you change the size of the **Signature** control by specifying the size in pixels. The **Signature** control does, however, have a minimum size of 350x100 pixels, and it can't be sized smaller than the minimum size.

The **Color** property sets the color of the pen graphic, while the **Background Color** property sets that background color of the signature area.


Finally, the **Pen Width** property sets the thickness, in points, of the pen graphic's lines when drawing the signature with a mouse.

In the Form Definition's **Form Controls** tab, this control has **Min** and **Max** properties that you can use to specify the minimum and/or maximum size of the signature, measured in points, that are allowed for the signature field.

 Process Director will disable the signature control after any save (including submission, save, and save and close). To change the default behavior, you can add a simple entry to the "Set Form Data" tab for the form that sets the signature control value to an empty string, with the condition that the control is empty on the display.

## Comment Log

This control enables users to add text of any sort, which will be displayed to all subsequent users of the Form. The control displays every user comment and the time the comment was entered.

 This control is a legacy control which remains in the product primarily for backward compatibility purposes. The Journal control is the recommended control for collecting user comments, as it enables threaded comments, and supports other features, such as Milestones, that can't be replicated by the Comment Log.

In this control's properties, you can configure the **Comment Log**'s group name, the control name, the default text, and the width of the log.

The screenshot shows a dialog box titled "Commentlog Control" with a close button (X) in the top right corner. Below the title bar, there are two tabs: "Commentlog Control" (which is active) and "Comments". The main area of the dialog contains five labeled input fields:

- Name:** A text input field containing "Commentlog63" and a help icon (question mark) to its right.
- Text:** A text input field containing "Text".
- Width:** An empty text input field.
- Columns:** An empty text input field.
- Rows:** An empty text input field.

### ListBox

This control is a multi-line list input control.

The screenshot shows a scrollable frame containing a list of checkboxes. The visible items are:

- USA
- UK
- France
- Germany

There are up and down arrow buttons on the right side of the scrollable frame.

The **ListBox** control contains a list of checkboxes in a scrollable frame, where multiple checkboxes may or may not be selected.

The screenshot shows a configuration window titled "Listbox Control" with a close button (X) in the top right corner. Below the title bar are three tabs: "Listbox" (selected), "Formatting", and "Comments". The main area contains the following fields:

- Name:** A text input field containing "Listbox63" and a help icon (question mark) to its right.
- Text:** A large, empty text area for entering list items.
- HTML Height:** An empty text input field.
- HTML Width:** An empty text input field.
- Support multiple selections:** A checkbox that is currently unchecked.

The following properties are configurable:

**Name:** The name of the control

**Text:** The values that will be displayed in the control. Like the Dropdown control, each line constitutes a different display item. Separate display strings and values can also be set, using the syntax `DisplayString:Value`. Just like the Dropdown control, setting the Text property is **not recommended** in most cases. The contents of a List Box should usually be set using a [Dropdown Object](#) or [Business Value](#).

**HTML Height/Width:** The height and width of the control, using HTML measurements, e.g., "300px" or "20%".

**Support Multiple Selections:** When checked, enables users to select more than one item in the list. The value of the control will, if this property is enabled, be returned as a comma-separated list of values.

### Radio Button List

The control returns a single value from a list that is displayed as a series of radio buttons.



**Radio Button List Control**
✕

Radio Button List
Formatting
Comments

**Name**  ?

**Items**

**List Direction**  ▼

**Radio buttons as blocks**

The **Items** property enables you to add items to the **Radio Button List** by placing one item in each line in the **Items** input box. Unlike the **Radio Button** control, which is static, the **Radio Button List** control is a type of list control, like a **Dropdown** or **List Box**, so items can also be added to a **Radio Button List** from a [Dropdown Object](#) or [Business Value](#).

The **List Direction** property enables you to select whether you wish the radio buttons to be arrayed horizontally across the page, or vertically in a single column.

### Slider

The Slider control allows you to select a numerical value using a slider.



In the slider's property window, you can configure its size, minimum and maximum values, and whether or not the handles and tick marks are displayed. The **SmallChange** property determines how much the value of the slider changes when the adjacent arrows are clicked, and the **LargeChange** property determines how much the value of the slider changes when a position on the slider is clicked.

### Slider Control ✕

Slider    Formatting    Comments

Name  ?

Height

Width

Minimum Value

Maximum Value

Small Change


Large Change

Slider Items

Show Handles     Show Ticks (only available for slider with no items)

Remember that if you change the **Width** of the slider, using a percentage width, e.g., "100%" will cause the slider to size itself to the width of its containing object, such as a table cell. If the **Slider** is placed in a narrow table cell, then the slider may be much smaller than you expect. Using a pixel width, on the other hand, will set the **Slider's** width to a fixed width. If you don't set the **Width**, the **Slider** will appear with its

default width, in pixels. So, it's probably best not to set the **Width**, unless you need the **Slider** to appear with a specific size, and then use a width in pixels, e.g., "250px" for the **Width** setting.

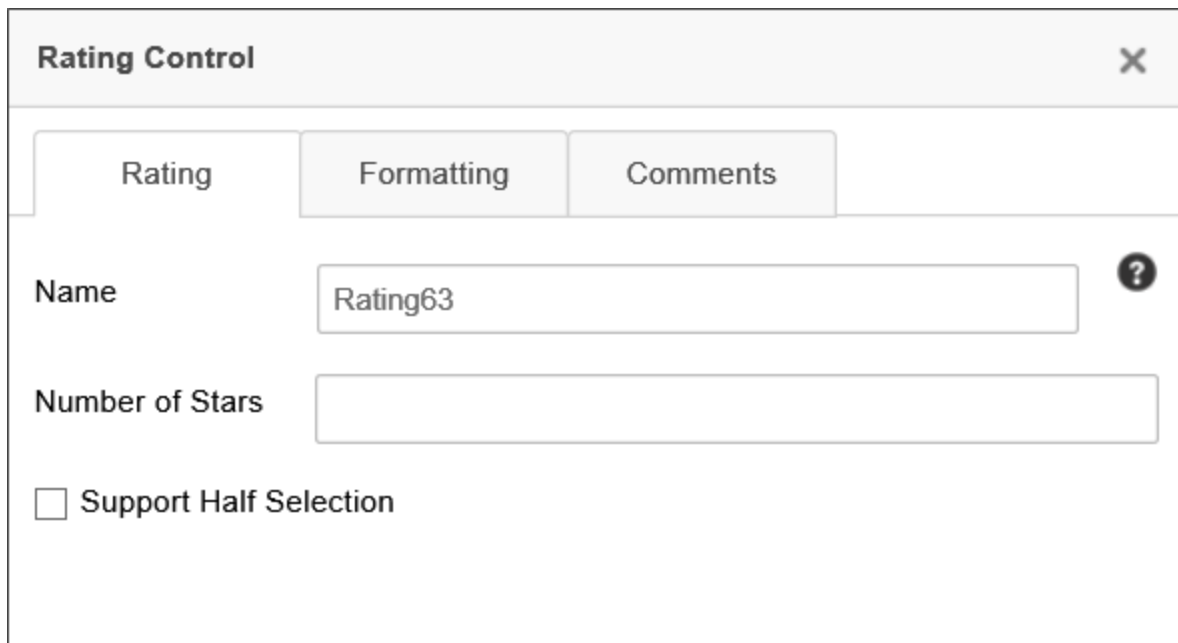
 The Show Ticks property *can't* be used for Slider controls that use Slider Items.

## Rating

This control enables users to set a rating out of a possible number of stars.



The properties window for the control enables you to configure the **Name** of the rating control, the maximum **Number of Stars**, and, via the **Support Half Selection** property, whether the user can select half stars.



Rating Control		X
Rating	Formatting	Comments
Name	Rating63	?
Number of Stars		
<input type="checkbox"/> Support Half Selection		

## User Picker

This control enables the Form user to select Process Director users.

### User Picker Control ✕

User Picker   Formatting   Comments

Name  ?

Picker Type  ▼

Only users in groups

Groups AND / OR

Ignore users in group


Only users in rule

DropDown prompt

HTML Height

HTML Width

Allow multiple users to be selected?

 In most use cases, the User Picker requires that a user be logged into Process Director for the user picker to function correctly. A User Picker of the "Dropdown" type may function on anonymous Forms, but you should use caution when using it on Forms that can be accessed by anonymous users, as the control will reveal some user-related information, which you may not wish to expose publicly, for security reasons. Once a malicious actor can verify the UserID, name, or email of a system user, they might leverage that information for use in gaining unauthorized access to the system.

The **User Picker**'s properties enable you to configure filters for the users you wish to display, which will restrict the choices available to the user. The control can be displayed in the Form as a dropdowns, pop-up, etc., but by default, the control displays as a type-ahead dropdown. The default option is the recommended method of display. You can change the display by setting the desired option from the **Picker Type** property.


You can specify that only users in certain groups show up in the user picker by specifying a comma-separated list of groups in the **Only users in groups** property . If you type "AND" into the **Groups AND / OR** property, then only users belonging to **all** the specified groups will appear as options in the control. If you type "OR", then users belonging to **any** of the specified groups will appear as options in the control.

The **DropDown Prompt** is the text that is shown before an option has been selected in the dropdown, e.g., "<Select User>".

The control won't list users who are members of groups specified in the **Ignore users in group** property, if configured.

The control will only list users returned by the Business Rule specified in the **Only users in rule** property, if configured.

**User Picker** controls display the names of users, but the actual value stored in the **User Picker** is the UID of the user, which is the GUID value that is used as the primary identifier for the user in Process Director. When setting the value of the **User Picker** via a Set Form Data action, you should use the UID or the UserID as the value to set the field value.


 For users of Process Director v5.31 and higher, you can set the field's default value by using the UserID. Previous versions *require* the UID to be set as the value.

## Group Picker

This control enables the Form user to select Process Director Groups.

The screenshot shows a dialog box titled "Group Picker Control" with a close button (X) in the top right corner. Below the title bar are three tabs: "Group Picker" (selected), "Formatting", and "Comments". The main area contains several configuration fields:

- Name:** A text input field containing "Grouppicker63" and a help icon (?) to its right.
- Picker Type:** A dropdown menu with a downward arrow icon.
- DropDown prompt:** An empty text input field.
- HTML Height:** An empty text input field.
- HTML Width:** An empty text input field.
- Allow multiple groups to be selected?:** A checkbox that is currently unchecked.

 In most use cases, the Group Picker requires that a user be logged into Process Director for the user picker to function correctly. A Group Picker of the "Dropdown" type may function on anonymous Forms, but you should use caution when using it on Forms that can be accessed by anonymous users, as the control will reveal some group-related information, which you may not wish to expose publicly, for security reasons.

Similar to the [User Picker](#), the [Group Picker](#) can be displayed in the Form as a dropdown, pop-up, etc., via the [Picker Type](#) property, but the default—and preferred—setting is the type-ahead dropdown mode.

The [DropDown Prompt](#) is the text that is shown before an option has been selected in the dropdown, e.g., "<Select Group>".

The [HTML Height/Width](#) properties set the height and width of the control, using HTML measurements, e.g., "300px" or "20%".

You can configure the [Group Picker](#) to enable the user to select multiple groups by clicking the [Allow multiple groups to be selected?](#) check box. When checked, this property will cause the control to return the selected values in a comma-separated list.

### *Other Control Tools*

You can view the documentation for all tools available in the Online Form Designer by using the Table of Contents on the upper right corner of the page, or by clicking one of the links below.

**Basic Controls:** The most commonly-used form design tools.

**Other Input Controls:** Additional Input controls, consisting mainly of the different content picker controls.

**Actions:** These controls enable you to control form actions, like placing buttons, or choosing objects via a picker,

**Other Controls:** Controls that perform miscellaneous tasks like adding HTML content, or labels.

**Layout:** Controls that are used to govern the control layout for the template, such as tabs and sections.

**Responsive Layout:** Controls that implement Bootstrap form layout objects.

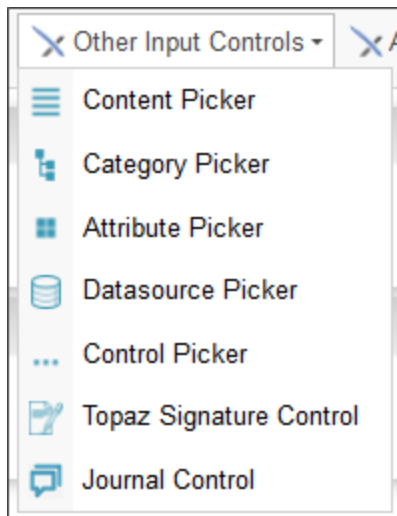
**Arrays:** Controls that enable to you create and control arrays on the Form.

**Attachments:** Controls that enable you to add and show attachments, such as documents or images, to the Form.

**Data List (v6.0.100 and higher):** Controls that enable the display of tabular data on a Form.

**Form Control Tags:** System Variables used to add controls to a Form, instead of using the UI controls.

## Other Input Controls



The following controls are available from the **Other Input Controls** control menu dropdown in the online Form Designer.

### Content Picker

This control enables the user to select objects within the **Content List** (e.g. process definitions , form definitions, etc.)

The **Content Picker** options allow you to set the following properties:

The screenshot shows a dialog box titled "Content Picker Control" with a close button (X) in the top right corner. Below the title bar are three tabs: "Content Picker" (which is selected), "Formatting", and "Comments". The main content area contains four labeled fields:

- Name:** A text input field containing the text "Contentpicker63". To the right of the field is a small circular icon with a question mark.
- Picker Type:** A dropdown menu with a downward-pointing arrow on the right side.
- Document Extension:** An empty text input field.
- Starting Folder:** An empty text input field.

**Name:** The control name.

**Picker Type:** The type of object it can pick from the [Content List](#). The available options are:

- Document
- Folder
- Business Rule
- Knowledge View
- Workflow
- Workflow Instance
- Form
- Form Instance
- Dropdown Object
- Database Connection
- Process Timeline
- Timeline Instance
- Process

**Document Extension:** An optional comma-separated list of document extensions, e.g., "doc, docx, pdf, xls, xlsx", the object must have if it is a document. Only documents with the configured file extension(s) will be visible for selection.

**Starting Folder:** The folder path of the highest-level folder the user will be allowed to access when selecting documents.

### Category Picker

The **Category Picker** enables the user to select from one or more categories in the Process Director Meta Data schema for the partition in which the Form resides.



You can configure the **Name** of the category picker, and the **Starting Category** property is the highest-level category that the user can select when using the category picker. You can also allow the user to select multiple categories by checking the **Allow multiple categories to be selected?** check box.

For more information about Meta Data and its use, please see the [Meta Data topic](#).

### Attribute Picker

This control enables the user to select an attribute in the Process Director Meta Data schema.

The attribute properties can be accessed by double clicking the attribute control. You can configure the control's **Name**, and the **Starting Category** property is the highest-level category that the user can select when using the control to find attributes.

For more information about Meta Data and its use, please see the [Meta Data topic](#).

### Datasource Picker

The **Datasource Picker** control enables a user to select one of the Datasources that have been added to the Process Director installation.

The screenshot shows a dialog box titled "Datasource Picker Control" with a close button (X) in the top right corner. Below the title bar are three tabs: "Datasource Picker" (selected), "Formatting", and "Comments". The main area contains three fields: "Name" with the value "Datasourcepicker63" and a help icon (?), "Datasource Type" with a dropdown arrow, and "DropDown prompt" with an empty text box.

The **Datasource Type** property enables you to select the specific type of Datasource, (e.g., Oracle, SQL Server, Box.net, etc.) that you'll allow the user to select.

The **Dropdown Prompt** property enables you to specify the prompt you'd like the user to see.

### Control Picker

The **Control Picker** control enables a user to select one of the available form controls.

The screenshot shows a dialog box titled "Control Picker Control" with a close button (X) in the top right corner. Below the title bar are three tabs: "Control Picker" (selected), "Formatting", and "Comments". The main area contains three fields: "Name" with the value "Controlpicker63" and a help icon (?), "Control Type" with a dropdown arrow, and "DropDown prompt" with an empty text box.

The **Control Type** property enables you to select the control type that you'll allow the user to select. The Available options are shown below.

Control Type	Input
	TextArea
	Date
DropDown prompt	Button
	Dropdown
	Password
	Array
	Section
	Radio
	CheckBox
	Custom
	CustomTaskConfigSection
	CustomTaskRunSection
	UserPicker
	GroupPicker
	Attach
	ShowAttach
	Label

The [DropDown Prompt](#) property enables you to specify the prompt you'd like the user to see.

### Topaz Signature Control

This control is a special signature control designed to be used with electronic signature pads and signature software produced by Topaz Systems. For users of Topaz Systems components, this signature control would be used in lieu of the regular [Signature](#) control.

In the Form definition's [Form Controls](#) tab, this control has [Min](#) and [Max](#) properties that you can use to specify the minimum and/or maximum size of the signature, measured in points, to display the control.

**i** The Topaz Signature Control will only work with properly installed Topaz Signature devices. These devices, when installed, should be read as a normal input device, like a keyboard or mouse.

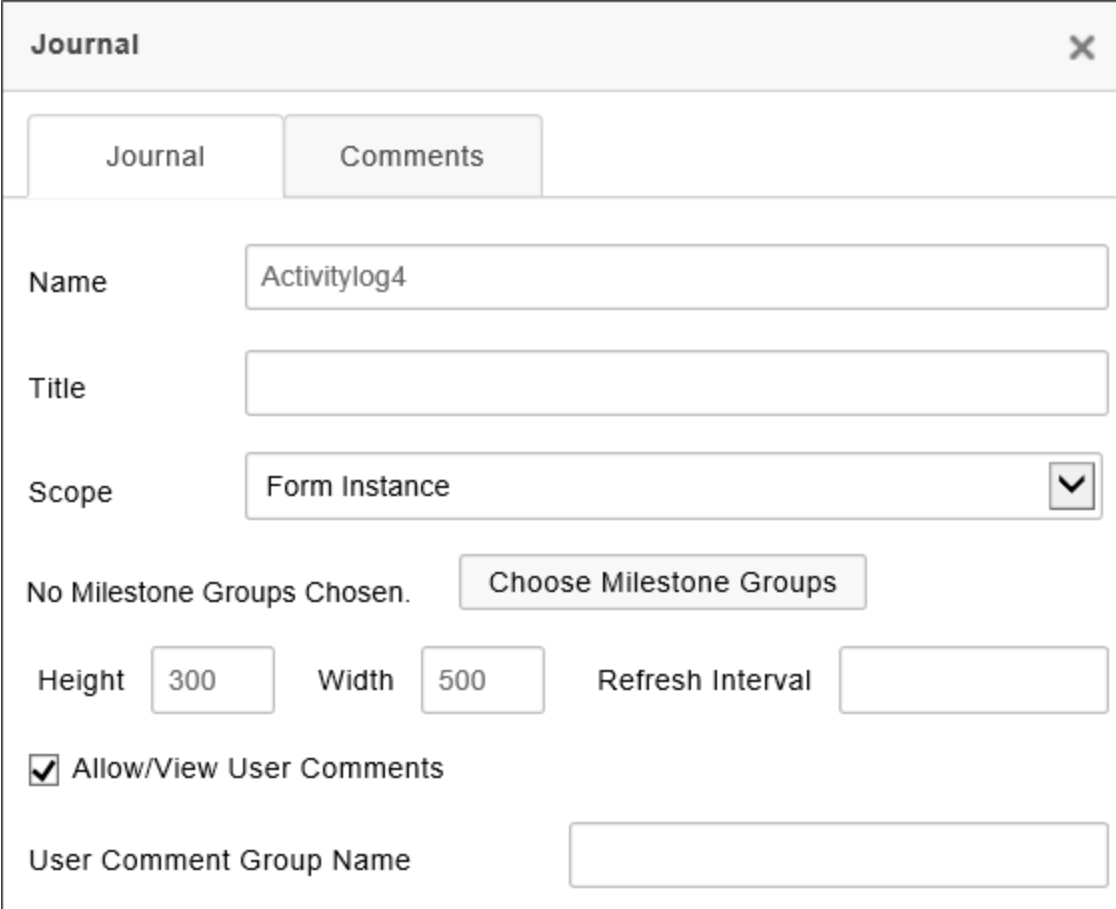
This control requires no special configuration other than that the Topaz signature device, with the appropriate browser plug-ins, is installed on the client machine. Please refer to the documentation for your Topaz Signature device for information on how to ensure the device is configured to work properly with a web browser.

## Journal

The **Journal** control enables you to display threaded comments from users. This control should replace the **CommentLog** control on most forms, and provides a better view of user comments, because comment threading is built into the control, unlike the **Comment Log**.

In addition to serving as a comment log, the **Journal** control has the equally important feature of being able to display all Milestones configured in the **Milestones** tab of the Timeline, Form, and Folder Definitions. Milestones drive collaboration through the **Journal** control, because the **Journal** provides a semantic history of cases, processes, and more, which are derived from the Milestones that occur.

For more information about Milestone events, see the [Milestones](#) topic.



The screenshot shows a configuration dialog box for the Journal control. It has a title bar with the text "Journal" and a close button (X). Below the title bar are two tabs: "Journal" (selected) and "Comments". The main area contains several fields and controls:

- Name:** A text input field containing "Activitylog4".
- Title:** An empty text input field.
- Scope:** A dropdown menu with "Form Instance" selected and a downward arrow.
- No Milestone Groups Chosen:** A label followed by a "Choose Milestone Groups" button.
- Height:** A text input field containing "300".
- Width:** A text input field containing "500".
- Refresh Interval:** An empty text input field.
- Allow/View User Comments:** A checked checkbox.
- User Comment Group Name:** An empty text input field.

The following properties are available:

The **Name** property specifies the name of the control.

The **Title** property specifies the text that will appear in the title bar of the **Journal** control when it is displayed on the form at run time. If you leave this property blank, the default title will be "Journal".

The **Scope** property enables you to determine at what level to capture events. By default, the scope is limited to just Activity log events that are specified in the Form definition. You can change the scope to the following settings.

SCOPE	DESCRIPTION
Limit to Case	Only show events from objects that are part of the current Case.
Limit to Form	Only show events that are specified in the current Form definition.
Limit to Process	Only show events from objects that are part of the current Process.
No Limit	Show any events from any object. This setting uses the widest possible scope, and will show any matching events.

The **Milestone Groups** property enables you to optionally display selected Milestones, in addition to, or in lieu of, user comments. To select the Milestone Groups you'd like to display, click the **Choose Milestone Groups** button to open the Milestone chooser dialog, and select the desired Milestone Groups to display in the log.

The size of the control's appearance at run time is set using the **Width** and **Height** properties, both of which specify the size in pixels or percentages. As a best practice, you should set the **Width** to "100%" to fill the horizontal area of the viewport automatically, for accessibility purposes. The **Height**, on the other hand, should be set in Pixels, which will display the height of the control at a fixed size.

The **Refresh Interval** determines the amount of time, in seconds, between each time the log is refreshed. Leaving this property blank will ensure that the log doesn't refresh automatically.

The **Allow/View User Comments** is turned on by default, so that you can use the **Journal** as a comment log. Unchecking this property will disable user comments. For Process Director v5.26 and higher, the data entry text box on the **Journal** control is resizable at run-time, within its parent container, which won't resize.

The **User Comment Group Name** property enables you to specify a group with which to associate comments. Using this property, you can place multiple **Journal** controls on a case or a form and each **Journal** control can have its own user comments.

### **Other Control Tools**

You can view the documentation for all tools available in the Online Form Designer by using the Table of Contents on the upper right corner of the page, or by clicking one of the links below.

**Basic Controls:** The most commonly-used form design tools.

**Input Controls:** Controls that are commonly used to collect data, but are a bit less widely used than the basic controls.

**Actions:** These controls enable you to control form actions, like placing buttons, or choosing objects via a picker,

**Other Controls:** Controls that perform miscellaneous tasks like adding HTML content, or labels.

**Layout:** Controls that are used to govern the control layout for the template, such as tabs and sections.

**Responsive Layout:** Controls that implement Bootstrap form layout objects.

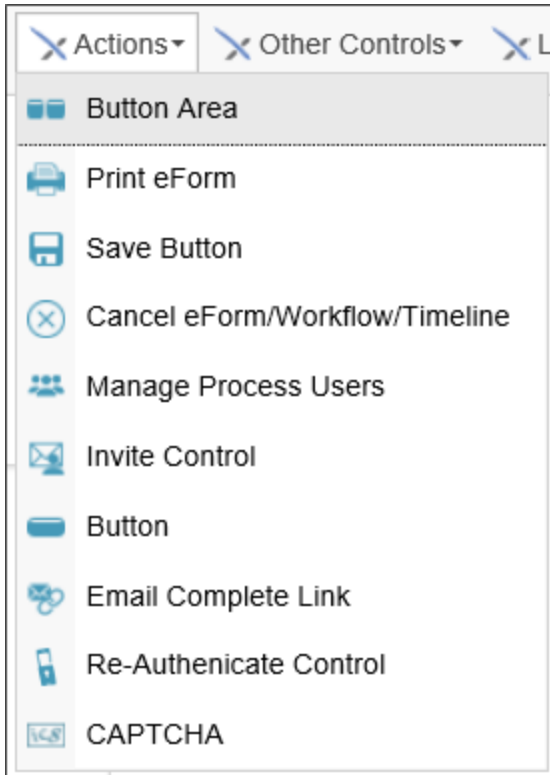
**Arrays:** Controls that enable to you create and control arrays on the Form.

**Attachments:** Controls that enable you to add and show attachments, such as documents or images, to the Form.

**Data List (v6.0.100 and higher):** Controls that enable the display of tabular data on a Form.

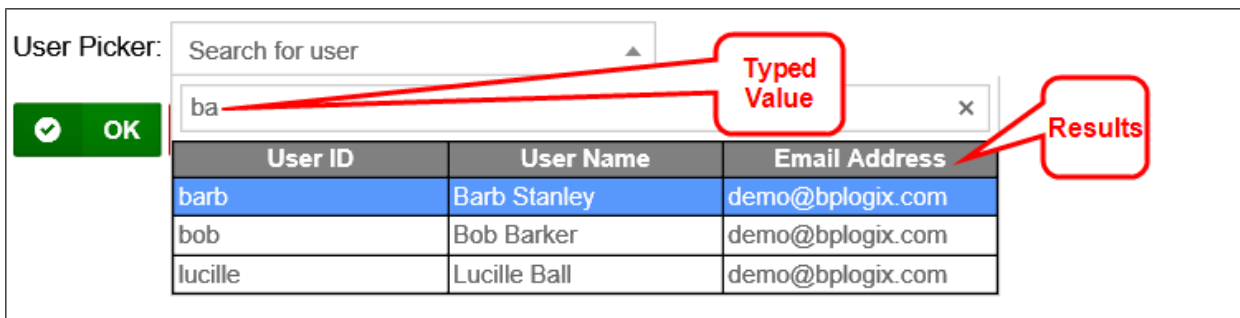
**Form Control Tags:** System Variables used to add controls to a Form, instead of using the UI controls.

## Actions Controls



The controls in this section include the various Picker controls that enable users to pick various Process Director objects from the Form.

All Picker controls incorporate type-ahead selection rather than selecting from a pop-up list, dropdown, or listbox. You can begin to type the name of a user, group, etc., and Process Director will display a drop-down that matches the value you type into the Picker control.

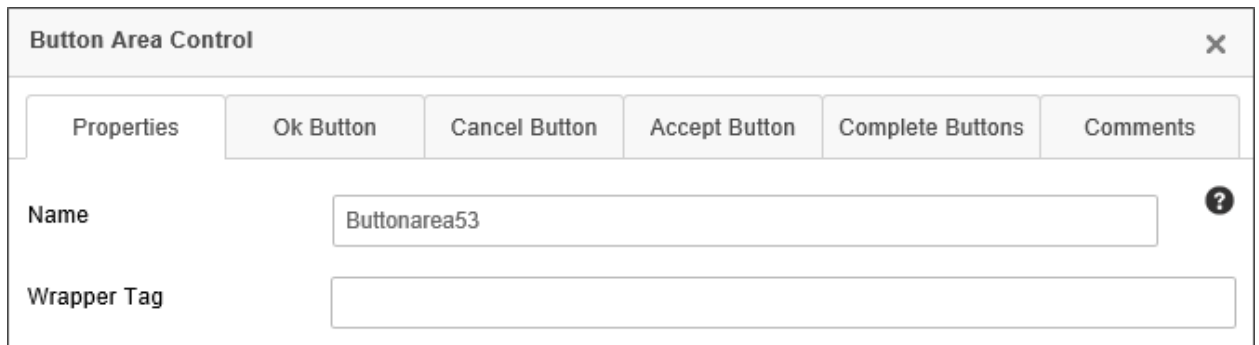


## Button Area

The **Button Area** enables you to specify where the result buttons will appear on the Form page, allowing the user to submit the Form. By default, result buttons always appear at the very bottom of the page, if no **Button Area** control exists. With a **Button Area** control, you can specify the location where the result buttons will appear.

We recommend as a best practice that, when creating a Form that uses a **Routing Slip**, you place the **Routing Slip** at the very bottom of the page, with a **Button Area** control above it. This will enable users to complete the Form without having to scroll to the bottom of the **Routing Slip**.

There are number of tabs for this controls **Properties** dialog, to enable you to individually configure each of the standard buttons that display on forms.



Button Area Control	
<div style="display: flex; justify-content: space-between;"> <span>Properties</span> <span>Ok Button</span> <span>Cancel Button</span> <span>Accept Button</span> <span>Complete Buttons</span> <span>Comments</span> </div>	
Name	<input type="text" value="Buttonarea53"/> ?
Wrapper Tag	<input type="text"/>

On the **Properties** tab, in addition to setting the **Name** of the control, just as you do with other controls, you have the ability to set the HTML tag that will contain the button area, such as an HTML DIV or SPAN tag by setting the **Wrapper Tag** property with the name of the tag, e.g. "div".

The screenshot shows a 'Button Area Control' dialog box with a close button (X) in the top right corner. The dialog contains six tabs: 'Properties', 'Ok Button', 'Cancel Button', 'Accept Button', 'Complete Buttons', and 'Comments'. The 'Properties' tab is active. It features the following controls:

- OK Text: Text input field
- OK Confirm Text: Text input field
- Submit Text: Text input field
- Submit Confirm Text: Text input field
- Icon Number: Text input field with a 'Choose' button
- Color: Text input field with a 'Choose' button
- Back Color: Text input field with a 'Choose' button
- Image URL: Text input field
- Show OK button: Checked checkbox

The **OK Button**, **Cancel Button**, and **Accept Button** tabs have similar properties for configuring each of the button types. The **OK** and **Cancel** buttons are standard buttons that appear on all forms. If the “Assign task to first user to accept” checkbox is checked on a Form instance’s step or Activity definition, the button area will first display an **Accept Task** button enabling the user to accept the task, in addition to the standard form buttons. Task completion buttons won't be displayed until the users accepts the task.

For Process Director v5.25 and higher, the **OK** button will be presented as a **Submit** button for new form instances. Both the **OK** and **Submit Text** and **Confirm Text** properties are configurable. If a **Button Area** control has no **Submit Text** property configured, the control will fall back to the **OK Text** property.

**[Button] Text:** The text that will appear on the button.

**[Button] Confirm Text:** The text that will appear to confirm the button selection when the button is clicked.

**Icon Number:** The ID number of any icon that you'd like to appear as a button icon.

**Color:** The text color for the button.

**Back Color:** The background color of the button.

**Image URL:** If you'd like to use an image, instead of an icon (or text) you can enter the URL of the image to display on the button.

**Show [Button] Button:** This checkbox enables you to show or hide the button on the Form.



The screenshot shows the 'Button Area Control' dialog box with the 'Complete Buttons' tab selected. It features a 'Complete Confirm Text' text input field and a checked checkbox for 'Show Task Complete buttons'.

On the **Complete Buttons** tab, you have a **Complete Confirm Text** property that enables you to set the confirmation text when a task completion button is clicked. You also have a **Show Task Complete buttons** property to show or hide the completion buttons.

The **Button Area** control also automatically incorporates **Form Info String** and **Form Error String** controls, which enables the appropriate form information to display adjacent to the button area when applicable. You can override where the form error text displays, however, by manually placing a **Form Error String** control in the desired location on the Form.

 **Print Form Button**

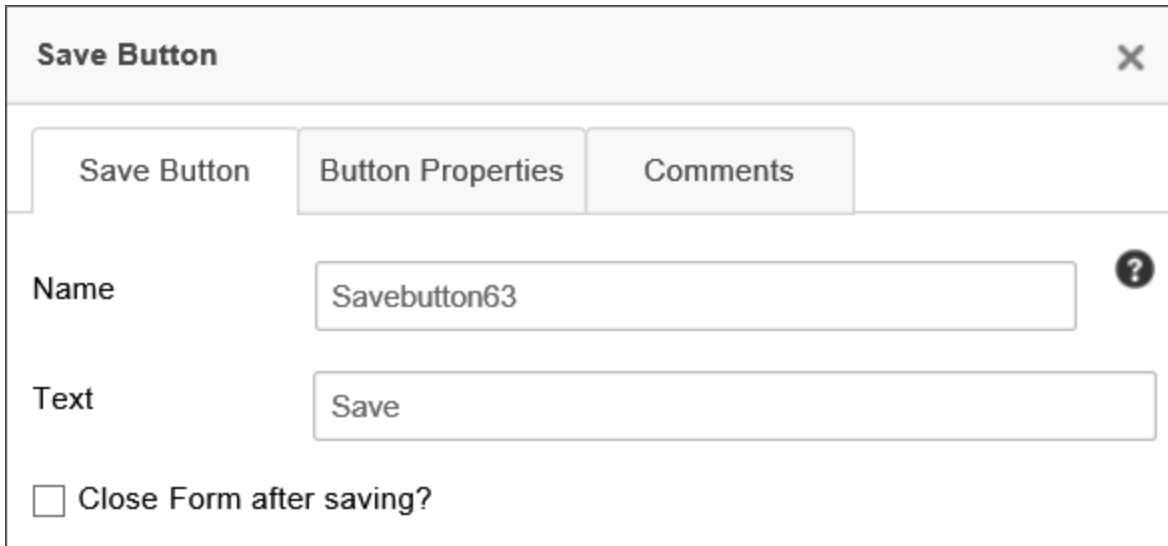
The screenshot shows the 'Print Button' dialog box with the 'Print Button' tab selected. It includes 'Name' and 'Text' text input fields. The 'Name' field contains 'Printbutton63' and has a help icon. The 'Text' field contains 'Print'.

The **Print Button** control is a regular print button that will print the Form when clicked.

The **Text** property configures the text label you'd like to appear on the button.

 **Save Button**

By default, Process Director does not save a Form until the Form is submitted. The **Save Button** enables you to save a Form as an interim version until the Form is submitted. The field values will be saved in the interim version, and you can open the Form at a later time to complete your editing, and submit the form when you're done, which will save the Form in it's final version. Saved Forms will also appear in your **Task List**, to remind you to complete and submit them.



The **Text** property configures the text label you'd like to appear on the button.

Checking the **Close Form after saving** property will close the Form, while keeping it unchecked will save the Form without closing it.


By default, the **Save Button** isn't visible on new form instances. If you want users to be able to save the form on new form instances you must both:

1. Select the **Close Form After Saving** property. Form instances can't be created in the database until the form is submitted to the server. So the button can't appear on new form instances unless the form is submitted on save. If you don't select this property, the **Save Form** button won't appear.
2. In the field properties for the control, on the **Form Controls** tab of the form definition, you must set a visibility condition to make the control visible when New Form Instance? = Yes.



## Cancel Form/Workflow/Timeline

The **Cancel Process** button *immediately* cancels the operation of a Form or process. The cancellation can't be rescinded.

 There is no "Undo" feature for this action! BP Logix recommends this control be used only in Forms that are accessed by administrators, to be used for administrative purposes.

**Cancel Button**
✕

Cancel Button
Button Properties
Comments

Name  ?

Text

Cancel Timeline?

Delete Timeline?

Cancel Workflow?

Delete Workflow?

Delete Form?

Delete Form References?

Cancel Parent Process?

Several properties will enable you to cancel or delete process and Form instances. These properties can be used alone or in conjunction with others. Once the button is invoked, the results are irrevocable.

BP Logix strongly suggests that this button be used *only in Forms that are used by administrators for system management*. Extreme care must be taken to ensure that invoking this button does not cause unwanted effects, or place processes in an unrecoverable error state. *Exhaustive testing should be performed in a development environment* before deploying any Form that uses the **Cancel Button** to your production environment.

PROPERTY	DESCRIPTION
Name	The name of the control
Text	Enables you to specify the text label that is displayed on the button.
Cancel Timeline?	Cancels the running Process Timeline instance. If this control is displayed in a parent process, this will also cancel all associated sub-processes.
Delete Timeline?	Cancels the running Process Timeline instance and then delete it from the system.
Cancel Workflow?	Cancels the running Workflow instance. If this control is displayed in a parent process, this will also cancel all associated sub-processes.

PROPERTY	DESCRIPTION
Delete Workflow?	Cancels the running Workflow instance and then delete it from the system.
Delete Form?	Deletes the current Form instance from the system. Deleting the Form without deleting the associated process will create an orphaned process instance, which may place the process in an unrecoverable error state.
Delete Form References?	Deletes all document attachments or other references for the current Form instance from the system.
Cancel Parent Process?	If invoked in a subprocess, cancels the subprocess' parent process. If the subprocess was invoked synchronously, it will also cancel the running subprocess.

### Manage Process Users

This control enables a user to delegate tasks to other users, remove users from tasks, and perform other process management functions without going to the process' administrator's page.

The **Manage Users** tab of the properties window for this control allows you to set the name, button text, and group name for the control. You can also specify what features the Form user is allowed to access.

✕
Manage Process User Control

Manage Users

Button Properties

Comments

Name ?

Text

Group Name

Step Name

Activity Name

In the current process, allow user to:

Add Users

Reassign Users

Cancel Users

Remove Users

Start Pending Users Only

Restart Users

Complete Users

PROPERTY	DESCRIPTION
Name	The name of the control
Text	Replaces the default text on the control's button.
Group Name	The name of Process Director user groups from which the users will be displayed. This is useful to restrict the users that can be selected. The Group(s) can be specified as a group name or a system variable that returns a group name.
Step Name/Activity Name	The name of the step or Activity, in which you wish the control to appear. If left blank, the control will appear in all steps/activities. You can enter multiple Activity names, separated by commas.

PROPERTY	DESCRIPTION
Add Users	Checking this property will enable the user to add new users to a task.
Cancel Users	Checking this property will enable the user to cancel users in a task.
Start Pending Users Only	Checking this property will enable the user to start any pending task users.
Complete Users	Checking this property will enable the user to complete the user task.
Reassign Users	Checking this property will enable the user to reassign users to a task.
Remove Users	Checking this property will enable the user to remove users from a task.
Restart Users	Checking this property will enable the user to restart the task users.

The Button Properties tab allows you to configure the URL of the button image and the style of the button, as well as the behavior upon clicking the button. These property are the standard button formatting properties that appear in the **Button Properties** tab of the [Button control](#).

## Invite

The **Invite** control allows a user to invite other users in Process Director to a task. The inciter can select to add the invitees to perform the task with the inciter, or to reassign the task from the inviter to the invitee.

The **Invite** tab contains a number of options to configure the Form control.

**Invite Control**
✕

Invite
Formatting
Comments

Name

Picker Type  ▼

Only users in groups

Groups AND / OR

Ignore users in group

Only users in rule

DropDown prompt

HTML Height

HTML Width

Allow multiple users to be selected?

Reassign the Inviter

Cancel Inviting User

PROPERTY	DESCRIPTION
Name	The name of the control
Text	Replaces the default text on the picker's button ("...").

PROPERTY	DESCRIPTION
Picker Type	Determines whether the user picker will show as a pop-up dialog box, or a dropdown control.
Only user in groups	The name of Process Director user groups from which the users will be displayed. This is useful to restrict the users that can be selected.
Ignore users in group	The name of Process Director user groups from which the users won't be displayed.
Dropdown Prompt	A brief prompt message that will appear in the dropdown, if the Dropdown picker type is selected.
HTML Height	The desired height of the control in pixels.
HTML Width	The desired width of the control in pixels.
Cancel Inviting User?	Checking this box will remove the task from the user's <a href="#">Task List</a> when others are invited.
Allow multiple users to be selected?	Allows more than one user to be invited.
Reassign the inviter?	Reassigns the task to the inviter.

When using the system variable syntax for this control, the system variable will return all of the formatting options for a [user system variable](#).



### Button Control

The **Button** control places a configurable button control on the Form. The actions associated with the button control can be set in the Form Definition's **Custom Task Event Mapping**, **Set Form Data** or **Fill Dropdowns** tab. There are two tabs that can be configured to specify the properties of the control

### Button Tab

**Button**
✕

Button
Button Properties
Comments

Name

Button63

?

Text

Button



The **Name** property specifies the name of the control.

The **Text** property configures the text label you'd like to appear on the button.

## Button Properties Tab

This tab enables you to configure additional properties for the Attach button.

The screenshot shows a dialog box titled "Button Control" with a close button (X) in the top right corner. Below the title bar are three tabs: "Button", "Button Properties", and "Comments". The "Button Properties" tab is selected. The form contains the following fields and controls:

- Image URL:** A text input field.
- Icon Number:** A text input field followed by a "Choose" button.
- Color:** A text input field followed by a "Choose" button.
- Back Color:** A text input field followed by a "Choose" button.
- Confirm Text:** A text input field.
- OnClickClientClick:** A text input field.
- Alt Tag:** A text input field.
- Style:** A text input field.
- Access Key:** A text input field.
- At the bottom, there are two checkboxes:
  - Use small image?
  - Use image as entire button?

**Image URL:** The button can be displayed as an image by providing an accessible image URL to display instead of a standard button. This property will override other button properties below, e.g., Back Color.

**Icon Number:** You can select an Icon to display on the button from the [Icon Chooser](#).

**Color:** You can select the text color to display on the button from the [Select Color](#) dialog box.

**Back Color:** You can select the background color to display on the button from the [Select Color](#) dialog box.

**Confirm Text:** You can specify confirmation text that will appear as a confirmation dialog box when the button is clicked.

**onClientClick:** You can specify client-side JavaScript to run on the button's onClientClick event.

**Alt Tag:** You can specify an ALT tag property for accessibility purposes.

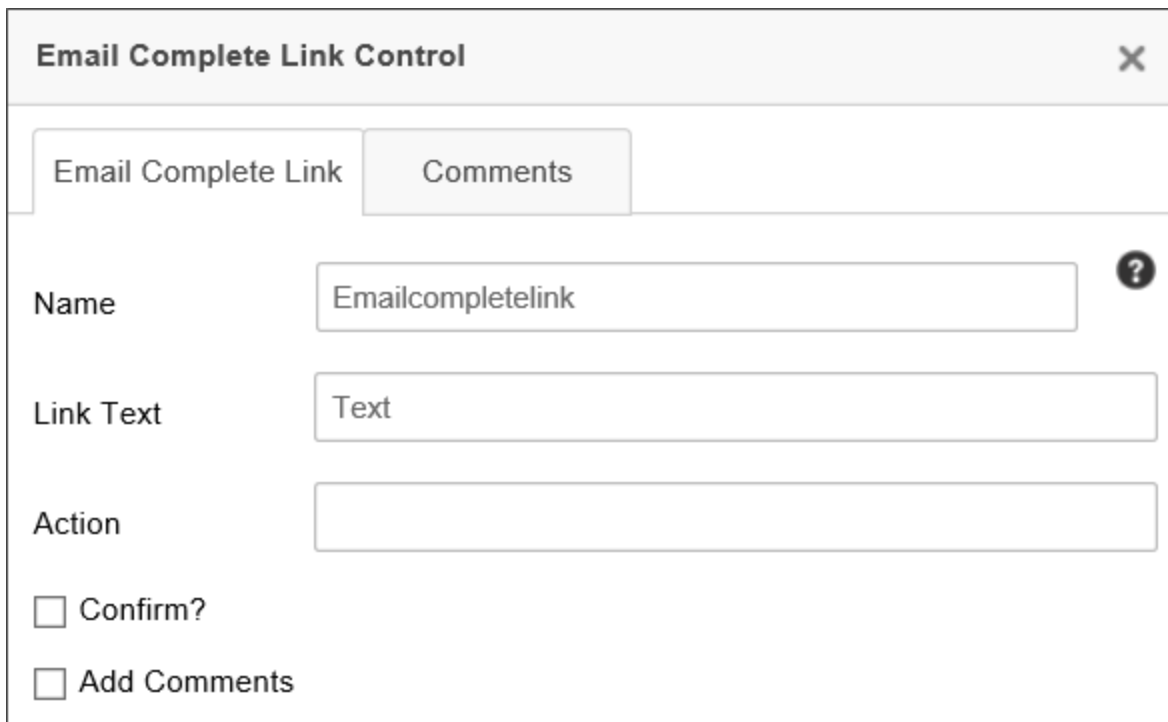
**Access Key:** You can specify the accessibility keys a user can press to activate the button, for accessibility purposes.

**Use Small Image?:** If you supply an image URL, the image can be displayed as an icon for the button by checking this property.

**Use Image as entire button?:** You can use the image as the displayed button, rather than displaying a traditional button on the form.

## Email Complete Link

This control provides a completion link for users to click in an Email Template so that they can complete their assigned task.



The screenshot shows a dialog box titled "Email Complete Link Control" with a close button (X) in the top right corner. The dialog has two tabs: "Email Complete Link" (selected) and "Comments". Below the tabs are several input fields and checkboxes:

- Name:** A text input field containing "Emailcompletelink" with a help icon (?) to its right.
- Link Text:** A text input field containing "Text".
- Action:** An empty text input field.
- Confirm?
- Add Comments

The **Link Text** property enables you to specify the text that will appear for the link, so that the user doesn't simply see the complicated URL that sends them to Process Director to complete their task.

The **Action** property enables you to specify the task action you want the user to perform. This action **MUST** match a result specified in the **Results** tab of the Timeline activity.

For more information on this control, see the section of this documentation entitled [Complete Links in an Email](#).

## Re-Authenticate

For any Timeline Activity where the **Re-Authenticate when a user completes the task** option is checked, Process Director will add a user login to the Form. The **Re-Authenticate** control enables you to choose the location where you'd like the login box to appear on the Form, rather than allowing Process Director to

insert it into the default location.

There are no configurable properties, other than the **Name**, for this control.

## Captcha

The **Captcha** control enables you to verify that a human submitted the form by placing a control that can't be manipulated by robots or scripts on a form. The **Captcha** requires that the person enter an alpha-numeric code that appears in an image that is generated by the control. If the image doesn't match the text entered by the user, the form can't be submitted. When the form is displayed, and the user can't decipher the **Captcha** image, the user can regenerate a new image by clicking the **Generate New Image** link.

**!** To enable the Captcha functionality, you must set the Captcha field as required in the field's properties on the Form Controls tab of the Form Definition. When you do so, the user must enter text that matches the text displayed in the Captcha image, or the form won't validate and submit.

**CAPTCHA Control**
✕

CAPTCHA

Formatting

Comments

**Name**  ?

**Prompt**

**Text Color**

**Background Color**

**Difficulty**  ▼

The **Prompt** property enables you to specify the prompt text to instruct the user how to fill out the Captcha.

The **Text Color** property specifies the text color to be used for the text in the Captcha image, while the **Background Color** specifies the background color of the image. You can use any HTML named color or HTML hexadecimal color representation for these values.

The **Difficulty** property specifies how easy the Captcha image will be to decipher. The following properties are available, along with a sample image of how each level of difficulty is represented in the Captcha.

DIFFICULTY	Example
None	 <p data-bbox="537 394 857 436"><a href="#">Generate New Image</a></p> <input data-bbox="537 464 932 527" type="text" value="Enter the text above."/>
Low	 <p data-bbox="537 716 857 758"><a href="#">Generate New Image</a></p> <input data-bbox="537 785 932 848" type="text" value="Enter the text above."/>
Medium	 <p data-bbox="537 1037 857 1079"><a href="#">Generate New Image</a></p> <input data-bbox="537 1106 932 1169" type="text" value="Enter the text above."/>
High	 <p data-bbox="537 1358 857 1400"><a href="#">Generate New Image</a></p> <input data-bbox="537 1428 932 1491" type="text" value="Enter the text above."/>
Extreme	 <p data-bbox="537 1680 857 1722"><a href="#">Generate New Image</a></p> <input data-bbox="537 1749 932 1812" type="text" value="Enter the text above."/>

### *Other Control Tools*

You can view the documentation for all tools available in the Online Form Designer by using the Table of Contents on the upper right corner of the page, or by clicking one of the links below.

**Basic Controls**: The most commonly-used form design tools.

**Input Controls**: Controls that are commonly used to collect data, but are a bit less widely used than the basic controls.

**Other Input Controls**: Additional Input controls, consisting mainly of the different content picker controls.

**Other Controls**: Controls that perform miscellaneous tasks like adding HTML content, or labels.

**Layout**: Controls that are used to govern the control layout for the template, such as tabs and sections.

**Responsive Layout**: Controls that implement Bootstrap form layout objects.

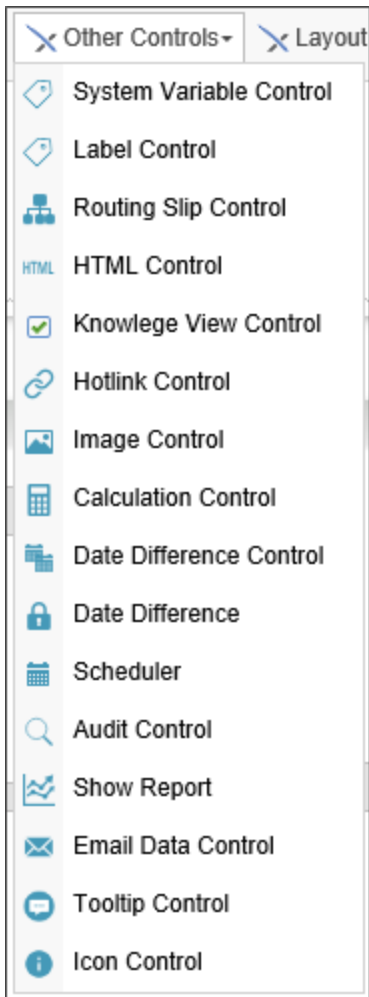
**Arrays**: Controls that enable to you create and control arrays on the Form.

**Attachments**: Controls that enable you to add and show attachments, such as documents or images, to the Form.

**Data List (v6.0.100 and higher)**: Controls that enable the display of tabular data on a Form.

**Form Control Tags**: System Variables used to add controls to a Form, instead of using the UI controls.

## Other Controls



The following controls are available from the **Other Controls** control menu dropdown in the online Form Designer.

### System Variable Control

The **System Variable** control enables System Variables to be displayed on a Form. The System Variable's text representation (e.g., {CURR\_DATE}) will be replaced by the actual value of the specified system variable when the form is displayed to the user. This control can be used to return the value of any evaluable system variable when the form is displayed at run-time.

In addition, this control can be used to encapsulate a [Form control tag](#) by placing the tag syntax in the **System Variable** property box (e.g., {DateTime:ControlName, Interval=15, StartTime="09:00 AM", EndTime="05:00 PM"}), since Form control tags are just System Variables that have a graphical representation.

The screenshot shows a dialog box titled "System Variable Control" with a close button (X) in the top right corner. The dialog has two tabs: "SysVar" (selected) and "Comments". Below the tabs, there are two input fields. The first is labeled "System Variable" and has a question mark icon to its right. The second is labeled "Label".

The **System Variable** property accepts the syntax of the system variable, complete with Curly Brackets, e.g., {CURR\_USER}.

The **Label** property is the text of the placeholder label that will appear when you place the control on the Form page.

### Label Control

The **Label** control adds a piece of text to the Form that can be dynamically changed by accessing it via the control name.

### Label Control ✕

Label	Comments
Name	<input type="text" value="Label63"/> <span>?</span>
Text	<input type="text" value="Text"/>
CSS Class	<input type="text"/>
Style	<input type="text"/>
Alt Tag	<input type="text"/>
Associated Control ID	<input type="text"/>
Display	<input type="text" value="Always"/> <span>▼</span>

The following properties are configurable:

**Name:** The name of the control.

**Text:** The text that will be displayed to Form users.

**CSS Class:** The name of the CSS class, if any, that exist in a custom CSS stylesheet for the Form.

**Style:** A text box into which you can configure a style for the control, using regular CSS syntax.

**Associated Control ID:** The **Name** of the input control to which this **Label** is associated for accessibility compliance.

**Display:** A dropdown that enables you to configure the viewports for which the **Label** should display. The available options are:

- Always: The control will always be displayed on every device.
- Small Device: The control will only be displayed on small devices, e.g., devices with a screen width of less than approx 720px.
- Large Device: The control will only be displayed on large devices, e.g., devices with a screen width of






- more than approx 720px.
- Array Column: The control will be displayed as the column label in an array.

**i** Labels used for accessibility purposes must use the Associated Control ID property to specify the Name property of the data entry control that should be associated with the Label. At run-time, the Label will be associated with the specified data entry control for accessibility compliance.

## Routing Slip

The **Routing Slip** shows the path that a Form has taken through a process. An optional signature image file can be associated with a User ID causing it to be displayed next to their name in the **Routing Slip** when they complete their task. You can upload a signature for each user in the User Administration section of the system. It is recommended that a GIF or PNG file be used with a transparent background so the image can be displayed on web pages with different color backgrounds. If a signature file is found, it is automatically displayed anytime the **Routing Slip** is viewed. Below is an example of how an actual **Routing Slip** will appear on a working form.

Routing Slip						
Participants	Signature	Completed	Status	Result	Comments	
Initiator						
Ron Harris		8/25/2014	Completed			
Approval Required 8/25/2014 3:25 PM						
Diana Stuart		8/25/2014	Completed	✓ Approve	Looks great to me, can't have too many batteries	
Department Head Approval 8/25/2014 3:26 PM						
Barb Stanley		8/29/2014	Completed	✓ Approve	I approve	
Buyer Processing 8/29/2014 8:32 AM						
Rick Rob		-	Active			

To add a user's signature to the database, see the [User Administration](#) documentation. At design time, the **Routing Slip** will be represented as an image of a sample **Routing Slip** when you are building the Form. When an Activity/Step is completed by an automated result condition based on "First Result Met", the **Routing Slip** will include that result name in for each of the users that did not complete the task. It will still identify them as being not required to complete the Activity/Step, but the result will be indicated. There are two primary configuration tabs for the **Routing Slip**.

## Routing Slip Control Tab

Routing Slip Control		
Routing Slip Control	Control Properties	Comments
Name	<input type="text" value="Routingslip"/>	?
Timeline Name	<input type="text"/>	
Activity Name	<input type="text"/>	
Workflow Name	<input type="text"/>	
Step Name	<input type="text"/>	
Sub Task Name	<input type="text"/>	
Only Result	<input type="text"/>	

The following properties are available for configuration on this tab.

**Name:** The name of the control. This property must conform to the regular naming conventions for any Form field.

**Timeline Name:** Entering the name of a Process Timeline in this property will restrict the results to the specified Process Timeline.

**Activity Name:** Entering the name of a Process Timeline Activity in this property will restrict the results to the specified Activity.

**Workflow Name:** Entering the name of a Workflow in this property will restrict the results to the specified Workflow.

**Sub Task Name:** Entering the name of a SubTask in this property will restrict the results to the specified Sub Task. Sub Task names are configured in the properties for User Timeline Activity types.

**Only Result:** If you specify text in the **Only Result** textbox, then only steps whose results match the specified text will display on the Routing Slip.

## Control Properties Tab

Routing Slip Control		
Routing Slip Control	Control Properties	Comments
<input type="checkbox"/> Most Recent Instance	<input checked="" type="checkbox"/> Show Signatures	<input checked="" type="checkbox"/> Show Comments
<input checked="" type="checkbox"/> Show Running	<input checked="" type="checkbox"/> Show Completed	<input checked="" type="checkbox"/> Show Result
<input type="checkbox"/> Show Pending	<input checked="" type="checkbox"/> Show Cancelled	<input checked="" type="checkbox"/> Show Completed On
<input checked="" type="checkbox"/> Show Reassigned	<input checked="" type="checkbox"/> Show Timed Out	<input checked="" type="checkbox"/> Show Participants
<input type="checkbox"/> Use Date Time	<input checked="" type="checkbox"/> Show Status	<input checked="" type="checkbox"/> Show Step
<input checked="" type="checkbox"/> Show Header	<input type="checkbox"/> Active Step Only	<input type="checkbox"/> Active Activity Only
<input checked="" type="checkbox"/> Show Initiator	<input checked="" type="checkbox"/> Show Sub Processes	<input checked="" type="checkbox"/> Show Parent Processes
<input type="checkbox"/> Show Pictures	<input checked="" type="checkbox"/> Show Not Needed	<input type="checkbox"/> Show Most Recent At Top
<input checked="" type="checkbox"/> Show Notify Tasks	<input type="checkbox"/> Show Related Processes	<input type="checkbox"/> Show Wait Tasks
<input type="checkbox"/> Show Predicted		

A **Routing Slip** can be configured to show a number of different information fields, all of which can be configured on this tab. The default configuration is shown above. For most use cases, the default configuration displays what you'd like to see, so changing the default configuration will probably be fairly rare. The following properties may be configured.

For Process Director v5.26 and higher, delegation that occurs from a disabled user will be noted on the **Routing Slip**, in addition to the normal delegation information.

**Most Recent Instance:** Generally only applicable to processes that use iterative steps, checking this property will display only the most recent iteration's results. The default behavior is to show ALL iteration results.

**Show Running:** Checking the box will display the currently running activities in the **Routing Slip**.

**Show Pending:** Checking the box will display the currently Pending activities in the **Routing Slip**.

**Show Reassigned:** Checking the box will display any task reassignments in the **Routing Slip**.

**Use Date Time:** Checking the box will display the time an event occurred, in addition to the date.

**Show Header:** Checking the box will display the **Routing Slip** header.

**Show Initiator:** Checking the box will display the name of the Process Initiator.

**Show Pictures:** Checking the box will display the pictures of the participating users. This will, of course, require that the users have a picture image uploaded to their user account in Process Director.

**Show Notify Tasks:** Checking the box will display any notification tasks, in addition to the user tasks that display be default.

**Show Predicted:** Checking the box will display the predicted path the process will take. For processes that rely on *ad hoc* decisions for task assignments or branching, this may not be reliably accurate, for obvious reasons. For more formalized process structures however, this property will enable the **Routing Slip** to show the predicted paths and predicted start dates of the subsequent activities, based on the past performance of the process.

**Show Signatures:** Checking the box will display the signature images of the participating users. This will, of course, require that the users have a signature image uploaded to their user account in Process Director.

**Show Completed:** Checking the box will display all completed activities.

**Show Canceled:** Checking the box will display all canceled activities.

**Show Timed Out:** Checking the box will display all activities that were ended by timing out.

**Show Status:** Checking the box will display the current status of all activities.

**Active Step Only:** Checking the box will display only information about the currently active Workflow Step.

**Show Sub Processes:** Checking the box will display the activities for any subprocesses to which the current process is the parent process.

**Show Not Needed:** Checking the box will display all activities that were skipped as "Not Needed".

**Show Related Processes:** Checking the box will display activities for any process that is called by the current process, and which has run, or is running, asynchronously.

**Show Comments:** Checking the box will display all completion comments for the activities.

**Show Result:** Checking the box will display all Activity results.

**Show Completed On:** Checking the box will display the completion date for all completed activities.

**Show Participants:** Checking the box will display all Activity participants.

**Show Step:** Checking the box will display all Workflow Steps.

**Active Activity Only:** Checking the box will display only the currently active Timeline Activities.

**Show Parent Processes:** Checking the box will display all activities for the process which called the current process as a subprocess.

**Show Most Recent At Top:** Checking the box will order the activities from the most recently completed to the oldest completed Activity. This is the reverse of the normal order.

**Show Wait Tasks:** Checking the box will display all Wait tasks, in addition to the User activities.

## HTML Code

The **HTML Code** control allows you to insert an HTML code snippet into the Form definition. You can include System Variables in the code snippet, and Process Director will parse them properly. The control has a character limit of 5,500 characters, so if you have a lengthy HTML snippet, you should break it up across multiple HTML controls to ensure that you don't exceed the character limit.

**!** If your inserted HTML includes { or } characters, such as used in JavaScript code, you must leave the control's Name property blank. Otherwise the "{" and "}" characters will be interpreted as System Variables.

### JavaScript

In addition to normal HTML, you can include JavaScript. So, to send an alert that greets the user, you could say:

```
<script>
  Alert("Hello {CURR_USER}");
</script>
```

If you create a JavaScript function called "bpUserJavaScript" and put it on the form inside an HTML control - it will run on the initial Form Load and all Postback events, enabling you to insert JavaScript into these events.

```
<script>
  function bpUserJavaScript()
  {
    //Script code for Load and Postback goes here.
  }
</script>
```

```
}  
</script>
```

This feature enables you to insert complicated client-side JavaScript that sets a control's state on Postback. In the example below, a notional Form has a **Checkbox** control that is set as an event field to fire the Postback. On Postback, the `bpUserJavaScript` function runs to control the state of the **Checkbox**.

```
<script>  
  var SaveClick;  
  
  function bpUserJavaScript()  
  {  
    SaveClick = CurrentForm.FormControls["CheckBox2"].onclick;  
    CurrentForm.FormControls["CheckBox2"].onclick = NewClick;  
  }  
  
  function NewClick()  
  {  
    if (confirm("Are you sure?"))  
    {  
      if (typeof(SaveClick) != "undefined" && SaveClick)  
      {  
        SaveClick();  
      }  
    }  
    else  
    {  
      CurrentForm.FormControls["CheckBox2"].checked =  
        !CurrentForm.FormControls["CheckBox2"].checked;  
    }  
  }  
</script>
```

## Server Side Controls

You can also place ASP.NET server -side controls in this tag. For example:

```
<asp:TextBox runat="server" ID="bpLogixTextBox" />
```

Server-side ASP.NET controls also act as normal BP Logix form fields, allowing you to reference them with System Variables and through Process Director. You can place an HTML control on a Form, leave the Name property blank, then place the ASP.NET control markup in the HTMLString Property. Entering a Name for the HTML control will cause Process Director to treat it like a Process Director Control and override the ASP.NET control. Leaving the Name property blank will cause Process Director to take the HTML markup for the ASP.NET control, and display the ASP.NET control in the Form. By wrapping a native ASP.NET control in an unnamed HTML control, Process Director supports the use of any native ASP.NET control on a Form.

You can stylize your HTML with CSS, either in the tag you want to stylize or in a separate `<style>` tag.

```
<!-- You can do this: -->  
<p style="text-align: center;">Text</p>  
  
<!-- Or this: -->  
<style type="text/css">  
  p{
```

```
        text-align: center;  
    }  
</style>
```

To use the ASP.NET controls, you must have Process Director v3.5 or higher.

### Knowledge View Control

This control enables you to display a Knowledge View within a Form, or within a popup window.

The **Properties** dialog enables you to configure the **Name** of the control, as well the **HTML text** that should be displayed to the user at the top of the Knowledge View, if any. The **HTML Width/Height** use HTML measurements, usually percentages or pixels, to define the size of the control. The **Type** property can be set to "Iframe" or "Pop-Up". If displayed as an HTML Iframe, the Knowledge View will display inline on the Form. If displayed as a Pop-Up, the Knowledge View will be displayed in a pop-up window. You can also send **QueryString Params**, i.e., query string parameters, to the Knowledge View, with each parameter on its own line in the text box.

### KView Control ✕

KView Comments

Name  ?

HTML Text

HTML Height

HTML Width

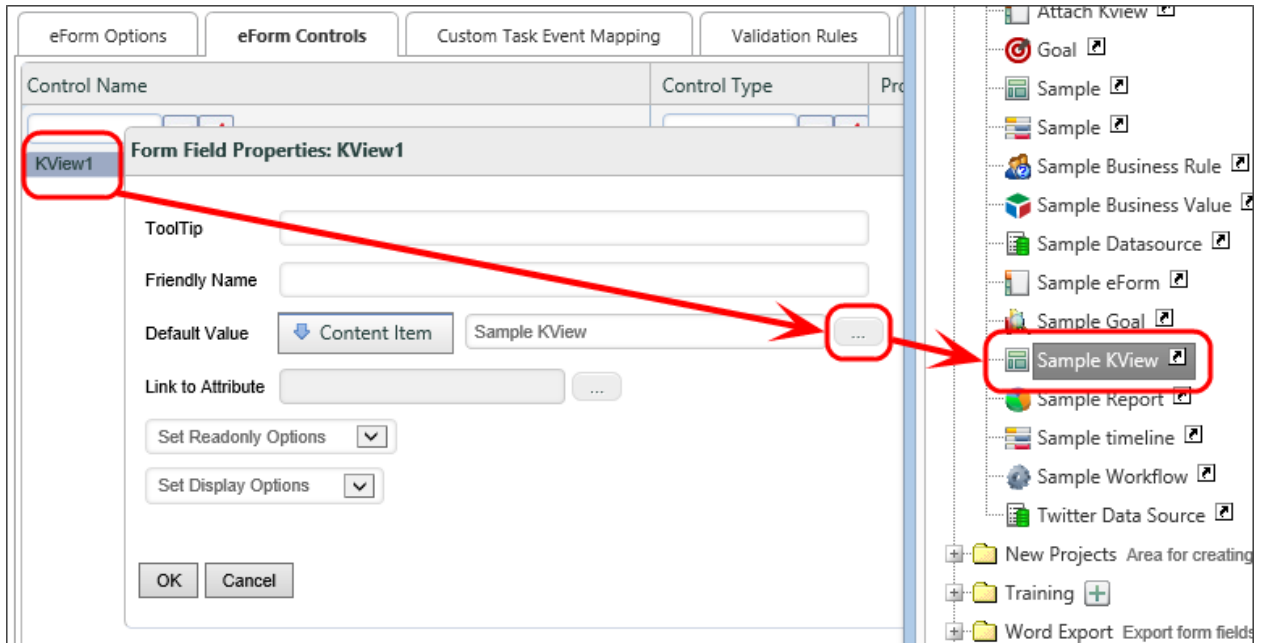
Type  ▼

QueryString Params (Separated by NewLines)

To specify which Knowledge View this control will display, you must configure this control's property on the **Form Controls** tab of the Form Definition by setting the **Default Value** of this control to the content item for the Knowledge View you want displayed.

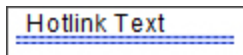
QueryString Parameters can also be set in the **QueryString Params** property, as described in the [KView control tag section of this document](#).





## Hotlink

The **Hotlink** control creates a link to a specified URL. These are often placed in emails to link users to the task they need to complete. As such, one will usually set the **Hotlink**'s URL dynamically using System Variables, e.g., {EMAIL\_URL}.



The following properties are configurable:

The screenshot shows a dialog box titled "Hotlink Control" with a close button (X) in the top right corner. Below the title bar are two tabs: "Hotlink" and "Comments". The "Hotlink" tab is active and contains the following fields:

- Name:** A text input field containing "Hotlink1" and a help icon (question mark in a circle) to its right.
- Text:** An empty text input field.
- URL:** An empty text input field.
- Style:** An empty text input field.
- Open in new window?:** An unchecked checkbox.
- OnClientClick:** An empty text input field.

The **Name** is the name of the control.

The **Text** property sets the text displayed on the form.

The **URL** property sets the URL to which the hotlink control will link.

The **Style** property determines CSS styling for the link.

The **Open in new window** checkbox will, when checked, force the link to open in a new browser tab/window.

The **OnClientClick** property specifies the name of a JavaScript subroutine to be executed when the user clicks on the link. It must be encased in quotation marks.

**i** A built-in CKEditor control, the [Link control \(documentation\)](#) offers additional functionality, and may serve as a better control for some use cases.

## Image Control

This control inserts an image from a specified URL into the Form.

**Image Control**
✕

Image Control
Comments

Name  ?

Image URL

URL

CssClass

Style

HTML Height

HTML Width

Alt Tag


Open hotlink in new window?

The **Image URL** field specifies the URL at which the image can be found. If text is entered in the **URL** field, the image will redirect the user to that URL if the image is clicked on. You can check the checkbox to open the link in a new window. For use of QR Codes, the **Image URL** property will accept the [QR Code system variable](#) to generate the QR code automatically as the image for this control.

The image can also be stylized using CSS, with the **CSS Class** property specifying the name of a CSS class that exists in a custom CSS stylesheet, and the **Style** property containing CSS style directives for solely this control. The **HTML Height** and **Width** can be set in pixels/Percent.

For accessibility, all **Image** controls should configure the **Alt Tag** text to be displayed if the image isn't available, or for users of assistive devices.

For images that use the **URL** property to specify a link to open when the image is clicked, the **Open hot-link in new window** checkbox will, when checked, force the link to open in a new browser tab/window.

 A built-in CKEditor control, the [Image](#) tool ([documentation](#)) offers additional functionality, and may serve as a better control for some use cases. This tool will not work with the QR Code system variable.

## Calculation

The **Calculation** control displays a number that is the result of a calculation involving any combination of the values of Form controls, constants, and the multiplication, division, addition, and subtraction operators (“\*”, “/”, “+”, and “-” respectively).


In the Online Form Designer, the calculate control will appear as a small calculator icon.



At run-time, only the calculation's result will be displayed to the Form user.

## Properties

### Calculation Control ✕

Calculation	Comments
Name	<input type="text" value="Calculation63"/> 
Formula	<input type="text"/>
Format String	<input type="text"/>
Style	<input type="text"/>

**Name:** The name of the control.

**Formula:** The mathematical formula that produces the calculation.

**Format String:** The built-in .NET Format String used to format the result. For the syntax of the available .NET Format Strings, please refer to Microsoft's MSDN documentation topic for [Standard Numeric Format Strings](#).

**Style:** CSS style tags to format the control, e.g., "font-weight: bold;"

The algorithm used in the calculation can incorporate system variables, and usually Form field system variables are used for calculations. For example, to add the value of two form fields, use the following equation, substituting the appropriate control names:

`{FORM:fieldA} + {FORM:fieldB}`

The alternate Form field system variable syntax will also work as expected, e.g.:

`{:fieldA} + {:fieldB}`

In addition, system variable encoding symbols, such as the numeric designator, can also be used, e.g.:

`{#:fieldA} + {#:fieldB}`

## Date Difference

The **Date Difference** control displays the number of days, hours, minutes, etc., between two dates.

There are three attributes that must be provided to enable the control to display the proper value in the Form:

- **Date1**: This is the start date for calculating the date difference.
- **Date2**: This is the end date for calculating the date difference.
- **Type**: This is the time increment to be calculated. This can be any standard .Net time increment, such as "Days" "Hours", etc.

Additionally, the **Format** property will accept any valid .NET formatter to change the display of the result to the specified format.

By default, the **Date Difference** control will display a simple text number reflecting the value of the date difference in the selected **Type** increments.

## Lock Form Control

The **Lock Form** control is used when a form may be opened by more than one user at a time. This control ensures that, when one user opens the form, it is locked so that, if a second user opens the form, the second user can't edit the form until the first user has completed editing. Instead, the second user receives a dialog box notifying them that the form has already been opened for editing by another user. In the Online Form Designer, the control will appear as a padlock icon.



The **Lock Form** control enables the user to see whether a form is locked and, if the control is configured to allow it by checking the **Allow user to unlock form checkbox**, to unlock it.

The properties window of the control enables you to set the **Lock Text** shown to the user when the form is locked. To put the name of the user who locked the control into the **Lock Text**, use the `{FORM_LOCK_USER}` system variable.

In most cases, the Form will remain locked until the locking user closes it via the use of the **Submit**, **Cancel**, or a task result button. The control can also be configured to periodically check to see whether the form has been locked after the number of seconds specified in the **Polling Seconds** property, by checking the **Use Polling to detect when the form is unlocked** checkbox. Checking the box requires that you specify the desired number of seconds in the **Polling Seconds** property. Forms that are closed without the use of a system button can be polled to determine if the Form is still open and, if not, can be unlocked after the appropriate poll result is received.

**Lock Form Control**
✕

Lock Form
Formatting
Comments

Name  ?

Lock Text

Polling Seconds

Allow user to unlock form

Use polling to detect when form unlocked

### Scheduler

Use of the Scheduler control has largely been deprecated in the product. BP Logix recommends using the [Goal object](#) for most scheduled operations that were formerly implemented via the Scheduler.

The **Scheduler** control is part of the Process Director Scheduler Module (PDSM), which is used to launch processes at pre-defined intervals within a specified window of time.

Repeats:  ▼

Repeat Every:  hours

Scheduler Period Begin:

Scheduler Period End:

Process Lead time:  days

Business Days Only

- **Repeats:** Choices are hourly, daily, monthly, or yearly.
- **Repeat Every:** Specify the frequency with which the process will be launched.
- **Starts On:** (Optional) The specified process won't be launched before this date/time.
- **Ends On:** (Optional) The specified process won't be launched after this date/time.
- **Lead Time:** (Optional) Tells the scheduler to launch the process a certain number of days prior to "Starts On" (effectively moves "Starts On" date up the indicated number of days).
- **Business Days Only:** When checked, only launch the process on business days.

Complete details about the scheduler component are available through the [Scheduler Module guide](#).

## Audit Control

The **Audit** control enables the user to view a history of the controls whose values have changed on the Form, when the **Audit Form Changes** checkbox is checked on the Form definition's **Properties** tab.

The screenshot shows the 'Options' tab of a form definition interface. It contains various configuration options for the form, including scheduling, background images, and validation settings. The 'Audit Form field changes' checkbox is highlighted with a red circle.

**Options**

- Automatically start this process after Form is submitted (optional) [Text Field] [...]
- Workflows Associated with Form (optional) [Text Field] [...]
- Timelines Associated with Form (optional) [Text Field] [...]
- Default Groupname for Form Instances [Text Field]
- Form Script (optional) [Text Field] [...]
- Automatically create an instance of this Case (optional) [Text Field] [...]
- Select a Content List background image for this form (optional) [Text Field] [...]
- Select a background image from this list for this form (optional) [Choose background] [Cover] [Opacity Level 1-100 0]
- Form Lock Group (optional) [Text Field]
- Display validation errors with alert boxes
- Do not show this object in 'Items I Can Run' Knowledge Views
- Use Visual Basic for scripts (leave unchecked for C#)
- Audit Form field changes**
- Enable Form Locking
- Display warning if user Cancels form changes
- Show disabled fields as text
- Show only the first validation error (instead of all)
- Hide disabled buttons
- Remove form from existing Case on submission
- Enable Form Data Cache

The **Audit** control's properties can be configured as follows:



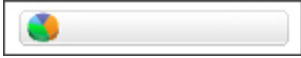
You can configure the control's **Name**, and the **Text** displayed on the audit button.

Using the **Control Name**, **Step Name**, and **Activity Name** fields, you can filter the audit history to display only Form changes relevant to the specified controls, activities, and steps. For instance, if you set the **Activity Name** property to "Step 1", then only auditable activities that occur during the Activity named "Step 1" will appear in the control at run-time. These three fields can accept multiple values by entering a comma-separated list of items. You can also set any of these field properties to a System Variable, instead of manually adding values to these fields. As an example, you can use a Business Rule to provide these values, and enter the System variable name for a Business Rule in each field, e.g., {RULE:BusinessRuleName}. You could then set the Business Rule to return the step completion date, so that you only capture changes made after that point in the process. Using a Business Rule enables you to change the filter values by editing the Business Rule, instead of re-opening the Form template and modifying the template.

If you check the **Show field changes in tooltip...** check box, the **Audit** control button won't be displayed at run-time. Instead, audit information will be displayed in a tooltip for the Form fields when the user hovers the mouse over the relevant fields.

### Show Report Control

The **Show Report** control displays a report in a Form. The **Show Report** control can display a report in a variety of ways, configurable in the control's properties.



The properties window allows the user to configure the **Name** of the control, the **Text** the **Show Report** button displays, and the **HTML Height/Width** of the report, using HTML measurements, i.e., pixels or percentages. It also enables the user to pass **QueryString Params** (parameters) to the report, configuring the data it returns. Separate parameters should be separated by new lines, using the [variable name] = [value] format to specify parameters and their values.

**Show Report Control**
✕

Show Report

Comments

Name  ?

Text

HTML Height

HTML Width

Type  ▼

Image URL

QueryString Params (Separated by NewLines)

The **Type** dropdown allows the report to be displayed in the Form in three different ways. The report can be displayed in an IFrame in the Form, as a popup, or as an image embedded in the Form by setting the **Type** property. The image is static, but the data it returns can still be configured using QueryString parameters.

### Email Data

The **Email Data** control is used in a email notification template to specify the configuration of the email that is sent via task notifications. In addition to the **Name** property that is common to all controls, the properties for this control are configured on the interface tabs shown below.

**i** For users of Process Director v5.27 and higher, Process Director enables the use of address strings with a Display Name component (e.g. "Test User <test@example.com>").

## Email Data Control Tab

The screenshot shows a window titled "Email Data Control" with a close button (X) in the top right corner. Below the title bar are three tabs: "Email Data Control" (selected), "Additional Properties", and "Comments". The main area contains the following fields:

- Name:** Text input field containing "Emaildata1" with a help icon (?) to its right.
- Subject:** Empty text input field.
- From Email:** Empty text input field.
- From Display:** Empty text input field.
- Group Name:** Empty text input field.
- Locale:** Empty text input field.
- CC Email:** Empty text input field.
- BCC Email:** Empty text input field.
- Priority:** Dropdown menu with "Normal" selected and a downward arrow.

PROPERTY	DESCRIPTION
Subject	Subject for email
From Email	The email address from whom the email is sent. Usually, this property is configured only when you wish to use a specific email address for this specific control. In most cases, you won't configure this property. By default, notification email "From" addresses will be set by the first value that isn't empty from the following list: <ul style="list-style-type: none"><li>• Value configured in <a href="#">IT Admin</a> &gt; <a href="#">Installation settings</a> &gt; <a href="#">Global Variables</a></li><li>• Configured in email or Form</li><li>• If Canceled, Administrator email that performed operation</li></ul>

PROPERTY	DESCRIPTION
	<ul style="list-style-type: none"> <li>Process Timeline Initiator</li> <li>The <b>Registered Email Address</b> setting in <a href="#">IT Admin &gt; Installation settings &gt; Properties</a></li> <li>process-director@bplogix.com</li> </ul>
<b>From Display</b>	When configuring a <b>From Email</b> address, this property enables you to specify a display name for the email address, e.g., the name of a person, instead of the email address alone.
<b>Group Name</b>	Only attach documents in the specified attachment group or groups. If you don't specify the <b>Group Name</b> of objects to include in the email, it will send <b>all</b> attachments to the email recipient. Multiple groups can be added in a comma-separated list, e.g., "Group1, Group2, Group3".
<b>Locale</b>	<p>An optional string property that accepts a valid .NET locale string, e.g., "en-US" for US English. This value can also be passed through a form field variable, e.g., <code>{:Locale}</code>, that passes the appropriate locale string.</p> <p>Normally when an email is sent, it will set the locale to that of the target user. If the user is an anonymous user, the locale will be set to the default locale of the server. If the locale is specified in this property, however, the default behavior will be overridden by this property setting.</p>
<b>CC Email</b>	A list of comma separated or semi-colon separated email address that will be CC'd on the email.
<b>BCC Email</b>	A list of comma separated or semi-colon separated email addresses that will be BCC'd on the email.
<b>Priority</b>	<p>Sets the priority of the email. Supports the following values:</p> <ul style="list-style-type: none"> <li>Low</li> <li>Normal</li> <li>High</li> </ul>

### Additional Properties Tab

The screenshot shows a dialog box titled "Email Data Control" with a close button (X) in the top right corner. Below the title bar are three tabs: "Email Data Control", "Additional Properties", and "Comments". The "Additional Properties" tab is selected and highlighted with a blue border. Inside this tab, there are four checkboxes, each followed by a label:
 

- Add Process Attachments to Email? (using optional Group Name filter)
- Add eForm Attachments to Email? (using optional Group Name filter)
- Cancel Email
- Plain Text?

PROPERTY	DESCRIPTION	OPTION
Add Process Attachments to Email	When checked, this property enables you to send Process attachments as attachments in the email. You can specify what attachments to send by using the Group Name property, as mentioned above.	Group Name
Add Form Attachments to Email	When checked, this property enables you to send Form attachments as an attachments in the email. You can specify what attachments to send by using the Group Name property, as mentioned above.	Group Name
Cancel Email	Check this box when an email doesn't need to be sent out, based on a condition.	
Plain Text	Checking the property will strip the email of HTML information and transmit it using plain text. This is used sometimes to make emails more legible on PDAs or other operating systems that receive email from Exchange servers, but which can't parse HTML emails.	

For information on this control's usage in an email template, see the [EmailData Control section](#) of the Email Templates documentation.

### Tooltip

The **Tooltip** control will place an icon on the Form page that, when moused over, will display text that you configure in the settings for the control.


**Tooltip Button** ✕

Tooltip Button

Comments

Name  ?

Text

The text you place in the **Text** property will appear as a tooltip when the user places the mouse over the tooltip icon () that appears on the page.

### Icon

This control will place an icon on the Form page.

The screenshot shows a dialog box titled "Icon Control" with a close button (X) in the top right corner. The dialog is divided into two tabs: "Icon" and "Comments". The "Icon" tab is currently selected. Below the tabs, there are several input fields:

- Name:** A text box containing "Icon8" and a help icon (question mark in a circle) to its right.
- Icon Number:** An empty text box.
- Color:** An empty text box.
- Back Color:** An empty text box.
- Css Class:** An empty text box.
- Size:** An empty text box.
- Tooltip:** An empty text box.

Use the **Icon Number** property to specify the icon number to display on the Form.

Other properties, such as the icon's **Color** or **Back Color** can also be configured to specify the icon's fore color or back color, respectively.

The **CSS Class** property enables you to specify a CSS class to use for the icon if you've implemented a custom style sheet.

The **Size** property will specify the size, in pixels, to display the icon. Icons are square, so a single number will specify both the horizontal and vertical dimensions of the icon.

The **Tooltip** property specifies the tooltip text to display when the user mouses over the icon.

### ***Other Control Tools***

You can view the documentation for all tools available in the Online Form Designer by using the Table of Contents on the upper right corner of the page, or by clicking one of the links below.

**Basic Controls:** The most commonly-used form design tools.

**Input Controls:** Controls that are commonly used to collect data, but are a bit less widely used than the basic controls.

**Other Input Controls:** Additional Input controls, consisting mainly of the different content picker controls.

**Actions:** These controls enable you to control form actions, like placing buttons, or choosing objects via a picker,

**Layout:** Controls that are used to govern the control layout for the template, such as tabs and sections.

**Responsive Layout:** Controls that implement Bootstrap form layout objects.

**Arrays:** Controls that enable to you create and control arrays on the Form.

**Attachments:** Controls that enable you to add and show attachments, such as documents or images, to the Form.

**Data List (v6.0.100 and higher):** Controls that enable the display of tabular data on a Form.

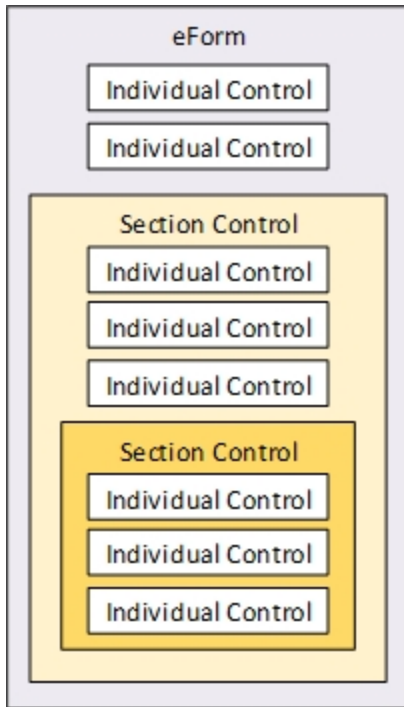
**Form Control Tags:** System Variables used to add controls to a Form, instead of using the UI controls.

## Layout Controls



A Layout control is a type of control that you use to organize, show, or hide controls on a form. The most commonly used Layout control is the **Section** control. Every control that is placed inside a **Section** in the online form designer becomes a child control of that **Section**. You can also nest **Sections**, meaning that you can place a **Section** inside of a **Section**.





There are two kinds of section controls: the **Section** and **Embedded Section**. Both section types are used to gather various controls on a Form into one group, making it easier to set properties common throughout the group. The **Section** is visible to the user, and is collapsible and expandable. The **Embedded Section** isn't visible to the user.

All section controls automatically include HTML anchors using the syntax:

```
<a name="SectionName"></a>
```

The anchor has the same name as that of the **Section** control. Form designers can give their users the ability to scroll down to any section by providing a link (using the Hotlink control) with the URL:

**#SectionName**

Simply replace "SectionName" with the actual name of the **Section** control. You may find the ability to link to document sections useful when designing long forms.

### Container

For Process Director v5.44.1100 and higher, the **Container** control enables you to place a fixed, non-scrolling content area on a page. The control is purely a visual control and has no data of its own. Instead, you can place tables, text, Form controls, or other contents inside the **Container** to control how they respond when the page scrolls vertically.

Container Control		×
Container Control	Formatting	Comments
Name	<input type="text" value="Container1"/>	?
Sticky Top	<input type="text" value="0"/>	

The **Container** control has a single property, **Sticky Top**, which enables you set the measurement, in pixels, from the top of the screen, at which the container control will stop scrolling. For example, if you set **Sticky Top** at "0", contents inside this control will scroll normally on the page, until the top of the **Container** control reaches the top of the screen. At that point, the container will remain fixed, and all other page objects will continue to scroll. The scrolling content will scroll behind the container.

## Section

See the [Section control's documentation](#) in the Basic Controls topic for detailed information about this control.

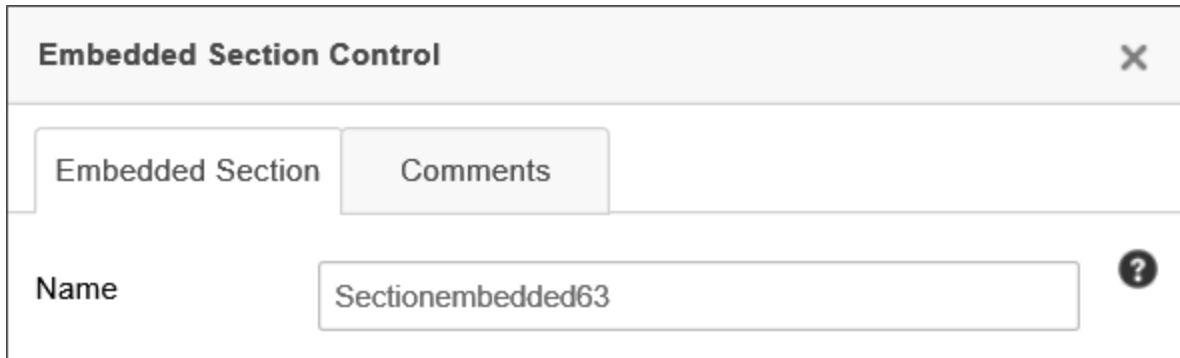
## Embedded Section / Embedded Section End

**Embedded Sections** are represented in the Online Form Designer as large square brackets, though they're invisible to the Form's end user at run-time.

[Section data goes between the bracket-like section controls.]

Every **Embedded Section** must have an **Embedded Section End** control. Every control that is placed between the section and section end controls will automatically be part of the embedded section.

Embedded sections are useful for conditionally showing and hiding items on a Form. For instance, you can use embedded sections to surround a table row you'd like to hide, and set the conditions under which it should be hidden by configuring the Form properties on Process Director. The **Name** property is the only configurable property for the embedded section.




When using an embedded section in a table, you should place the **Embedded Section** control at the end of the row *before* the row that you want to hide, and place the **Embedded Section End** control at the end of the row you want to hide.

Text Box:	
Date Picker:	 [
Hidden Field:	]
Text Box:	

Save the Form template then navigate to the **Form Controls** tab of the Form definition. Find the **Embedded Section** surrounding the row that you want to hide, and click on the **Edit** link. In the **Set Visibility Options** dropdown, select either “Field is Hidden When” or “Field is Visible When”, and use the **Condition Builder** by clicking the **Click to Create Condition** link to set the conditions under which the row will be hidden.

For Process Director v5.26 and higher, embedded sections will be specifically identified as such on the **Form Controls** tab of the Form Definition.

### Tab Strip/Tab Content Controls

 A display issue may occur for TabStrip controls placed inside a table. To resolve this issue, add the following CSS style to the Style property of the table:  
`table-layout: fixed;`

This control enables you to separate form content into different tabs, thus conserving page space. Each tab can be viewed individually. To declare the start of a tab strip, use the **Tab Strip** control.



The **Properties** dialog box enables you to configure the number, text, and IDs of tabs. By default, tabs are arrayed horizontally on the page. You can, however, arrange the tabs vertically by checking the **Use Vertical Tabs** property.

### Tab Strip Control ✕

**Tab Strip** **Comments**

Name  ?

Use Vertical Tabs?

Accessibility Keys

	ID	Tab Text (optional)
Tab 1	<input type="text"/>	<input type="text"/>
Tab 2	<input type="text"/>	<input type="text"/>
Tab 3	<input type="text"/>	<input type="text"/>
Tab 4	<input type="text"/>	<input type="text"/>
Tab 5	<input type="text"/>	<input type="text"/>
Tab 6	<input type="text"/>	<input type="text"/>
Tab 7	<input type="text"/>	<input type="text"/>

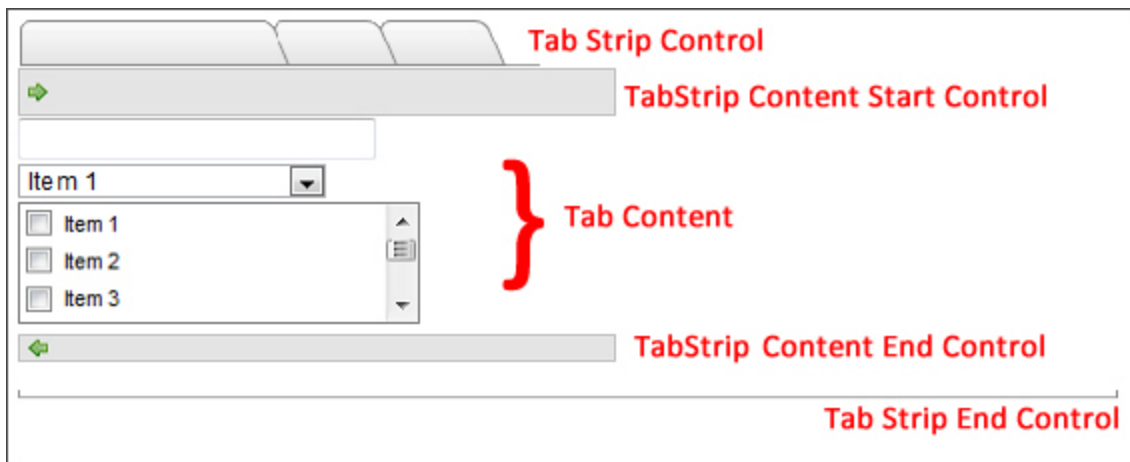
The **ID** property for each tab is the name Process Director will use for the tab to identify it. The **Tab Text** is the text that the user will see on each tab at run-time.

For Process Director v5.43 and higher, two new accessibility-related properties were added to the **TabStrip** control, **CommandKey** and **FocusKey**. These properties consist of dropdown controls that enable you to choose a set of command keys and/or regular keyboard keys to send the focus to the **TabStrip** control when the configured keys are pressed. These properties were added to satisfy the optional **AccessKeys** feature of accessibility. Keep in mind that each control needs to have a unique **AccessKey**, and applying the same configuration to two different controls on the same page may prevent the feature from working as expected. Also, different operating systems and browsers will implement the use of **AccessKeys** differently. Similarly, screen reader implementation of **AccessKeys** will depend largely upon the browser being used, and screen readers additionally have a larger set of available **AccessKeys** that may override the ones configured on the Form.


Everything inside the **Tab Strip** control must be part of a tab. To declare the start of a tab, use the **Tab Content** control.



Form content goes between the **Tab Content** and **Tab Content End** controls. These two controls define the start and end of a tab.



The **ID** you configure for each tab on the **Tab Strip** control, configured in the properties, must match the **ID** property of the tabs defined in the **Tab Content** control's properties.

 IDs are case sensitive, so the case must match between the IDs in the **TabStrip** and **TabStripContent** controls.



**TabStrip** and **TabStrip Content** controls are both ended with an "End" control, just as sections are.

If a **TabStrip**'s first tab is hidden, the Form will select the first visible tab by default.

You can set the focus to a specific tab in the **TabStrip** control by setting the value of the **TabStrip** to the **ID** of the tab you want to display.

### Comment Start / Comment End

The **Comment Start** and **Comment End** controls are used to define an area in the Form to contain a comment. A commented area will only display when building the Form in Online Form Designer. Any controls or text put inside the commented area won't affect or be displayed in the Form once it is deployed.

```
<!-- This is commented text. -->
```

The primary use for these controls is to provide comments for other designers.

### Form Error String Location

This Form control is used to identify the area(s) where error messages are displayed on the Form. If this control isn't present on a Form, then the error messages are placed at the top and bottom of the form. The placeholder looks like this:

```
--FORM ERROR STRING--
```

The only configurable property for this control is the **Name**.

### Form Info String Location

This Form control is used to identify the area(s) where informational messages are displayed on the Form. If this control isn't present on a Form, then the informational messages are placed at the top and bottom of the form. The placeholder looks like this:

```
--FORM INFO STRING--
```

The only configurable property for this control is the **Name**.

### *Other Control Tools*

You can view the documentation for all tools available in the Online Form Designer by using the Table of Contents on the upper right corner of the page, or by clicking one of the links below.

**Basic Controls**: The most commonly-used form design tools.

**Input Controls**: Controls that are commonly used to collect data, but are a bit less widely used than the basic controls.

**Other Input Controls**: Additional Input controls, consisting mainly of the different content picker controls.

**Actions**: These controls enable you to control form actions, like placing buttons, or choosing objects via a picker,

**Other Controls**: Controls that perform miscellaneous tasks like adding HTML content, or labels.

**Responsive Layout**: Controls that implement Bootstrap form layout objects.

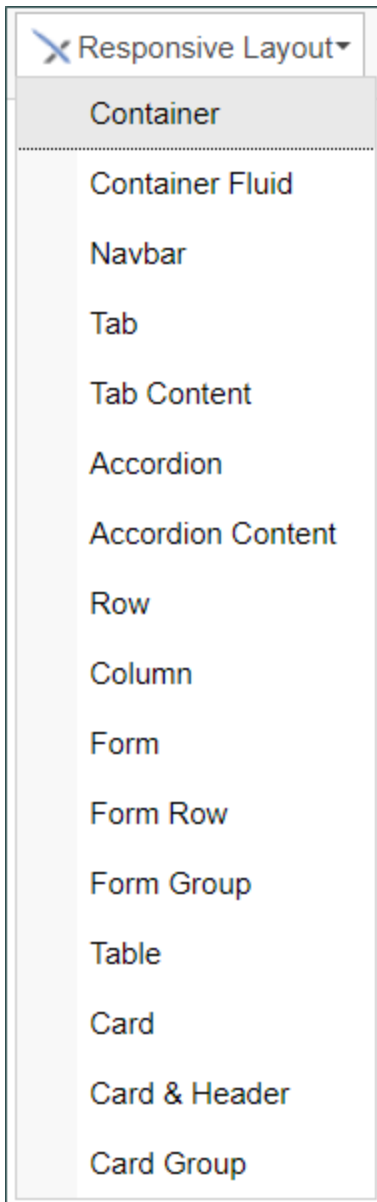
**Arrays**: Controls that enable to you create and control arrays on the Form.

**Attachments**: Controls that enable you to add and show attachments, such as documents or images, to the Form.

**Data List (v6.0.100 and higher)**: Controls that enable the display of tabular data on a Form.

**Form Control Tags**: System Variables used to add controls to a Form, instead of using the UI controls.

## Responsive Layout Controls




Users of Process Director v5.35 and higher have a **Responsive Layout** menu in the Online Form Designer that implements Bootstrap integration with Process Director forms. [Bootstrap](#) is a free, open-source CSS framework used to create responsive, mobile-friendly layouts. Process Director integrates the CSS objects in Bootstrap as form controls that can be used to create responsive forms.

Use of these controls, of course, implies some basic knowledge of what the Bootstrap objects are, and how they work, in order to create forms using the appropriate Bootstrap objects. The documentation on this page doesn't provide general information about how to implement Bootstrap based forms, but does provide information about the control properties available in the Process Director implementation. Otherwise, we assume that you already have the appropriate general Bootstrap knowledge required to use the available objects correctly. If you aren't familiar with the use of Bootstrap, we don't recommend using it to create production forms until you have the appropriate knowledge to do so.



If you'd like information about Bootstrap objects, and how they are implemented, please refer to the [Bootstrap documentation](#) at the Get Bootstrap web site.

 Use of the Responsive Layout controls requires that the [fIncludeBootstrap](#) custom variable be set to True. If this variable isn't set to True, the Responsive Layout controls menu won't appear in the UI of the Online Form Designer.

### *Control Settings*

Unlike other controls, the Responsive Layout controls, when placed on a page, don't have **Properties** dialog boxes. Instead, editing the controls may require using the Source view of the designer to manually edit the HTML that is generated. Primarily, this is because the controls are responsive, so inserting a layout control on the page will include specific settings. For instance the container controls are the fundamental containers for all other Responsive Layout controls. The standard **Container** control automatically creates a container that is fixed-width, but changes at several breakpoints that depend on the size of the viewport. The **Fluid Container** control, on the other hand, is always set to 100% of the viewport. Since all other Responsive Layout controls are placed inside one of the container controls, the choice of container control will affect the width and layout of all other controls placed inside the container.

Also, keep in mind that the Responsive Layout controls are purely HTML/CSS entities. They aren't Process Director controls. You can't, therefore, assign visibility or enabling properties to the controls, in the same way that you can to a standard **TabContent** or **Section** layout control.

### *Responsive Layout Usage Tips*

All forms created with the Responsive Layout controls must have a **Container** or **Fluid Container** as the base container for all other controls.

While there are many different controls in this group, The primary controls for creating forms should be the **Container**, **Form Row**, and **Form Group** controls. The **Form Group** control is especially useful in that it already provides a visual association between the label and data control, placing the label immediately above the data control automatically.

All of the Responsive Layout controls are just HTML DIV tags with specific CSS styling applied to them. In the Online Form Designer, some editing options are available by right-clicking inside the control's visual placeholder, and selecting **Edit DIV Tag** or **Delete Div Tag** from the popup menu.

In general, new controls are added at the bottom of the form. It is difficult, without directly editing the HTML source, to move objects to different locations on the form. It would be best, therefore, to plan the layout of your form before starting, and adding the layout controls in the correct order.

Some controls can't be added to parent controls, unless the parent control is relevant to the child control you are attempting to add. You can't, for instance, add a **Form Group** control directly to a **Tab Content** container, even though the **Tab Content** is a general container object. The **Form Group** is only relevant to the **Form Row** parent container, so you'd first need to insert a **Form Row** control into the **Tab Content** container, then add the **Form Group** inside the **Form Row**.

To insert a child control into a parent container, click inside the container, then select the appropriate child control from the menu. You can, once a parent container has been selected, insert multiple child containers in sequence.

### *Available Layout Controls*

#### **Container**

This control is one of the base container controls that serve as the primary layout boundary for Bootstrap. This control is fixed-width at 1140px, with breakpoints at viewpoint widths of 960px, 720px, and 540px.

#### **Container Fluid**

This control is one of the base container controls that serve as the primary layout boundary for Bootstrap. This control is relative-width at 100% of the viewport width.

#### **Navbar**

This control inserts a navigation bar with an included image location and three configurable navigation links.

#### **Tab**

This control inserts a **Tab** bar that contains two **Tab Content** controls.

#### **Tab Content**

This control, when a **Tab** control is selected, inserts an additional **Tab Content** control to the **Tab**.

#### **Accordion**

This control inserts an **Accordion** control that contains two **Accordion Content** controls.

#### **Accordion Content**

This control, when an **Accordion** control is selected, inserts an additional **Accordion Content** control to the **Accordion**.

#### **Row**

This control inserts a **Row** control inside of a container.

#### **Column**

This control inserts a **Column** control inside of a **Row** container. Bootstrap layouts allow up to 12 **Columns** in a **Row**.

#### **Form**

This control inserts a simple HTML FORM control. This control isn't usually needed for Process Director forms, though there are some use cases where a simple HTML form might be appropriate.

#### **Form Row**

This control is the basic row control that should be used for form layout. It can be placed in any parent container, and **Form Group** controls can subsequently be placed inside the **Form Row** control to automatically format form controls.

## Form Group

This control is the basic control container that should be used for form layout. It can only be placed inside of a **Form Row** container. The Form Group should contain both a **Label** control and an input control, e.g., **Input**, **Dropdown**, etc. controls. The **Form Group** will automatically place the **Label** control immediately above, and aligned to the left side of, the input control.

## Table

This control inserts an HTML table into a container. This is primarily designed for the display of tabular data, and not for layout, so it shouldn't be used as a form design element.

## Card

This control inserts container that includes a configurable image placeholder, formatted title and content container. These containers can, unlike a standard **Section** control, be stacked horizontally, and will automatically refactor as the viewport width gets smaller, to display vertically, instead of horizontally.

## Card and Header

This control inserts **Card** container that includes a header and content container. The **Card and Header** control can be inserted inside of an existing **Card** container. So you can have a parent card control with a title and content that contains a card and header control as additional related content.

## Card Group

This control inserts container that includes three, horizontally-arrayed **Card** controls.

## *Other Control Tools*

You can view the documentation for all tools available in the Online Form Designer by using the Table of Contents on the upper right corner of the page, or by clicking one of the links below.

**Basic Controls:** The most commonly-used form design tools.

**Input Controls:** Controls that are commonly used to collect data, but are a bit less widely used than the basic controls.

**Other Input Controls:** Additional Input controls, consisting mainly of the different content picker controls.

**Actions:** These controls enable you to control form actions, like placing buttons, or choosing objects via a picker,

**Other Controls:** Controls that perform miscellaneous tasks like adding HTML content, or labels.

**Layout:** Controls that are used to govern the control layout for the template, such as tabs and sections.

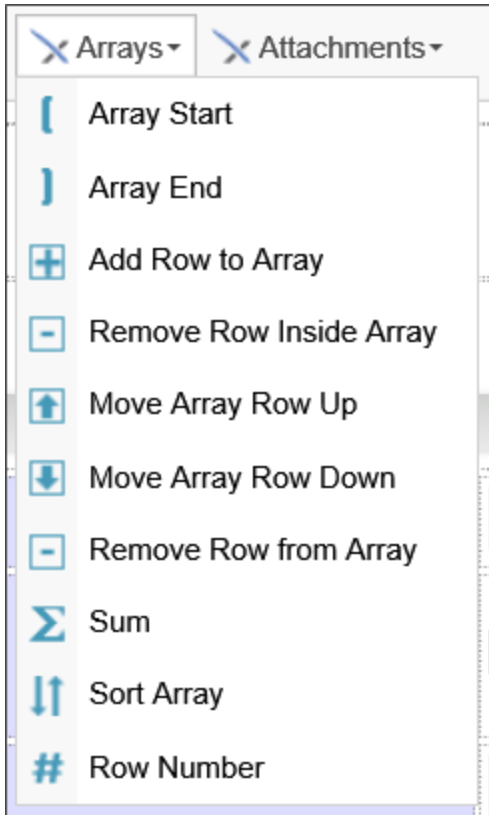
**Arrays:** Controls that enable to you create and control arrays on the Form.

**Attachments:** Controls that enable you to add and show attachments, such as documents or images, to the Form.

**Data List (v6.0.100 and higher):** Controls that enable the display of tabular data on a Form.

**Form Control Tags:** System Variables used to add controls to a Form, instead of using the UI controls.

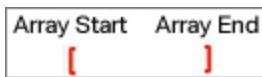
## Array Controls



When configuring a form, controls can be placed into a dynamic list, known as an array. An **array** consists of Form controls placed in a single table row, with array boundary controls at the beginning and end of the table row. At run-time, users can dynamically add or remove rows from the array. A control placed inside an array will always return the values entered into that field in a comma-separated list. For instance, a field inside an array that contains three rows might return the string "Value1,Value2,Value3" as the value of the array control. The value of a specific row in an array can also be retrieved as a single value through the use of the array row's index (**ROW\_NUM**). Please see the [Parameters topic](#) of the system variables guide for more information on extracting an array row value using the **ROW\_NUM** formatter.

### Array Start/End (Repeating Rows)

The array controls allow rows of items to be dynamically increased or decreased using an array tag. Using the array controls, you'll notice that red brackets mark the beginning and end of each array.



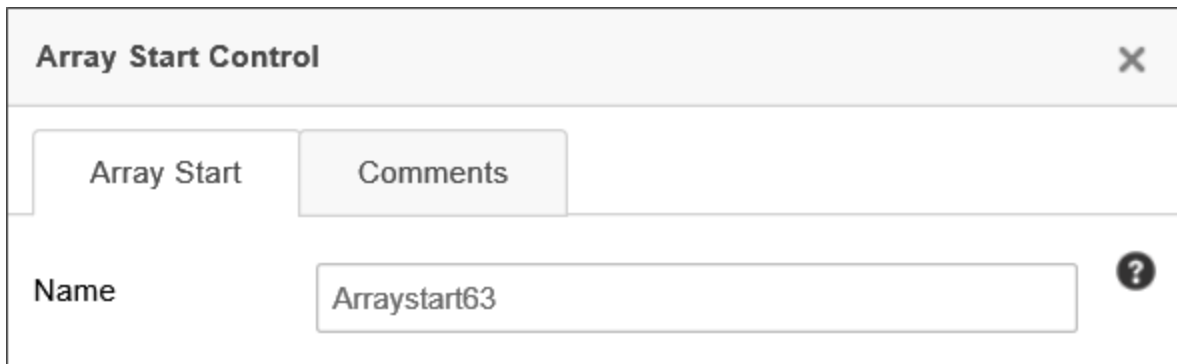
In general, Arrays are used in conjunction with table rows, with the **Array Start** control placed as the very first item in the first column of the table, and the **Array End** control placed as the very last item in the last column of the table. Each **Array Start** control requires a corresponding **Array End** control to close the array. All of the controls placed between the **Array Start** and **Array End** controls comprise the array row.



**!** Arrays can't be nested inside each other.

**i** An Array row can span more than one table row, as long as the Array End control is the last control in the last column cell of the table row.

Double-clicking the **Array Start** control will open the **Properties** dialog box, in which you can configure the settings for the array.



An array has a **Value** property, which is the number of rows in the array. An array with a value of "2", therefore, will display two rows. The default value for an array is "0", which means that, as a default, an array won't display any rows when a Form first opens. It is common, however, to want the first row of the array to appear on the Form when it opens, which means that the default value of the array needs to be set to "1". Setting the value for an array is done in the **Form Controls** tab of the Form definition by configuring the field properties for the array. In the **Form Field Properties** dialog box for the array, set the **Default Value** property to a number with the value of "1".

The image shows a dialog box titled "Form Field Properties: ArrayItems". It contains several configuration options for an array item. The "Default Value" section is highlighted with a red box, showing a dropdown menu set to "Number" and a text input field containing the value "1". Other options include "Friendly Name", "Link to Attribute", "Case Property" (set to "None"), "Set Readonly Options", "Set Display Options", "Field is Required", and "Set Style Options". At the bottom, there are "OK" and "Cancel" buttons.

### Add/Remove Row Controls

You can allow the user to manually add or remove a row from an array by adding the [Add Row to Array](#), [Remove Row from Inside Array](#), and [Remove Row from Array](#) controls to your Form.

When placing these controls on the page, the [Add Row to Array](#) and [Remove Row From Array](#) controls should be placed outside of the array controls. Each of these controls has an [Array Name](#) property, into which you must enter the name of the array to which you wish to add or remove rows. Both controls will add/remove the *last* row from the array.

The **Remove Row from Inside Array** control should be placed inside the array controls, so that it will appear on each row of the array. When a user clicks this control, the row corresponding to the button will be removed from the array, unlike the **Remove Row From Array** control, which always removes the last row from the array.

### Move Row Up/Down

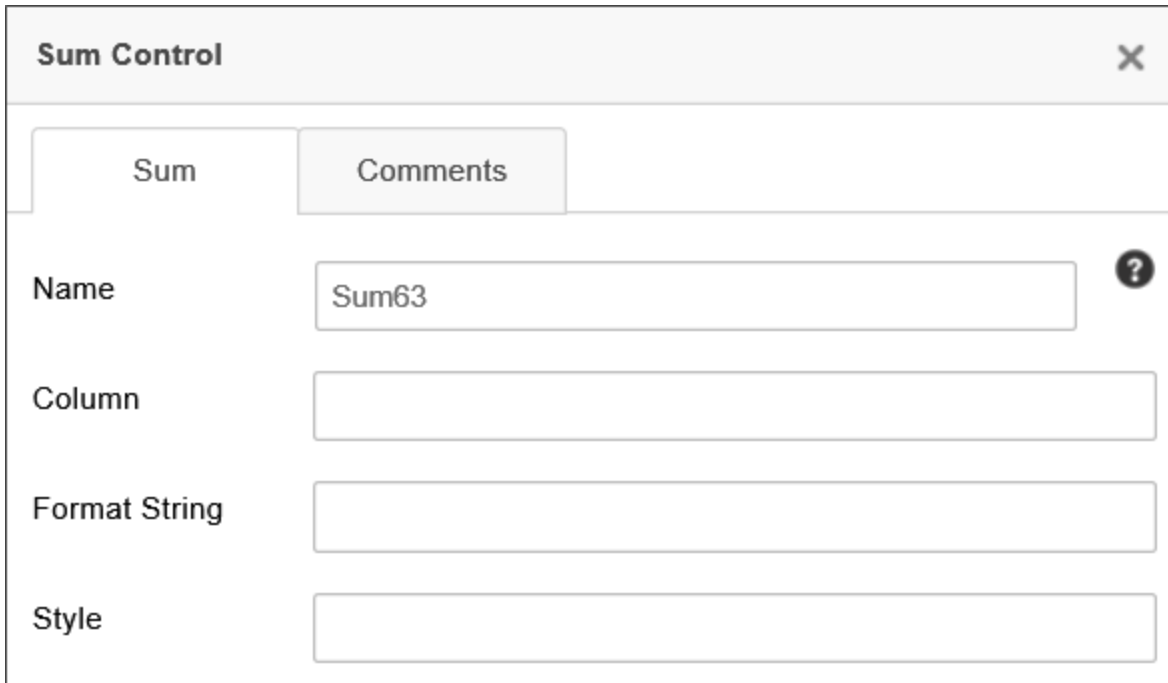
The **Move Rows** controls should be placed inside the array controls so that you can use them to move the row up or down in the array.

### Sum

The **Sum** control displays the sum of all of the values in a column of the array.



To configure a sum control, open up the properties window by double clicking on the control inside the Form builder.



The image shows a configuration dialog box titled "Sum Control" with a close button (X) in the top right corner. The dialog has two tabs: "Sum" and "Comments". The "Sum" tab is currently selected. Below the tabs, there are four labeled input fields: "Name" (containing "Sum63" and a help icon), "Column" (empty), "Format String" (empty), and "Style" (empty).

In the **Column** field, enter the name of the control in the column you'd like to sum.

### Sort Array

The **Sort Array** control gives the user a button to sort the array alphabetically. The array can be sorted by three columns, in order of preference. This means that, if two elements in the primary column are the same, that those rows will be sorted according to the secondary column, and so on.



To configure the **Sort Array** control, open the properties window by double clicking the control, to open the **Properties** dialog box for the control. The control has two primary configuration tabs: The **Sort** and **Button Properties** tabs. The **Comments** tab has no effect on the control's display, but enables you to enter and save designer comments for the control.



## Sort Tab

**Sort Array Button**
✕

Sort
Button Properties
Comments

**Name**  ?

**Text**

**Array Name**

**Primary Column**

**Secondary Column**

**Tertiary Column**

**Descending?**

The **Sort** tab allows you to configure how the control sorts. You can define the text of the control button, its name, and the columns it uses as keys to sort the array. You can also have the control sort the array in descending order.

### Properties

**Name:** The Control name for the control.

**Text:** The Text that will be displayed on the control's button at run-time.

**Array Name:** The Control name for the array you wish to sort, e.g., "Arraystart1".

**Primary Column:** The control name of the control you wish to use as the first column to sort, e.g., "LastName".

**Secondary Column:** The control name of the control you wish to use as the second column to sort, e.g., "FirstName".

**Tertiary Column:** The control name of the control you wish to use as the third column to sort, e.g., "City".

## Button Properties Tab

### Sort Array Button ✕

Sort Array | **Button Properties** | Comments

Image URL

Icon Number

Color

Back Color

Confirm Text

OnClickClick

Alt Tag

Style

Access Key

Use small image?  Use image as entire button?

The **Button Properties** tab enables you to configure the button for the **Sort Array** control. You can provide the URL of an image to use for the button, and style it via CSS. Upon clicking the button, you can ask the user to confirm that he wants to sort the array. You can also specify a JavaScript function to be run when the user clicks on the button.

## Properties

**Image URL:** The optional fully-qualified URL of a GIF, JPG, or PNG file to use as an image for the button, if any, instead of an icon.

**Icon Number:** This property will, when the **Choose** button is clicked, open the **Icon Chooser**, from which you can choose the icon, and its color, to display on the button, in lieu of the standard **Sort** icon that is displayed by default.

**Color:** This property will, when the **Choose** button is clicked, open the **Choose Color** dialog box to choose the text color of the button.

**Back Color:** This property will, when the **Choose** button is clicked, open the **Choose Color** dialog box to choose the background color of the button.

**Confirm Text:** The optional confirmation message you'd like to appear in a confirmation dialog when the button is clicked. Leaving this property blank won't display a confirmation dialog.

**OnClickClient:** The optional name of a JavaScript routine that is contained on the form, and which you'd like to run as client-side script when the button is clicked, e.g., "SortButtonClickHandler".

**Alt Tag:** The Alt Tag text for the button, for accessibility.

**Style:** Optional CSS style commands to format the button via CSS.

**Access Key:** The keystroke combination to set focus to the control, for accessibility.

**Use Small Image?:** This option, when checked, will reduce large images to a fixed size for use as a button icon, for images specified using the **Image URL** property.

**Use Image as Entire Button?:** This option, when checked, will, for images specified using the **Image URL** property, display the specified image as an image button. The image will **not** be resized when this setting is used.

## Row Number

The **Row Number** control simply displays the row number (starting with row #1).

## *Array Data, Conditions, and Validation #*

## Set Form Data

When using Array Form fields in the **Set Form Data** tab of a Form Definition, an additional text box will be displayed to the right of the array field name to enable you to enter a specific row number. By default, this field is blank, and the field value will return all of the values in that array field as a comma-separated list.

[View State Init]

[Click to create condition ...](#)

Array ▼ 0 ... ↑ ↓ ✖

Business Value: Excel Array + Add Form Field

Input1 (Input 1) ▼		Column1 ▼	↑ ↓ ✖	↑ ↓ ✖
Input2 (Input 2) ▼		Column2 ▼	↑ ↓ ✖	
Input3 (Input 3) ▼		Column3 ▼	↑ ↓ ✖	

You can specify a specific row to change by entering the row number in this text box, e.g.:

[View State Init]

[Click to create condition ...](#)

Array ▼ 0 ... ↑ ↓ ✖

Business Value: Excel Array + Add Form Field

Input1 (Input 1) ▼	2	Column1 ▼	↑ ↓ ✖	↑ ↓ ✖
Input2 (Input 2) ▼	2	Column2 ▼	↑ ↓ ✖	
Input3 (Input 3) ▼	2	Column3 ▼	↑ ↓ ✖	

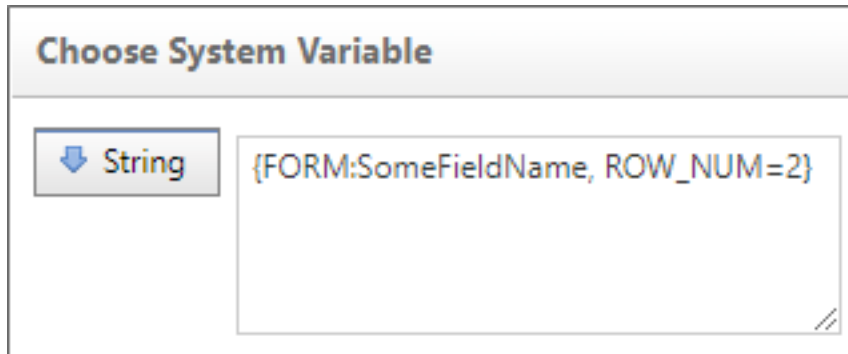
In the example above, the data would be set only in the second row of the array. Arrays rows are 1-based, which is to say that the first row of the array starts at the number 1.

## Conditions

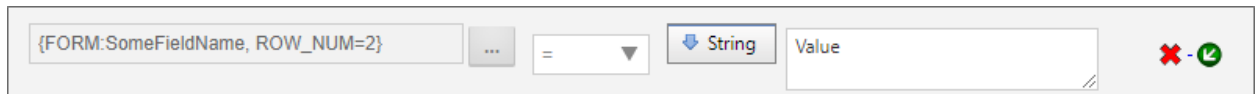
In the [Condition Builder](#), using an array field will, by default, try to evaluate the entire contents of an array field, meaning the entire list of field values will be returned in a single string, as a comma-separated

list. You can, however, evaluate a single array value by using a Form field System Variable with the syntax:  
`{FORM:SomeFieldName, ROW_NUM=2}`

In this case, you won't be able to select the **Form Field** menu item of the **Choose System Variable** dialog box to specify the field. Instead, you'll need to use the **Value > String** menu item, and enter the System Variable on the left side of the condition.



You can enter the System Variable as the string value and, at run-time, the system will use only the value from the specified row of the array to use in the comparison. The condition row in the **Condition Builder** will then appear similar to the example below.



Arrays can, of course, have any number of rows, so, if you use a specific row number, be aware that the specified row may not actually exist in the array. You can, however, always access the last row in an array using the System Variable syntax:

`{FORM:SomeFieldName, ROW_NUM={NUM_ROWS}}`

The `{NUM_ROWS}` system variable always returns the number of rows in an array, and thus can always be used as a pointer to the last array row, irrespective of how many rows the array contains.

## Validation

Array fields can be used in validation conditions, but you should be aware that complex conditions may not return the results you expect, because Process Director will evaluate each element of a complex condition separately. For instance, let's say that you have two columns in an array that contain the following values.

COLUMN 1	COLUMN 2
3	ABC
1	DEF

If you set a complex condition, such as, Column 1 = "3" AND Column 2 = "DEF", the condition will return "True", even though the values you're looking for don't exist in the same row.

A more comprehensive look at array conditions is presented in the [Using Arrays in Conditions](#) section of the **Condition Builder** topic.

Row numbers for an array row can also be accessed via the [Row Number System Variable](#).

### *Other Control Tools*

You can view the documentation for all tools available in the Online Form Designer by using the Table of Contents on the upper right corner of the page, or by clicking one of the links below.

**Basic Controls:** The most commonly-used form design tools.

**Input Controls:** Controls that are commonly used to collect data, but are a bit less widely used than the basic controls.

**Other Input Controls:** Additional Input controls, consisting mainly of the different content picker controls.

**Actions:** These controls enable you to control form actions, like placing buttons, or choosing objects via a picker,

**Other Controls:** Controls that perform miscellaneous tasks like adding HTML content, or labels.

**Layout:** Controls that are used to govern the control layout for the template, such as tabs and sections.

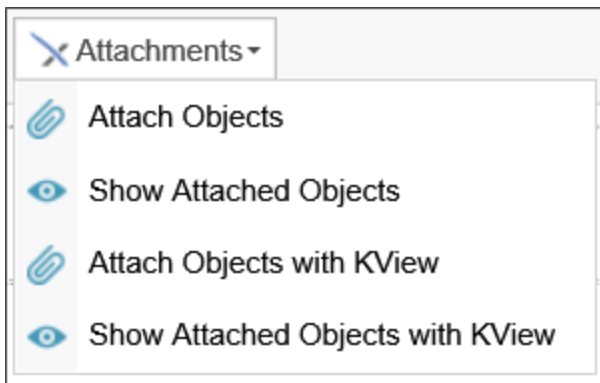
**Responsive Layout:** Controls that implement Bootstrap form layout objects.

**Attachments:** Controls that enable you to add and show attachments, such as documents or images, to the Form.

**Data List (v6.0.100 and higher):** Controls that enable the display of tabular data on a Form.

**Form Control Tags:** System Variables used to add controls to a Form, instead of using the UI controls.

## Attachments



The attachment controls enable items to be attached to, and viewed from, a Form (as a reference or as part of a process package). Documents that are references or part of a process package can be embedded and viewed on the Form or displayed in a separate window via hotlinks, using these controls.

### Attach Objects

This control enables users to add document attachments to the Process or Form.

## Control Properties

Double clicking the control allows you to access and configure its properties.

As of Apple iOS 6.2, when clicking on the **Attach** button in Safari or Chrome, iOS will prompt the user to either take a picture or select one to attach from his photo library.

In addition, the **Attach Objects** control also allows you to call JavaScript client behaviors. You can specify the JavaScript call you'd like to make by entering it in the **OnClientClick** property of the control.

The **Properties** dialog for this control has a number of configurable tabs.

## Button Tab

The Button tab is the primary configuration tab for the control, and contains the properties listed below.

**Name:** The Control Name.


**Text:** The Text that will appear on the button's label.

**Attach Type:** Determines to which Process Director object the item will be attached. You can set the item to attach to the following objects:


- Form
- Workflow
- Timeline
- Process

The default selection is "Process".

**Object Type:** Determines whether the item to be attached is a document/file, or a clipboard image item that will be attached from the machine's clipboard memory. The default is "Document".

 The "Clipboard" selection is largely deprecated, as it requires ActiveX technology that works only with Internet Explorer when the BP Logix plugin is installed, and is running on a 32-bit computer.

**Group Name:** The Group Name of the group to which the item will be attached.

 As a best practice, you should always apply Group Names for attachments, even if you only have a single attachment control, or a just a single document to attach. This will make your life much easier if you later want to export just certain documents, or access just certain types of forms in a process.

**Clipboard Image Name:** If the attachment item comes from the clipboard, this property will assign a file name to the item. Please see the note in **Object Type** above for the clipboard's operability.

**Use MFT Support?:** Selecting this property will enable users to upload multiple files to form asynchronously. When the files are uploading, a separate, pop-up, upload window will appear in the browser. This window **MUST** remain open for the uploads to complete. Closing prematurely will stop the uploads. In the meantime, while the upload is being completed, you can continue filling out the form without waiting on the uploads. When using this feature, in Process Director v5.12 or higher, the Show Attach control will display the % complete of the uploaded file, and the tooltip will display the last time data was received. This can be used by a user to determine if they accidentally closed the browser window doing the upload, and remove the file and upload again.

Large file uploads may require that you specify a temporary folder for the file during MFT uploads, in lieu of the default temp folder, as sometimes large documents get caught in the temp folder. To alleviate this, you can set the [UploadTempPath](#) Custom Variable to the folder you'd like to use for temporary storage.

**MFT Page Title:** This property enables you to configure the title of the upload window that will be displayed to the user when the file upload begins.

**Allowed File Extensions:** This property accepts a comma-separated list of file extensions (e.g. docx,xls,pdf). At run-time, only files with the specified extensions can be uploaded. Optionally, there is a Custom Variable called [FileUploadBlacklist](#) that also accepts a comma-separated list of file extensions to create a blacklist that will reject all file uploads of the enumerated file types. If no extensions are entered in the Attachment control, then all file types can be uploaded, with the exception of any that may be in the FileUploadBlacklist. For Process Director v5.26 and higher, the allowed file types are displayed to the user when uploading a file attachment.



**Allowed File Alternate Text:** This property enables you to specify the text you'd like to display to the user when you use the **Allowed File Extensions** property to restrict file types for upload. This property can be set universally through the use of the [FileUploadBlacklistAlternateText](#) Custom Variable.

## Button Properties Tab

This tab enables you to configure additional properties for the **Attach** button.

The screenshot shows a dialog box titled "Attach Button" with a close button (X) in the top right corner. The dialog has four tabs: "Button", "Button Properties", "More Button Properties", and "Comments". The "Button Properties" tab is selected. The form contains the following fields and controls:

- Image URL:** A text input field.
- Icon Number:** A text input field with a "Choose" button to its right.
- Color:** A text input field with a "Choose" button to its right.
- Back Color:** A text input field with a "Choose" button to its right.
- Confirm Text:** A text input field.
- OnClientClick:** A text input field.
- Alt Tag:** A text input field.
- Style:** A text input field.
- Access Key:** A text input field.
- At the bottom, there are two checkboxes:
  - Use small image?
  - Use image as entire button?

**Image URL:** The button can be displayed as an image by providing an accessible image URL to display instead of a standard button. This property will override other button properties below, e.g., Back Color.

**Icon Number:** You can select an Icon to display on the button from the [Icon Chooser](#).

**Color:** You can select the text color to display on the button from the [Select Color](#) dialog box.

**Back Color:** You can select the background color to display on the button from the [Select Color](#) dialog box.

**Confirm Text:** You can specify confirmation text that will appear as a confirmation dialog box when the button is clicked.

**onClientClick:** You can specify client-side JavaScript to run on the button's onClientClick event.

**Alt Tag:** You can specify an ALT tag property for accessibility purposes.

**Accessibility Key:** You can specify the accessibility keys a user can press to activate the button, for accessibility purposes.

**onClientClick:** You can specify client-side JavaScript to run on the button's onClientClick event.

**Use Small Image?:** If you supply an image URL, the image can be displayed as an icon for the button by checking this property.

**Use Image as entire button?:** You can use the image as the displayed button, rather than displaying a traditional button on the form.

## More Button Properties Tab

Additional properties are configured on this tab.

The screenshot shows a dialog box titled "Attach Button" with a close button (X) in the top right corner. Below the title bar are four tabs: "Button", "Button Properties", "More Button Properties" (which is selected), and "Comments".

Under the "More Button Properties" tab, there are two checkboxes: "Drag and Drop?" and "Single File Upload?". Below these is a text area labeled "Message on attach control".

At the bottom, there are three input fields labeled "ConceptShare Project", "ConceptShare Folder", and "ConceptShare Version".

**Drag and Drop?:** Selecting this property will enable users to drag a file from the file system directly onto the Form to upload it.

**Single File Upload?:** Selecting this property will restrict each upload to a single file.

**Message on Attach Control:** You can display a longer message on the attach control, in addition to the button text. This may be useful for supplying instructions, etc., to the user.

**ConceptShare Project:** Users of ConceptShare can specify a Project to assign to the attachment.

**ConceptShare Folder:** Users of ConceptShare can specify a Folder to assign to the attachment.

**ConceptShare Version:** Users of ConceptShare can specify a Version to assign to the attachment.

## Form Definition Properties

Additional properties are available for the Attach Objects control in the **Form Controls** Tab of the form definition. Opening the control's property in the **Form Controls** tab will display the properties dialog box to access the additional properties.

The relevant properties are:

PROPERTY	DESCRIPTION
Min	The minimum allowable file size for the attachment, in Kilobytes, e.g., a value of "64" would be a size limit of 64KB.
Max	The maximum allowable file size for the attachment, in Kilobytes, e.g., a value of "64" would be a size limit of 64KB.

 For Process Director v5.32 and higher, attachments may be added to completed processes. Prior versions don't allow post-completion attachments.

### Show Attached Objects

A **Show Attached Objects** field appears like this in the edit mode of the Online Form Designer.

This control displays the document attachments that you uploaded via the **Attach** control described above. The **Properties** dialog for the **Show Attached Objects** control has two configuration tabs.

## Attach Tab

The primary properties to configure are found on the **Attach** tab.

You *must* ensure that the **Attach Type**, **Object Type** and **Group Name** properties are the same properties you configured for the **ShowAttach** control, if you want to show the specific documents that were uploaded via that **Attach** control. For the **Object Type** option, however, you have an option to show *All* attachments, in addition to Form or Process attachments specifically.


Setting the **Group Name** property to "" will only display attachments that have no **Group Name** associated with the attachment, while leaving it blank will show all attachments of that **Attach Type** and **Object Type**. An additional **Object Type**, "Case", is available to display objects that are attached to the Case instance.

For Process Director v5.26 and higher, you can also use the percent sign (%) as a wildcard character to match a partial **Group Name**. For example, if you have two groups named "Final Drafts" and "Final Documents", setting the **Group Name** to "Final%" will return attachments in both groups. This functionality can be disabled using the [fEnableOldShowAttach](#) custom variable.

For Process Director v5.31 and higher, specific Group Names can be excluded from display by using the "!" character with the group name, e.g., "!MyGroup" will exclude items from the group named "MyGroup".

The screenshot shows a dialog box titled "Show Attachments" with a close button (X) in the top right corner. The dialog has three tabs: "Attach", "Attach Properties", and "Comments". The "Attach" tab is selected. The "Attach" tab contains the following fields:

- Name:** A text input field containing "Showattach2" and a help icon (?) on the right.
- Attach Type:** A dropdown menu with "Process" selected.
- Object Type:** A dropdown menu with "All" selected.
- Group Name:** An empty text input field.
- Sort By:** An empty text input field.
- Sort Type:** An empty text input field.

 As a best practice, you should always apply Group Names for attachments, even if you only have a single attachment control, or a just a single document to attach. This will make your life much easier if you later want to export just certain documents, or access just certain types of forms in a process.

Additional configurable properties on the **Attach** tab are the **Sort By** property, which will sort the display of attachments by Name or Date. The **Sort Type** will sort in Ascending or Descending order. The text values that are valid for these properties are:

#### Sort By Options

- Name - The name of the attachment.
- CreateTime - The date and time the attachment was created.
- UpdateTime - The date and time that the attachment was last edited.

#### Sort Type Options

- Ascending
- Descending

For Process Director v5.38 and higher, when an anonymous user is assigned a task via an email and they attach a document to a form, the document's "Create User" attribute will display the anonymous user's email address on the Show Attached Objects control.

## Attach Properties Tab

### Show Attachments ✕

**Attach** | **Attach Properties** | Comments

<input type="checkbox"/> View Inline?	Inline Height <input type="text"/>
<input type="checkbox"/> Use MFT Support?	MFT Page Title <input type="text"/>
<input type="checkbox"/> Show Edit	<input type="checkbox"/> Show Modify User
<input type="checkbox"/> Show Remove	<input type="checkbox"/> Show Modify Date
<input checked="" type="checkbox"/> Show User	<input type="checkbox"/> Show Review
<input checked="" type="checkbox"/> Show Date	<input checked="" type="checkbox"/> Show Download
<input checked="" type="checkbox"/> Show View	<input type="checkbox"/> Show History
<input type="checkbox"/> Show SharePoint	<input type="checkbox"/> Show Group Name
<input type="checkbox"/> Show Description	<input type="checkbox"/> Show Thumbnail
<input type="checkbox"/> Use Definition Name	<input type="checkbox"/> Hotlink Name

Custom Text  Custom Script

Review Permissions  ▼

Edit Permissions  ▼

This control offers a variety of options to display information in the control when it is displayed. If you are displaying Process Timeline/Workflow/Process references, the [ShowAttach](#) control won't display the current form despite the fact that it is a Process Timeline/Workflow/Process reference.

The default settings for this control are shown above. Most are fairly obvious but a few require more explanation.

By default, the **ShowAttach** control doesn't display the attached object in the browser, but just adds a link to be able to download or view it. To show it within the browser, check the **View Inline?** checkbox. You may then configure the **Inline Height** property to specify how large, vertically, in pixels, the inline display of the attachments will be when the form is displayed.

Process Director versions 3.45 and higher have the ability to show thumbnails in the ShowAttach control by selecting the **Show Thumbnail** check box. Thumbnail display is available for the following file types:

- JPG/GIF/PNG/TIF/BMP
- DOC/DOCX
- XLS/XLSX
- PPT/PPTX
- TXT/CSV
- PDF

If you check the **Show Edit** check box, an **Edit** link will appear in the control to allow the user to checkout and download the attachment for editing. When they do so, an **Upload** and **Cancel** links will appear. When the document is uploaded or canceled, the file will be checked back in or the checkout voided, depending on what you choose. When the Edit feature is enabled, you can also select the **Use MFT Support** property to upload the edited file asynchronously. When the file is uploading, a separate, pop-up, upload window will appear in the browser. This window **MUST** remain open for the upload to complete. Closing prematurely will stop the upload. In the meantime, while the upload is being completed, you can continue filling out the form without waiting on the upload. The **MFT Page Title** property enables you to configure the title of the upload window that will be displayed to the user when the file upload begins. Additionally, an Administrator can use **Debug Mode** to set the status to complete from the document properties page, if necessary.

For Process Director v5.34 and higher, when a **Download** link is displayed in the **ShowAttach** control for a document that has been converted to PDF, the download action will download the version of the attachment that was converted to PDF.

## Collaborative Document Authoring

For Cloud installations of Process Director v5.13 and higher, document attachments may be edited using [Collaborative Document Authoring](#), as an optionally licensed component. If both the **Show Edit** and **Use Online Editor** properties are checked, clicking the **Edit** link in the running form will open the document in an online document editor for Word, Excel, or PowerPoint, instead of downloading a copy of the document to the local computer. Change tracking is automatically enabled in Collaborative Document Authoring, so that edits from multiple authors can be reconciled, accepted, or rejected.

The **Edit Permissions** property enables you to set the edit methods for collaborative authoring. The following options are available:

- **Default Checkout and Edit:** Checks out the document for offline editing and check-in.
- **Online Editor Edit Only:** Allows editing only in the Collaborative Document Authoring UI.

- **Online Editor Track Changes On:** Allows editing in the Collaborative Document Authoring UI, with Track Changes turned on.
- **Online Editor Review Changes:** Allows editing in the Collaborative Document Authoring UI, with Track Changes turned on, and enables users to review/accept changes.

## Collaborative Document Markup

For users of Process Director v5.13 and higher, document attachments may be annotated using Collaborative Document Markup, as an optionally licensed component. To enable this, check the **Show Review** property. Clicking the **Review** link in the running form will open the Markup tool. For more information on the use of this tool, please refer to the [Collaborative Document Markup](#) topic.

Markups/comments may be added, edited, or deleted by end users using the Markup tool, depending on the setting you select from the **Review Permissions** dropdown control. The following options are available:

- **No Comments (default):** No commenting is allowed.
- **User Can View Comments:** Users can view only their own comments.
- **View All Users Comments:** Users can view all comments.
- **User Can Edit Comments and View Others:** Users can view all comments, but edit only their own.
- **Edit All Users Comments:** Users can edit/delete all comments by any user.

## Additional Custom Variable Settings

There are also some settings you can make to the Customization file to alter the way the **ShowAttach** control works.

For thumbnails to display, you must also set the [fEnableThumbnails](#) custom variable to "True" in the Customization File.

Also, for Process Director v3.45 and higher, when selecting the **Show Edit?** option, Process Director will no longer display the **History** tab in the popup. If a user requires access to the history of a document from the Form, select the **Show History?** property to display a link on the form that opens a popup to the document history.

As a default, if the ShowAttach control is disabled, the **History** tab won't display. This behavior can be changed by setting the [ShowDocHistoryWhenDisabled](#) custom variable to "true".

**Custom Text/Script:** Custom text and custom JavaScript can be added to an attachment displayed on a form. Simply enter the custom text or JavaScript into the provided text boxes to display the desired text or run the JavaScript code when the form is displayed.

## Attach Objects With KView

This control enables you to attach an object to a Form instance by browsing for the object using a Knowledge View.



To configure the Knowledge View that you wish to display in this control, set the Default Value of the control to a "Content Item" in the control's Form field properties on the Form Controls tab of the Form definition, then point to the appropriate Knowledge View. This configuration must be



done *in addition* to any configuration settings you configure in the Online Form Designer as part of the control's properties.

The primary control configuration for this control is done via the **Button** tab of the control's **Properties** dialog.

**Attach Objects with KView** [X]

Button | **Button Properties** | Comments

Name: Attachkview63 [?]

Text: Add Document

Attach Type: Process [v]

KView ID: [ ]

Group Name: [ ]

Items: [ ]

Attach to Parent     Move Object     Copy Object

**Name:** The Control Name.

**Text:** The Text that will appear on the button's label.

**Attach Type:** Determines to which Process Director object the item will be attached. You can set the item to attach to the following objects:

- Form
- Workflow
- Timeline
- Process

The default selection is "Process".

**KView ID:** This is the ID of the Knowledge View you wish to display to attach the object.

**Group Name:** Only display objects from the specified group

**Items:** A query string to send data to the Knowledge View.

**AttachToParent:** If checked, this object will be attached to the parent of this Form instance

**Move Object:** If checked, this object will be moved to a new location (removing the object in its old location)

**Copy Object:** If checked, this object will be copied to the new location (leaving the old object alone).

### Show Attach Objects With KView

This Form control will add an Iframe to the Form showing a Knowledge View of attached objects.



To configure the Knowledge View that you wish to display in this control, set the Default Value of the control to a "Content Item" in the control's Form field properties on the Form Controls tab of the Form definition, then point to the appropriate Knowledge View. This configuration must be done *in addition* to any configuration settings you configure in the Online Form Designer as part of the control's properties.

The primary control configuration for this control is done via the **Control** tab of the control's **Properties** dialog.

### Show Attached Objects with KView ✕

Control Comments

Name  ?

Attach Type  ▼

KView ID

Group Name

Height

Width

Show Parents

Items

**Name:** The Control Name.

**Attach Type:** Determines to which Process Director object the item will be attached. You can set the item to attach to the following objects:

- Form
- Workflow

- Timeline
- Process

The default selection is "Process".

**KView ID:** This is the ID of the Knowledge View you wish to display to attach the object.

**Group Name:** Only display objects from the specified group

**Height:** Height of the Iframe.

**Width:** Width of the Iframe

**ShowParents:** If checked, the Knowledge View will show the parents of the object.

**Items:** A query string to send data to the Knowledge View.

### *Using Groups with Attachments*



As a best practice, you should always apply Group Names for attachments, even if you only have a single attachment control, or a just a single document to attach. This will make your life much easier if you later want to export just certain documents, or access just certain types of forms in a process, or many other things.

Both the **Attach Objects** and **Show Attached Objects** controls have a **Group Name** property, which is particularly useful when you need to enable users to upload more than one set of attachments. For example, let's say that during a process, different people must load different types of documents, as follows:

- Initial Requester: Supporting documents, invoices, etc.
- QA Section: Test reports
- Compliance Section: Regulatory documents

In this example, there are three different types of documents that need to be attached to the process or Form. While a single set of **Attach Objects** and **Show Attached Objects** controls will allow everyone to upload their documents, they would all be collected in a single set of attachments. The **Group Name** property enables you to place three sets of attachment controls on the Form, each with a different group name, so that each set of people can upload documents into a Form section that is specific to their group. The first set of **Attach Objects** and **Show Attached Objects** controls might have the **Group Name** "SupportingDocs", the second set would have the **Group Name** "QA", and the third set might have the **Group Name** "Compliance". With this configuration, you can allow each group of people to upload documents only to the Group that is relevant to them, while disabling or hiding the other attachment groups, as desired, when the Form is being edited. As long as an **Attach Objects** control and a **Show Attached Objects** control share the same **Group Name**, Process Director will organize the attachments properly by group.

Supporting Docs

Attach Supporting Documentation: Attach

Launch_Request.doc	<a href="#">View</a> <a href="#">Download</a> <a href="#">Edit</a> <a href="#">Remove</a>	1/26/2016 2:03 PM	Dale Franks
--------------------	---	-------------------	-------------

Test Reports

Attach Test Reports: Attach

test.xml	<a href="#">View</a> <a href="#">Download</a> <a href="#">Edit</a> <a href="#">Remove</a>	1/26/2016 2:04 PM	Dale Franks
test2.xml	<a href="#">View</a> <a href="#">Download</a> <a href="#">Edit</a> <a href="#">Remove</a>	1/26/2016 2:04 PM	Dale Franks

Compliance Docs

Attach Compliance/Regulatory Documents: Attach

case.png	<a href="#">View</a> <a href="#">Download</a> <a href="#">Edit</a> <a href="#">Remove</a>	1/26/2016 2:05 PM	Dale Franks
casework.png	<a href="#">View</a> <a href="#">Download</a> <a href="#">Edit</a> <a href="#">Remove</a>	1/26/2016 2:05 PM	Dale Franks

### Other Control Tools

You can view the documentation for all tools available in the Online Form Designer by using the Table of Contents on the upper right corner of the page, or by clicking one of the links below.

**Basic Controls:** The most commonly-used form design tools.

**Input Controls:** Controls that are commonly used to collect data, but are a bit less widely used than the basic controls.

**Other Input Controls:** Additional Input controls, consisting mainly of the different content picker controls.

**Actions:** These controls enable you to control form actions, like placing buttons, or choosing objects via a picker,

**Other Controls:** Controls that perform miscellaneous tasks like adding HTML content, or labels.

**Layout:** Controls that are used to govern the control layout for the template, such as tabs and sections.

**Responsive Layout:** Controls that implement Bootstrap form layout objects.

**Arrays:** Controls that enable to you create and control arrays on the Form.

**Data List (v6.0.100 and higher):** Controls that enable the display of tabular data on a Form.

**Form Control Tags:** System Variables used to add controls to a Form, instead of using the UI controls.

### Data List Controls

 This topic is a work in progress for the future release of Process Director v6.1.0. It is subject to change without notice until release.

For Process Director v6.1.0 and higher, a new control dropdown menu, **Data List**, will appear. This menu will contain the selection for the **Data List** Form control, and its associated controls.

<input type="checkbox"/>	Column 1 ▾	Column 2 ▾	Column 3 ▾
<input type="checkbox"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="checkbox"/>	1	Carl Jackson	13.25
<input type="checkbox"/>	2	Barry French	9.45
<input type="checkbox"/>	3	Joby O'Brien	7.61
<input type="checkbox"/>	4	Carlos Lopez	5.25
<input type="checkbox"/>	5	Jay O'Brien	4.12

Navigation:

Unlike other types of Form field, the contents of the **Data List** control do not get saved with the Form instance as Form data. Instead, the **Data List** control is largely a display control that presents large data-sets from external sources on the Form. The data displayed in the **Data List** can be selected and transferred to other Form fields for saving as Form data when desired.

As an example, you might have a long list of products that you present in a **Data List** on a product order form. When ordering products, a user could select the desired products from the **Data List**, which, when selected, could be stored in array fields on the Form. When submitted, the selected products would be stored as part of the order. Please see the [Usage](#) section below for an example of this use case.

 **Data List**

### Data List Control

Data List

Formatting

Field Properties

Comments

Name  ?

Selection mode

No Selection  Row Selection

Virtualization type

Visible Rows

Pagination Page Size

None (Not recommended)

Sorting

Column Filtering

Static Header

Export to CSV

Export to PDF

Refresh

Clear

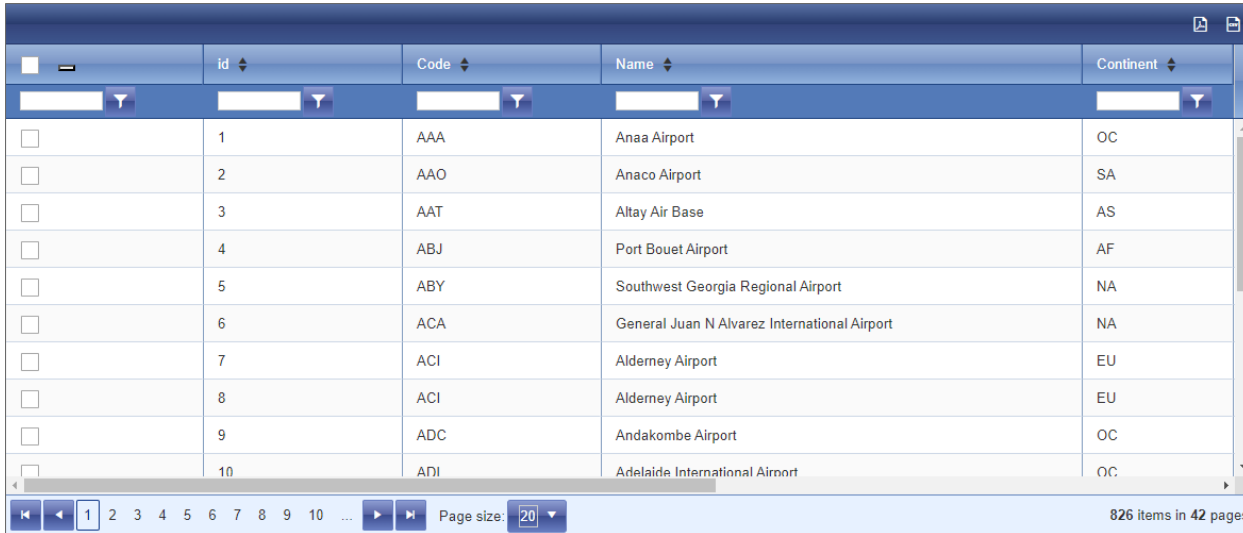
Column Resize

Width:

Height:

The **Data List** control enables the display of table data of nearly any size for display on a Process Director Form. Prior to Process Director v6.0.100, the only method of displaying tabular data was through the use of [Array controls](#). Array controls, however, can only display relatively large sets of tabular data when the array controls are disabled and displayed only as text. Otherwise, the use of a large number of active controls in a Form array uses excessive system resources, and will make the form nonfunctional after a certain number of fields.

The **Data List** control, on the other hand, completely solves the issue of presenting and using large data sets on a Form. Data can be selected by cell or by row, and transferred from the **Data List** cell or row to a separate set of Form fields (either regular or Array fields). For example, in the screen shot below, the **Data List** is displaying a recordset containing 826 rows of data on a Form, with no perceptible loss of Form performance or resource usage.



	id	Code	Name	Continent
<input type="checkbox"/>	1	AAA	Anaa Airport	OC
<input type="checkbox"/>	2	AAO	Anaco Airport	SA
<input type="checkbox"/>	3	AAT	Altay Air Base	AS
<input type="checkbox"/>	4	ABJ	Port Bouet Airport	AF
<input type="checkbox"/>	5	ABY	Southwest Georgia Regional Airport	NA
<input type="checkbox"/>	6	ACA	General Juan N Alvarez International Airport	NA
<input type="checkbox"/>	7	ACI	Alderney Airport	EU
<input type="checkbox"/>	8	ACI	Alderney Airport	EU
<input type="checkbox"/>	9	ADC	Andakombe Airport	OC
<input type="checkbox"/>	10	ADI	Adelaide International Airport	OC

The following fields are available to configure the Data List control.

**Name:** The Control's name.

**Selection Mode:** This property enables you to specify the method, if any, that will be used to select data in the control. There are three selection options:

- *No Selection:* No data will be selected. This is the default option.
- *Cell Selection:* This option enables users to select one or more specific cells in the list, in order to extract their data.
- *Row Selection:* This option will select one or more data rows, in order to extract data from all of the cells in the selected row(s).

**Virtualization Type:** In order for the **Data List** control to handle large numbers of records without affecting performance or using excessive memory, the **Data List** control provides two options for controlling when records are rendered visible on the page: *Visible Rows* and *Pagination*. A third option, *None*, will not invoke any virtualization for the data.

- *Visible Rows:* The **Data List** appears as if all of the records were being displayed on a single page, with a vertical scroll bar of the appropriate length. The records themselves, however, are not actually rendered on the page until the vertical scroll bar is moved to the record's location. Each record is rendered individually as its row "scrolls" into view and is discarded when the row scrolls out of view. This is the default selection.
- *Pagination:* When using *Pagination*, the rows displayed on each page are only loaded into the **Data List** when the page containing those rows is displayed, and are discarded when a different page is selected for view, rather than displaying it in a long, unbroken table. End users can elect to display 10, 20,



or 50 records per page when *Pagination* is enabled. Though not the default option, and less resource intensive in terms of rendering records, this selection also adds additional pagination controls on the page.

- **None:** All records are rendered when the **Data List** is displayed. For a large dataset, this selection will use much more memory, since all records have to be retrieved and rendered when the control displays on the page. Thus, for data that contains more than 50 rows, it's best to use *Visible Rows* or *Pagination* to display the data, to minimize the resources needed to display it. Using this selection for large datasets eliminates the memory and resource savings provided by virtualization, and thus largely undercuts the control's purpose, which is to display large datasets with minimal resource usage.

**Width:** Sets the display width of the control on the form, in either percent or pixels. In most cases, a setting of 100% is the appropriate value for this property.

**Height:** Sets the display width of the control on the form, in pixels.

**Sorting:** Enables the sorting feature, to allow users to sort the table data in ascending or descending order by clicking on the column header of the desired column they wish to sort.

**Column Filtering:** Enables users to filter the contents of the list, by adding a text box and filtering criteria to the header of each column.

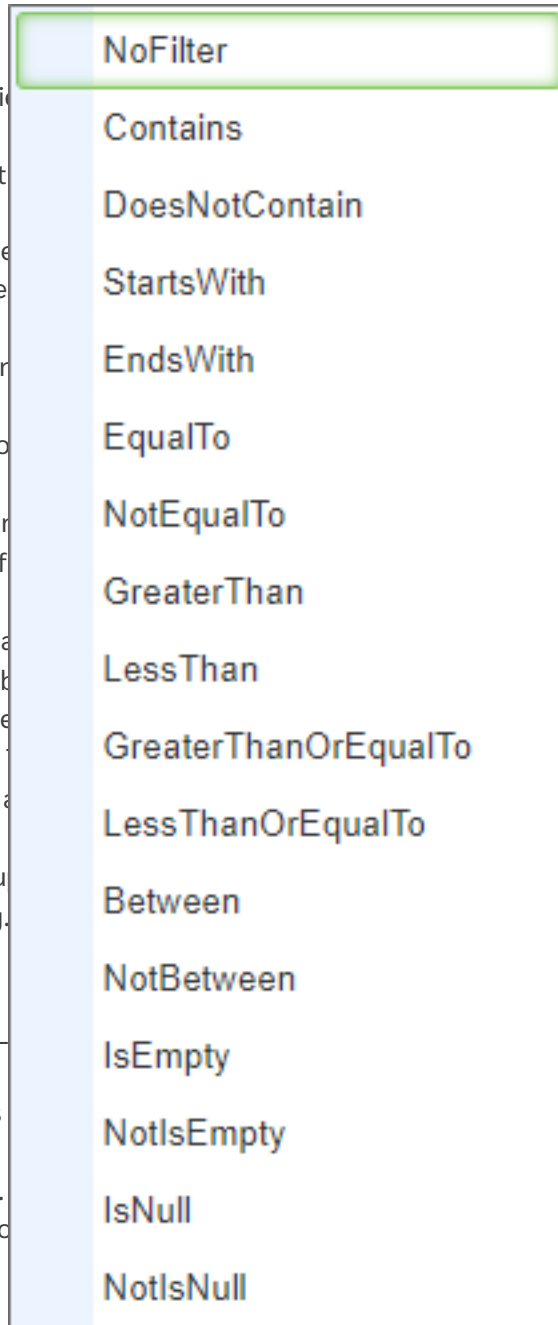
	id	Code	Name	Continent	Country	Region
<input type="checkbox"/>	5	ABY	Southwest Georgia Region		ID	US-GA
<input type="checkbox"/>	91	BWD	Brownwood Regional Airport		CZ	US-TX
<input type="checkbox"/>	93	BWM	Bowman Regional Airport		LY	US-ND
<input type="checkbox"/>	121	CFQ	Creston Valley Regional Airport - Art Sutcliffe Field	NA	FR	CA-BC
<input type="checkbox"/>	157	DCU	Pryor Field Regional Airport	NA	BG	US-AL
<input type="checkbox"/>	159	DDC	Dodge City Regional Airport	NA	BD	US-KS
<input type="checkbox"/>	187	ENW	Kenosha Regional Airport	NA	BG	US-WI
<input type="checkbox"/>	195	EWB	New Bedford Regional Airport	NA	EH	US-MA
<input type="checkbox"/>	222	GGG	East Texas Regional Airport	NA	ZA	US-TX

Filter "Contains" the word "Region", with 29 matching results.

Page size: 20 | 29 items in 2 pages

Users can filter the data by one or more columns by adding the desired filter term in the text box, then selecting a filter type from the adjacent dropdown menu button. Filters are not case-sensitive. The following filter methods are available from the dropdown menu:

- No Filter: Removes the filter from the column.
- Contains: The column data contains the specified text.
- Does Not Contain: The column does not contain the specified text.
- Starts With: The column data begins with the specified text.
- Ends With: The column data ends with the specified text.
- Equal To: The column data exactly matches the specified text.
- Not Equal To: The column data does not exactly match the specified text.
- Greater Than: For numerical values, the column data returns only values greater than the specified number.
- Less Than: For numerical values, the column data returns only values less than the specified number.
- Greater Than or Equal To: For numerical values, the column data returns only values greater than or equal to the specified number.
- Less Than or Equal To: For numerical values, the column data returns only values less than or equal to the specified number.
- Between: For numerical values, when the text box is supplied with two different numbers, separated by a space, e.g., "60 70", the filter will return data where the values are between 60 and 70 (including 60 and 70).
- Not Between: For numerical values, when the text box is supplied with two different numbers, separated by a space, e.g., "60 70", the filter will return data where the values are not between 60 and 70 (including 60 and 70).
- Is Empty: Returns column data where the field value is NULL or has no value.
- Not Is Empty: Returns column data where the field value is not NULL, or has a value.
- Is Null: Returns column data where the field value is NULL.
- Not Is Null: Returns column data where the field value is not NULL.



**Static Header:** Fixes the position of the column headers, so that they don't scroll with the rest of the data when scrolling the list vertically.

**Export to CSV:** Displays an CSV export icon in the upper right corner of the control that, when clicked, will export the list data to a CSV file.

**Export to PDF:** Displays a PDF export icon in the upper right corner of the control that, when clicked, will export the list data to a PDF file.

**Refresh:** This property will display an icon in the upper right corner of the control that, when clicked, will refresh the list data.

**Clear:** This property will display an icon in the upper right corner of the control that, when clicked, will remove all filters and data selections from the list.

**Column Resize:** This property will enable end users to resize the column widths for any column by dragging and dropping a column's borders.

## Configuring the List Data

The Field Properties for the data list control are accessible from the **Form Controls** tab of the Form definition and, for appropriately licensed installations, from the Design Console of the Online Form Designer. One of the **Data List** control's Field Properties, the **Default Value** property, *must* be set to an appropriate **Content List** item, usually a Business Value.

The screenshot shows a dialog box titled "Form Field Properties: Datalist2". It contains the following elements:

- ToolTip:** An empty text input field.
- Friendly Name:** A text input field containing "Datalist 2".
- Default Value:** A dropdown menu showing "Content Item", a text input field containing "Airports", and a button with three dots.
- Set Readonly Options:** A dropdown menu.
- Set Display Options:** A dropdown menu.
- Buttons:** "OK" and "Cancel" buttons at the bottom.

In this example, the **Default Value** is provided via a Business Value named **Airports**. When the Form first opens, **Airports** will be called, and the **Data List** will be automatically populated with the data returned by this Business Value. The data will be cached in memory until the Form is closed. Only clicking the **Refresh** icon, if it's displayed, or closing and re-opening the form, will refresh the data displayed in the **Data List** control.

## Columns

The **Columns** tab of the control properties enables you to specify the columns you which to display in the **Data List** control.

### Data List Control

Data List   Columns   Par

Select which columns are displayed in the Data

- id
- Code
- Name
- Continent
- Country
- Region
- City
- GPS
- Elevation

This tab enables you to select only the columns you wish to display by checking the appropriate field names. At run-time, only the selected columns will be displayed in the Data List control.

	Code	Name	Country	City
<input type="checkbox"/>	AAA	Anaa Airport	UA	N/A
<input type="checkbox"/>	AAO	Anaco Airport	CN	Anaco
<input type="checkbox"/>	AAT	Altay Air Base	US	Altay
<input type="checkbox"/>	ABJ	Port Bouet Airport	CN	Abidjan
<input type="checkbox"/>	ABY	Southwest Georgia Regional Airport	ID	Albany
<input type="checkbox"/>	ACA	General Juan N Alvarez International Airport	HN	Acapulco
<input type="checkbox"/>	ACI	Alderney Airport	NG	Saint Anne
<input type="checkbox"/>	ACI	Alderney Airport	CN	Saint Anne
<input type="checkbox"/>	ADC	Andakombe Airport	UA	Andekombe
<input type="checkbox"/>	ADL	Adelaide International Airport	PL	Adelaide

Page size: 20 826 items in 42 pages

## Parameters

When using a Business Value as a data source, you often need to pass one or more parameters to the Business Value to return the expected data. For the **Data List** control, these parameters can be passed via the **Parameters** tab of the control's properties.

**Data List Control**

Data List   Parameters   Formatting   Field Properties   Com

**Separate Parameters by New Lines**

```
Param1=Value
Param2={:FormField}
```

This tab consists of a text box that will accept one parameter per line, using the syntax:

**ParameterName = Value**

The value portion of the parameter will accept either static text or a system variable, such as a Form field system variable, as shown above.

When the Business Value is invoked to fill the Data List control with data, the parameter(s) entered here will be passed automatically to the Business Value to ensure the correct data is returned.

### *Usage #*

The primary use case for the **Data List** control is to return large datasets from external sources without serious impact on system performance. This data, once retrieved, can be used to select items from the large dataset to transfer into Form fields. You can select a single row in the **Data List** to transfer data into individual form fields, or multiple rows to transfer the data to Array rows.

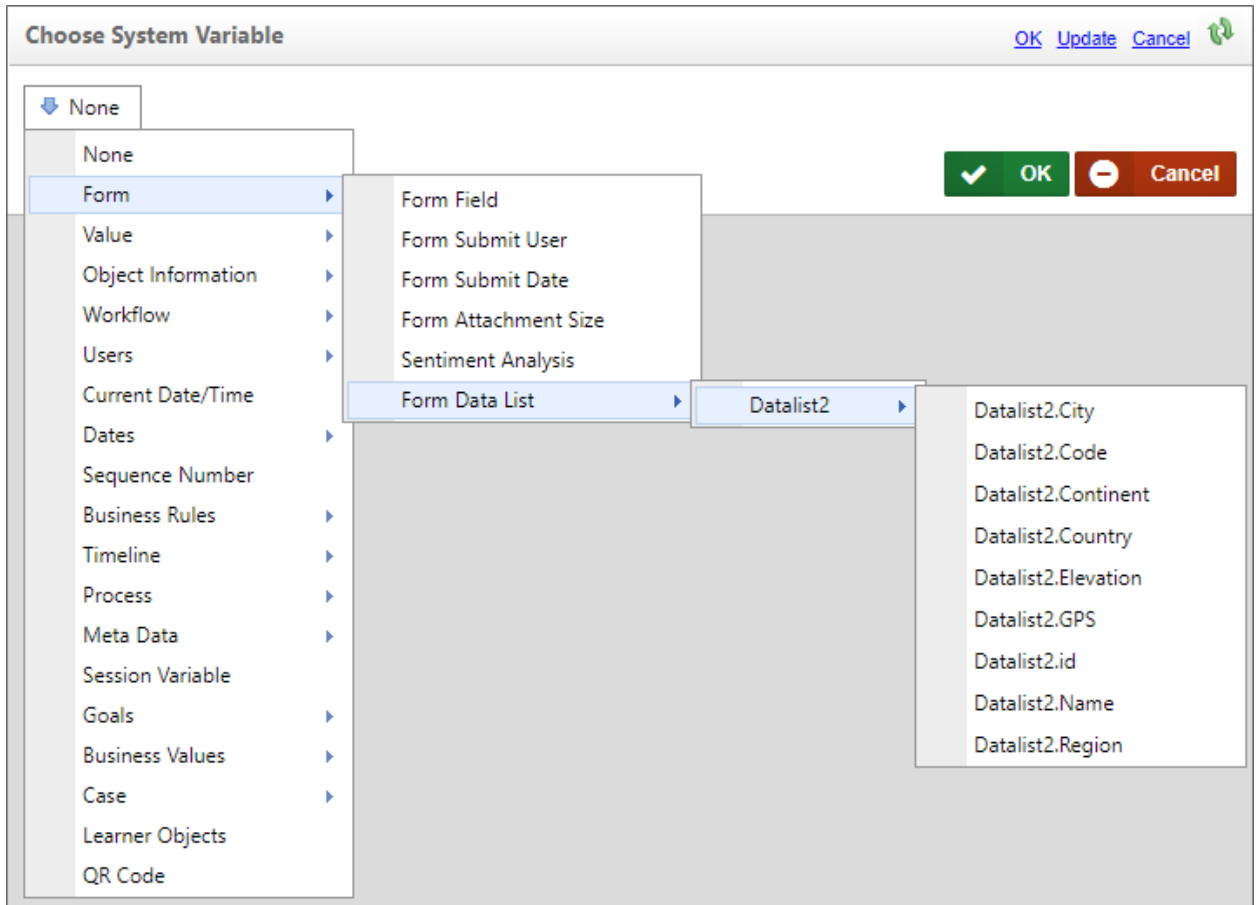
Data is transferred from the **Data List** into Form fields by invoking a Set Form Data action on the form. For example, you can add a **Button** control to a form that, when clicked, will call a Set Form Data action that you configure on the **Set Form Data** tab of the Form definition. We'll use that method in this example.

In this example, we'll use a button on the Form that we'll call **DLButton**, which we'll place immediately below the DataList control, as shown in the Form screenshot below. This button's visible label is configured to display the text, **Add Row Data**.

Immediately below the button is an **Array** containing four fields that will be the target fields for the data selected from the Data List control.

Since we need to retrieve the data via a Set Form Data action, we need to navigate to the **Set Form Data** tab of the Form Definition. Once there, we can create a new action on the **DLButton** event. We can then map the Form fields to the Data List columns we want to use.

The **Data List** control, because of its unique nature, has been added to the **Form** menu of the **Choose System Variable** dialog box with its own **Form Data List** submenu. It is *not* addressed through the **Form Field** menu selection like other Form fields.



The **Form Data List** submenu will *only* appear on Forms that have a **Data List** control. Each **Data List** control will have its own submenu that displays the name of the control, and mousing over it will display a submenu that lists all of the columns in the selected control. In the example above, therefore, the **Datalist2** control is displayed with its associated columns: **Code**, **Continent**, **Country**, etc. This display convention is the same as the one used for Business Values, and operates similarly. Thus, the **Set Form Data** tab can be configured to set the value of Form fields to the value of a selected row or rows in a **Data List** control, just as you would a Business Value.

In this example, when a **Data List** is configured to perform **Row Selection**, the **Set Form Data** tab is configured to set the value of the **AID**, **Airport**, **Location**, and **Alt Array** fields to the value of the **Code**, **Name**, **City**, and **Elevation** columns, respectively, of the selected **Data List** row.

Data List: Datalist2 (Datalist 2) + Add Form Field

AID (Airport ID) ▼		Code ▼	↑ ↓ ✖
Airport ▼		Name ▼	↑ ↓ ✖
Location ▼		City ▼	↑ ↓ ✖
Alt (Altitude) ▼		Elevation ▼	↑ ↓ ✖

With this configuration, if one or more rows of data are selected in the **Data List**, the data from the selected row(s) will be appended to the array fields when the **DLButton** is clicked.

	id	Code	Name	Continent	Country	Region
<input type="checkbox"/>	5	ABY	Southwest Georgia Regional Airport	NA	ID	US-GA
<input type="checkbox"/>	91	BWD	Brownwood Regional Airport	NA	CZ	US-TX
<input checked="" type="checkbox"/>	93	BWM	Bowman Regional Airport	NA	LY	US-ND
<input type="checkbox"/>	121	CFQ	Creston Valley Regional Airport - Art Sutcliffe Field	NA	FR	CA-BC
<input checked="" type="checkbox"/>	157	DCU	Pryor Field Regional Airport	NA	BG	US-AL
<input checked="" type="checkbox"/>	159	DDC	Dodge City Regional Airport	NA	BD	US-KS
<input type="checkbox"/>	187	ENW	Kenosha Regional Airport	NA	BG	US-WI
<input checked="" type="checkbox"/>	195	EWB	New Bedford Regional Airport	NA	EH	US-MA
<input type="checkbox"/>	222	GGG	East Texas Regional Airport	NA	ZA	US-TX

Page size: 20 29 items in 2 pages

Add Row Data

Code	Airport	City	Elevation
BWM	Bowman Regional Airport	Bowman	2965
DCU	Pryor Field Regional Airport	Decatur	592
DDC	Dodge City Regional Airport	Dodge City	2594
EWB	New Bedford Regional Airport	New Bedford	80

As shown above, the four airports selected by the user have been added to the Form, using a set of array fields to store the selected row values.

This feature enables you to take any desired data from the **Data List** control and transfer it to Form fields for saving as Form data.

**Documentation Samples**

**Video Example**



### *Other Control Tools*

You can view the documentation for all tools available in the Online Form Designer by using the Table of Contents on the upper right corner of the page, or by clicking one of the links below.

**[Input Controls](#)**: Controls that are commonly used to collect data, but are a bit less widely used than the basic controls.

**[Other Input Controls](#)**: Additional Input controls, consisting mainly of the different content picker controls.

**[Actions](#)**: These controls enable you to control form actions, like placing buttons, or choosing objects via a picker,

**[Other Controls](#)**: Controls that perform miscellaneous tasks like adding HTML content, or labels.

**[Layout](#)**: Controls that are used to govern the control layout for the template, such as tabs and sections.

**Responsive Layout:** Controls that implement Bootstrap form layout objects.

**Arrays:** Controls that enable to you create and control arrays on the Form.

**Attachments:** Controls that enable you to add and show attachments, such as documents or images, to the Form.

**Form Control Tags:** System Variables used to add controls to a Form, instead of using the UI controls.

## Adding Online Form Designer Controls

The control toolbar enables you to add controls to the page design surface. The toolbar consists of seven tool icons that enables you to insert the most common controls onto the page, followed by a series of dropdown menus from which you can select lesser used controls.



To add a control to the page place the cursor on the page where you want the control to appear, then click on the control you wish to insert. The control will be placed in the desired location, after which you can configure the control's properties.

When a form field is added, it must contain a value in the **Name** field. Default names are used when fields are added (e.g. Input1, Input2); however, BP Logix recommends these default names be changed to a field name that is meaningful in the context of the form. If the **Name** property is empty, the field won't appear on the Form.

The **Name** property's value must meet the following conditions:

- The value must be unique, that is to say, used only once on the Form. If two fields use the same **Name** value, only one of the fields will be added to the Form definition, and an error will be displayed when the Form is opened.
- The value must be less than 64 characters in length.
- The value can't contain any spaces or most special characters. Only letters, numbers, dashes, and underscores are allowable.

While all controls have a **Name** property, the other properties for each control are very different, with each type of control having their own, unique sets of properties. To view detailed information about configuring the different controls that are available in the Online Form Designer, please see the [Form Controls](#) topic.



For Process Director v5.25 and lower, the default text field on some controls was set to 'Text'. It now defaults to empty, to enable the use of null text with icons/images. This includes the [Array Move Up/Down](#) buttons, the [Button](#) control, the [Cancel Process](#) and [Audit](#) buttons.

## Configuring Control Properties

Depending on the version of Process Director you're using, the method for configuring control properties will be different.

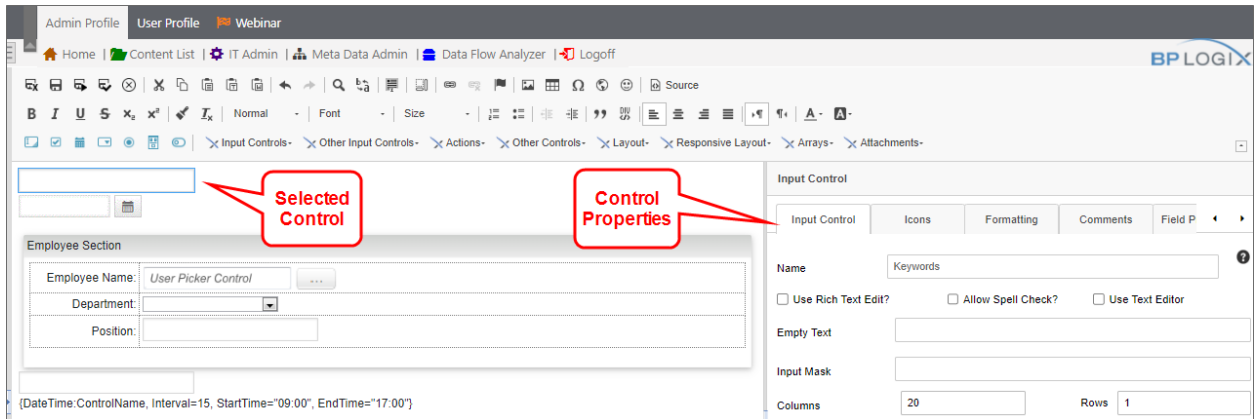
- For Process Director v6.0 and higher installations that use the Subscription license model for Process Director, control properties are displayed in the [Design Console](#), a persistent **Properties** pane that

- appears on the right side of the design surface.
- For all earlier versions of Process Director, or installations that don't use the Subscription license, each control displays its own pop-up **Properties** dialog box.

These two methods of configuring control properties are discussed below.

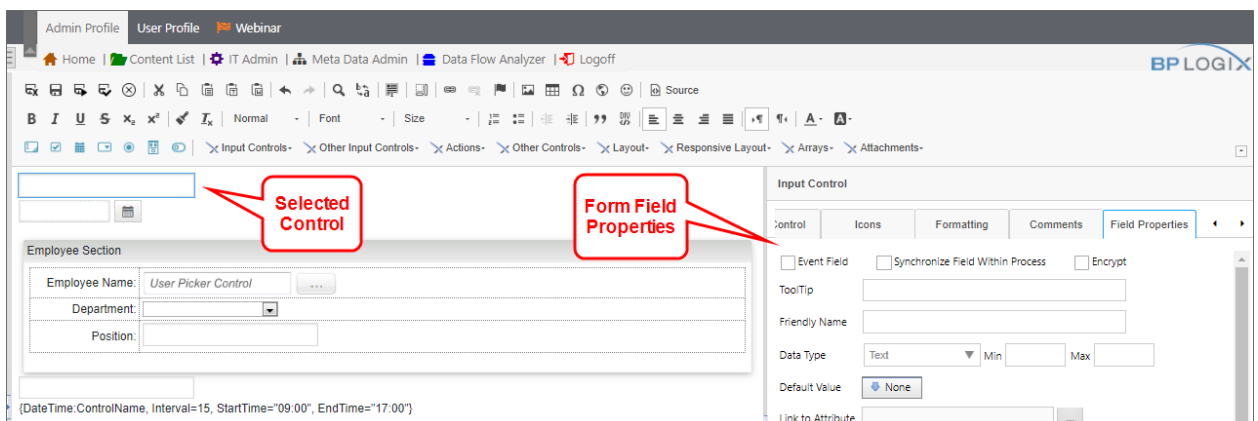
## Design Console (v6.0 and Higher)

When a control is first inserted onto the Form's design surface, or when a Form control is selected by clicking it, a persistent properties pane called the **Design Console** will appear on the Form design surface will display all of the available properties for that control.



The control properties displayed in both the **Design Console** and the **Properties** dialog box in previous versions are exactly the same. For example, the **Input** control shown here will display the same **Input Control**, **Icons**, **Formatting** and **Comments** tabs.

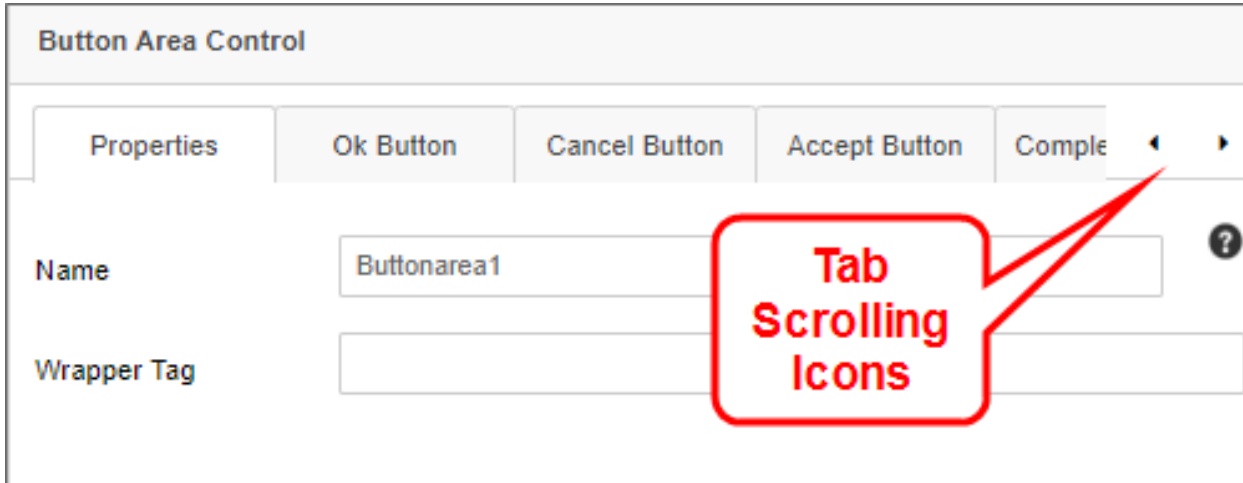
In addition to that, the **Design Console** will, for appropriately licensed version of Process Director v6.0, display the **Field Properties** tab, which displays the field properties that were accessible only through the Form definition's **Form Controls** tab in previous versions. The ability to simultaneously configure both control and field properties significantly reduces the time needed for designing Forms.



Thus, the **Design Console** enables you to configure all of the relevant properties for that control, without having to switch between the Form's design view and Form Controls tab of the Form definition. The **Form**

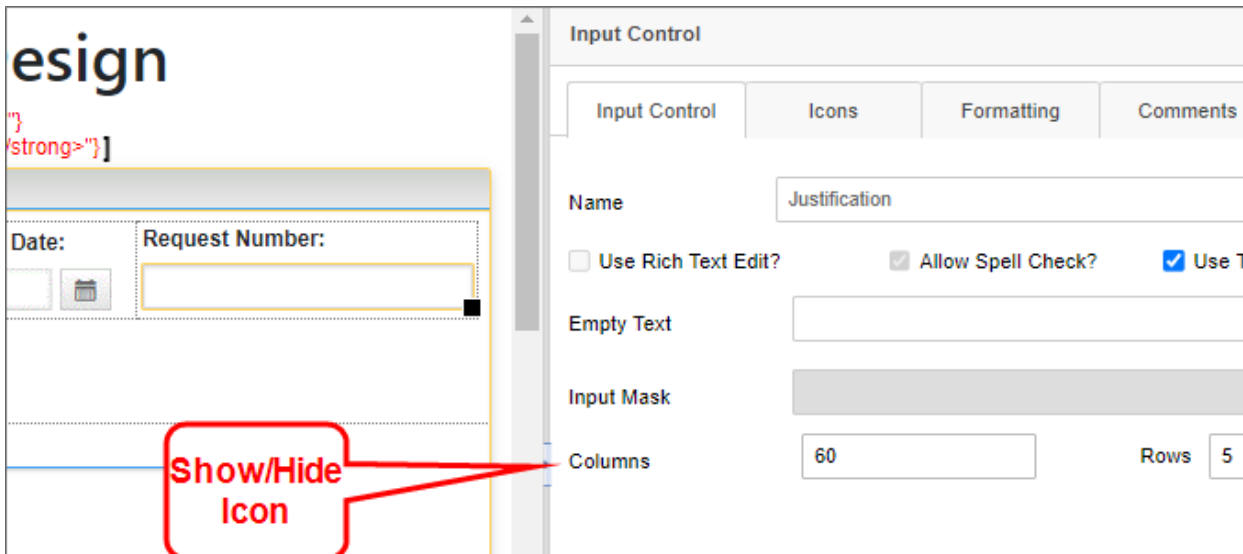
**Controls** tab will still be available in the Form definition, but the somewhat laborious process of switching back and forth while designing the Form will no longer be necessary.

When placing controls on the Form's design surface, some controls, like the **Button Area** control, have a larger number of tabs than can be conveniently viewed in the **Properties** pane. Two scrolling icons on the right side of property tabs enable you to scroll horizontally, left or right, through the array of tabs to display the tab you need to view.



Selecting a form control on the design surface will automatically switch the **Properties** pane to display the properties for the selected control. Unlike the Properties dialog box in previous versions, there is no **OK** button to close and save the properties for each control. Property values are saved automatically with the form.

A small **Show/Hide** icon appears in the center of the divider bar between the **Design Console** and Form's design surface to enable you to close the pane to see your form in full-screen mode. This icon looks and works just like the icon that appears between the folder list and folder contents panes of the **Content List**.



When placing a control onto the Form's design surface, the properties for the control will be displayed automatically in the [Design Console](#). You can view the properties for any other control by simply selecting it on the design surface, and the [Design Console](#) will display its properties automatically.

## Properties Dialog Box

For all versions prior to Process Director v6.0, when a control is inserted on the Form's design surface, a pop-up dialog box will appear that enables you to configure the control's properties. Once you've done so, you can click the **OK** button to save the configuration and close the [Properties](#) dialog box.

You can re-open the [Properties](#) dialog box for any control by simply double-clicking the control whose properties you'd like to configure.

### *Control Tools*

You can view the documentation for all of the Form control tools available in the Online Form Designer by using the Table of Contents on the upper right corner of the page, or by clicking one of the links below.

[Form Controls](#): Overview of the available form controls.

[Basic Controls](#): The most commonly-used form design tools.

[Input Controls](#): Controls that are commonly used to collect data, but are a bit less widely used than the basic controls.

[Other Input Controls](#): Additional Input controls, consisting mainly of the different content picker controls.

[Actions](#): These controls enable you to control form actions, like placing buttons, or choosing objects via a picker,

[Other Controls](#): Controls that perform miscellaneous tasks like adding HTML content, or labels.

[Layout](#): Controls that are used to govern the control layout for the template, such as tabs and sections.

[Responsive Layout](#): Controls that implement Bootstrap form layout objects.

[Arrays](#): Controls that enable to you create and control arrays on the Form.

[Attachments](#): Controls that enable you to add and show attachments, such as documents or images, to the Form.

## Form Control Tags

Form Control Tags are generally not required since they all have analogs in the Online Form Designer; however, they still function should you prefer to use them. There are some exceptions to this general rule, though.

- The [Tab Strip](#) control in the product's UI is limited in the number of tabs that can be configured, so you'll need to use the Form Control Tag to configure a larger number of tabs than are available in the graphical control's properties dialog.
- To display both date and time in [Date Picker](#) controls, and to use custom time increments, you must also use the Form Control Tag instead of the graphical control.
- When using arrays in a form, the responsive Form Control Tag for each column, the [Array Column](#), is required for properly labeling array columns when the responsive form is displayed in smaller view-ports.

Form Control Tags can be placed on a form to provide additional input controls or for conditional formatting and display, or as an alternative to using the graphical controls in the Online Form Builder. The Form Control Tags are optional.

 Though the UI controls have a visual placeholder in the form design UI, all Form controls are actually form control tags, even though the visual Form designers hide the Form Control Tags with the graphical placeholder.

## AddRow

This Form control will create a button a user can click to add row(s) to an array. This is placed outside of the array.

## Properties

PROPERTY NAME	DESCRIPTION
ArrayName	The name of the array this button is attached to.
At	The location to add the new row(s).
ConfirmText	Pops up a confirmation box when a user clicks the button with the specified text, allowing a user to cancel or confirm the action which the button will take.
ImageURL	Sets an optional 24px height image for the button.
Rows	Number of rows to add every time the button is clicked
SmallImage	Sets an optional 16px height image for the button.
Style	To set the style (using any CSS style).
Text	Sets the optional button text.

## Example

```
{AddRow:ControlName,ArrayName=ArrayControlName,Text="Text to Display"}
```

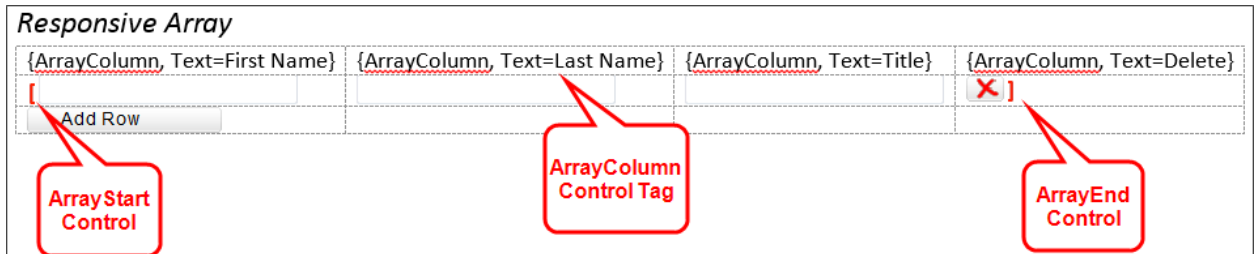
## ArrayColumn

Starting with Process Director v3.78, for better appearance in smaller viewports, especially on mobile devices, you can use the **ArrayColumn** form tag to indicate the columns in an array. When the form is viewed on a small viewport, tables will be displayed so that every cell is on a new line.

The [ResponsiveType custom Variable](#) must be set to true in the vars file in order to enable this feature.

## Implementation

The **ArrayColumn** tag should be placed in the first row of an array table, outside of the **Array** row, as shown below:



On a wider screen, the `ArrayColumn` control will appear as a column header when a user runs the Form:  
On smaller screens, such as phones, the responsive table will now collapse into a single column, so that users don't have to scroll horizontally, and the `ArrayColumn` controls will appear as Field labels:

### Properties

PROPERTY NAME	DESCRIPTION
Alt	Sets an alternate value that will appear in browsers that support tooltip text for HTML "ALT" tags.
Style	An optional CSS style to apply to the column.
Text	Sets the column text that will appear for each control when the Form is reformatted for display on smaller screens.

When associating labels with an array column for accessibility purposes, use the `Column Header` control tag with the appropriate `Text` property to create the label.

### Example

```
{ArrayColumn, Text=Column Name}
```

### ArrayRemoveRow

This Form control will create a button a user can click to remove a specific row from an array. You should place this control directly in an array, so that the button is displayed on each row.

### Properties

PROPERTY NAME	DESCRIPTION
ConfirmText	Pops up a confirmation box when a user clicks the button with the specified text, allowing a user to cancel or confirm the action which the button will take.
ImageURL	Sets an optional 24px height image for the button.
SmallImage	Sets an optional 16px height image for the button.
Style	To set the style (using any CSS style).
Text	Sets the optional button text.

### Example

```
{ArrayRemoveRow:ControlName,Text="Text to display"}
```

### ArrayMoveUp

This Form control will create a button a user can click to move a Row up in the array.

## Properties

PROPERTY NAME	DESCRIPTION
ConfirmText	Pops up a confirmation box when a user clicks the button with the specified text, allowing a user to cancel or confirm the action which the button will take.
ImageURL	Sets an optional 24px height image for the button.
SmallImage	Sets an optional 16px height image for the button.
Style	To set the style (using any CSS style).
Text	Sets the optional button text.

## Example

```
{ArrayMoveUp:ControlName,Text="Text to Display"}
```

## ArrayMoveDown

This Form control will create a button a user can click to move a Row down in the array.

## Properties

PROPERTY NAME	DESCRIPTION
ConfirmText	Pops up a confirmation box when a user clicks the button with the specified text, allowing a user to cancel or confirm the action which the button will take.
ImageURL	Sets an optional 24px height image for the button.
SmallImage	Sets an optional 16px height image for the button.
Style	To set the style (using any CSS style).
Text	Sets the optional button text.

## Example

```
{ArrayMoveDown:ControlName,Text="Text to display"}
```

## Array

This Form control places a repeating template section on a Form. To default the array to a number of rows simply go to the Form properties page and click on the edit link next to the array control that is in the list of controls. Select Value>> Number from the Default Value dropdown and enter the number to default the number of rows to.

## ArrayEnd

This Form control closes the array on a Form. Every **Array** control must have an **ArrayEnd** control to close the array.

## Properties

None



## Example

```
{Array:ControlName}<!--Table Row Contents-->{ArrayEnd}
```

## Attach

This Form control will display a button to allow the user to attach files to the form.

## Properties

PROPERTY NAME	DESCRIPTION
AttachType	<p><b>Form:</b> Attach object(s) directly to Form.</p> <p><b>Workflow:</b> Attach object(s) to the current Workflow instance as a Workflow reference.</p> <p><b>Process Timeline:</b> Attach object(s) to the current Process Timeline instance as an Process Timeline reference.</p> <p><b>Process:</b> Attach object(s) to the current process instance. This the default choice, and what BP Logix recommends for most use cases.</p>
ClipboardImageName	You can optionally set ClipboardImageName to the name you want the uploaded file to be. You can optionally use system variables in this property. If the items on the clipboard are files, the actual file name will be used as the new attachment name.
ConfirmText	Pops up a confirmation box when a user clicks the button with the specified text, allowing a user to cancel or confirm the action which the button will take.
GroupName	Optional name of the group to place the attachment(s) into. You can optionally use system variables in this property
HTMLDesc	Sets the content in the pop-up box after attachment. Can be plain-text or well-formed HTML. Can also include SysVar tags.
ImageURL	Sets an optional 24px height image for the button.
ObjectType	<p><b>Document:</b> Allow user to upload document</p> <p><b>Clipboard:</b> Allow the user to use an item from the clipboard. Microsoft Internet Explorer and the BP Logix Plug-in is required to utilize this option. This option is largely deprecated, and is largely for legacy implementations.</p>
PageTitle	If MFT support is used, you can specify the title of the upload dialog box with this property.
SingleFileUpload	You can optionally set the control to allow only one file to be attached at a time.
SmallImage	Sets an optional 16px height image for the button.
UseMFT	Enables the use of multi-file asynchronous upload using the control. If used, this value will be set to "1".

## Example

```
{Attach:ControlName, AttachType=Process, ObjectType=Document, GroupName=Group, UseMFT=1, PageTitle="Title Text"}
```

## AttachDrop

This Form Control Tag is a different version of the Attach tag. The AttachDrop tag will display an upload area to allow the user to attach files to the form by dragging and dropping the files into the upload area.

## Properties

PROPERTY NAME	DESCRIPTION
AttachType	<p><b>Form:</b> Attach object(s) directly to Form.</p> <p><b>Workflow:</b> Attach object(s) to the current Workflow instance as a Workflow reference.</p> <p><b>Process Timeline:</b> Attach object(s) to the current Process Timeline instance as an Process Timeline reference.</p> <p><b>Process:</b> Attach object(s) to the current Process instance. This the default choice, and what BP Logix recommends for most use cases.</p>
GroupName	Optional name of the group to place the attachment(s) into. You can optionally use system variables in this property
ObjectType	<p><b>Document:</b> Allow user to upload document</p> <p><b>Clipboard:</b> Allow the user to use an item from the clipboard. Microsoft Internet Explorer and the BP Logix Plug-in is required to utilize this option. This option is largely deprecated, and is largely for legacy implementations.</p>
SingleFileUpload	You can optionally set the control to allow only one file to be attached at a time.

## Example

```
{AttachDrop:ControlName, AttachType=Process, ObjectType=Document, GroupName=Group}
```

## AttachKView

This control allows the user to attach an object to a Form instance by browsing for the object using a Knowledge View. To configure the Knowledge View, set the “Default Value” of the control to a “Content Item” in the control's properties in the Form definition, and point to the appropriate Knowledge View.

Default Value

↓ Content Item

Sample KView

...

## Properties

PROPERTY NAME	DESCRIPTION
AttachToParent=[1 0]	If set to 1, this object will be attached to the parent of this Form

PROPERTY NAME	DESCRIPTION
	instance
AttachType	<p><b>Form:</b> Attach object(s) directly to Form.</p> <p><b>Workflow:</b> Attach object(s) to the current Workflow instance as a Workflow reference.</p> <p><b>Process Timeline:</b> Attach object(s) to the current Process Timeline instance as an Process Timeline reference.</p> <p><b>Process:</b> Attach object(s) to the current process instance as a process reference. This the default choice, and what BP Logix recommends for most use cases.</p>
CopyObject=[1 0]	If set to 1, this object will be copied to the new location (leaving the old object alone)
GroupName	Only display objects from the specified group
MoveObject=[1 0]	If set to 1, this object will be moved to a new location (removing the object in its old location)
QS	A querystring to send data to the Knowledge View
Text	Text displayed on the Form button

### Example

```
{AttachKView:ControlName, AttachType=Process, GroupName=Group, Text="Text to Display", MoveObject=0}
```



### AuditViewer

The AuditViewer control enables you to view a history of the controls whose values have changed on the Form, so long as the “Audit Form Changes” checkbox is checked on the Form properties tab in Process Director. Additionally, this control enables you to view Case data changes as well.

### Properties

PROPERTY NAME	DESCRIPTION
ActivityName	The name of the Timeline Activity to audit.
CaseData	If set to 1, this object will display changes to case properties instead of the Form field.
ControlName	The name of the control to audit.
OnlyTooltip	Display the auditing information in a tooltip only.
Step name	The name of the Workflow step to audit.
Text	The text displayed on the audit button control at run time.

## Example

```
{AuditViewer:ControlName, ControlName=MyControl, Text="Text to Display" , OnlyTool-  
tip=1, CaseData=1}
```

## AttachPolling

This control has the Form repeatedly check to see if any new attachments have been added to the form by some external process. This control is only necessary if external API calls or web services are able to add attachments to the Form. This control can also be used with ConceptShare integration by updating the status of documents imported into ConceptShare.

Hiding this control will disable polling. If there are multiple AttachPolling controls, the most recent poll result will be used.

## Properties

PROPERTY NAME	DESCRIPTION
PollingSeconds	The number of seconds between each poll taken. If set to 0, polling will be disabled.

## Example

```
{AttachPolling:ControlName, PollingSeconds=30}
```

## Button

This Form control is used to place a button on a Form. The button will typically be used when calling a Custom Task, or custom script.

## Properties

PROPERTY NAME	DESCRIPTION
Alt	The Alternative Text for the button image.
BackColor	The background color of the button.
Color	The fore color (e.g. text color) of the button.
ConfirmText	Pops up a confirmation box when a user clicks the button with the specified text, allowing a user to cancel or confirm the action which the button will take.
IconNumber	The Icon Number for the icon you wish to display on the button as the button image.
ImageOnly	When set to true, displays only the image for the button, with no text.
ImageURL	Sets an optional 24px height image for the button.
OnClientClick	Used to execute client-side JavaScript or call client JavaScript functions. To prevent the button from causing a Post-back, place a return false; at the end of the JavaScript string.

PROPERTY NAME	DESCRIPTION
SmallImage	Sets an optional 16px height image for the button.
SmallImage	When set to true, uses only the small format for the button image.
Style	To set the style (using any CSS style).
Text	Sets the optional button text.

### Example

```
{Button:ControlName,Text="Text to Display"}
```

### ButtonArea

This Form control is used to control where the buttons for the Form are placed. The **ButtonArea** includes the **OK**, **Cancel** and any task completion buttons set by the process. The actual buttons that are placed in this area are dependent on the configured results for the current process step if any. If this control isn't present on a Form, then the buttons are added to the bottom of the form, by default.

### Properties

PROPERTY NAME	PROPERTY ATTRIBUTES	DESCRIPTION	DEFAULT VALUE
BackColor		Sets the Background color for the Button Area.	
AcceptBackColor		Sets the Background Color of the button.	
AcceptConfirmText		Sets the text of the confirmation dialog the user must click to accept the task	
AcceptColor		Sets the Text Color of the button.	
AcceptGlyphColor		Sets the color of the glyph icon that is displayed on the button.	
AcceptIconNumber		Sets the icon, via the icon number, of the icon to display on the button.	
AcceptImageURL		Sets the URL of the image that appears on the button.	
AcceptText		Sets the text of the button the user clicks to accept the task	
CancelBackColor		Sets the Background Color of the button.	
CancelConfirmText		Pops up a confirmation box when a user clicks the Cancel button with the specified text, allowing a user to return to the form or cancel out of the form.	
CancelColor		Sets the Text Color of the button.	

PROPERTY NAME	PROPERTY ATTRIBUTES	DESCRIPTION	DEFAULT VALUE
CancelGlyphColor		Sets the color of the glyph icon that is displayed on the button.	
CancelIconNumber		Sets the icon, via the icon number, of the icon to display on the button.	
CancelImageURL		Sets the URL of the image that appears on the button.	
CancelShow	True False	Allows the form to show or hide the Cancel button.	True
CancelText		Sets the text for the Cancel button	
CompleteConfirmText		Pops up a confirmation box when a user clicks the Complete button with the specified text, allowing a user to cancel the Complete or continue to submit the form.	
CompleteShow	True False	Allows the form to show or hide (all of) the Complete buttons.	True
OKBackColor		Sets the Background Color of the button.	
OKColor		Sets the Text Color of the button.	
OKConfirmText		Pops up a confirmation box when a user clicks the OK button with the specified text, allowing a user to cancel the OK or continue submitting the form.	
OKGlyphColor		Sets the color of the glyph icon that is displayed on the button.	
OKIconNumber		Sets the icon, via the icon number, of the icon to display on the button.	
OKImageURL		Sets the URL of the image that appears on the button.	
OKShow	True False	Allows the form to show or hide the OK button.	True
OKText		Sets the text for the OK button	

### Example

```
{ButtonArea:ControlName, OKText="OK", CancelText="Cancel"}
```

### Calculate

This Form control calculates an expression and places the result as text on a Form.

## Properties

PROPERTY NAME	DESCRIPTION
FormatString	(optional) The format in which to display the result of the Formula (Defaults to "{0:0.00}") - See <a href="#">Microsoft's documentation</a> on C# string formatting.
Formula	Expression to calculate a numerical value (can accept System Variables)

## Example

```
{Calculate:ControlName,Formula="{@FORM:Control1} * {@FORM:Control2}"}
```

## Cancel

This Form control will place a button on a Form which will cancel or delete the current Form or process.

## Properties

PROPERTY NAME	DESCRIPTION
CancelParentProcess	Determines whether parent processes of the process to which this Form belongs should be canceled. If true, then if this form is part of a process running within another process, both processes will be canceled.
CancelWorkflow	This allows you to cancel the associated Workflow.
ConfirmText	Pops up a confirmation box when a user clicks the button with the specified text, allowing a user to cancel or confirm the action which the button will take.
DeleteWorkflow	This allows you to delete the Workflow.
DeleteForm	This allows you to delete the form.
ID	Optional ID of control.
ImageURL	Sets an optional 24px height image for the button.
SmallImage	Sets an optional 16px height image for the button.
Text	Sets the optional button text.

## Example

To cancel associated Workflow:

```
{Cancel:ControlName, CancelWorkflow=true, Text="Cancel Process"}
```

To just delete this form:

```
{Cancel:ControlName, DeleteForm=true, Text="Delete This Form"}
```

To delete Workflow and form:

```
{Cancel:ControlName, DeleteWorkflow=true, DeleteForm=true, Text="Delete Form and Workflow"}
```

To cancel parent processes when a process is canceled:

```
{Cancel:ControlName, CancelParentProcess=true, Text="Cancel Parent Process?"}
```

## CheckBox

This Form control places a two-state (checked - true/unchecked - false) checkbox on the Form. Useful for yes/no data and enabling/disabling sections on a Form. For more information, see the ASP CheckBox documentation.

## Properties

PROPERTY NAME	DESCRIPTION	DEFAULT VALUE
CssClass	To set the CSS class name for this control.	
Text	(optional) Accompanying label text for the check box	

## Example

This example will cause a Post-back to the server

```
{CheckBox:ControlName, Text="Text to Display"}
```

## ClientSection

This Form control creates a section on the Form. Client sections can be used to visually separate a portion of the Form. Unlike the **Section** control, the **ClientSection** operates entirely within the browser. There are no postbacks when the section is expanded or collapsed. The only option available is the Text option.

```
{ClientSection} <!--Does not support an ID option. You can't give a client section an ID/Name.-->
```

To define the end of a client section, use the TAG `{ClientSectionEnd}`.

The section can optionally be enclosed with an HTML element such as a div or span. A div uses “block” formatting in HTML (so that the section appears on the next line), where the span uses “inline” formatting.

## Properties

WRAPPER TAG	DESCRIPTION	DIV
Text	The text for the “header” of the CollapseSection	

## Example

```
{ClientSection, Text="Text to Display"}
```

```
    This text is within a section.
```

```
{ClientSectionEnd}
```

## CommentLog

This Form control allows a user to place a Comment Log on a Form.



## Properties

PROPERTY NAME	DESCRIPTION	DEFAULT VALUE
Columns	(Optional) The number of columns to display while prompting for a comment	70
ControlName	(Optional) The name of the comment log section. Use this property if you have multiple comment logs on a Form	
Rows	(Optional) The number of rows to display while prompting for a comment	4
Text	(Optional) The text for the button used to add a comment	Add Comment
Width	(Optional) Width of the displayed comments	100%

### Example

Simple **Comment Log** syntax:

```
{CommentLog:ControlName}
```

A **Comment Log** that displays at a specified percentage of the display page width:

```
{CommentLog:ControlName, Width=50%}
```

### ContentPicker

This Form control allows a user to choose an object in the [Content List](#).

### Properties

PROPERTY NAME	PROPERTY ATTRIBUTES	DESCRIPTION	DEFAULT VALUE
DocExtension		(Optional) Limits the user to choose only documents of the specified extension (Type of "Document" only)	
StartingFolder		(Optional) The path to a folder, limiting a user	

PROPERTY NAME	PROPERTY ATTRIBUTES	DESCRIPTION	DEFAULT VALUE
		to choose only objects in that folder and its subfolders	
Type	Document ContentObject Workflow Process Process	(Optional) The type of object (Folder, ContentObject, Script, etc.) to pick.	Folder

### Example

```
{ContentPicker:ControlName,Type=ContentObject} {ContentPicker:InstructionsDoc, Type=Document, DocExtension=pdf}
```

### ControlPicker

This Form control will display a dropdown of all controls on this page.

### Properties

PROPERTY NAME	PROPERTY ATTRIBUTES	DESCRIPTION	DEFAULT VALUE
ControlType	Input, Text Area, Date, Button, Dropdown, Password, Array, Section, Radio, CheckBox, Custom, CustomTaskConfigSection, CustomTaskRunSection, User Picker, Group Picker, Attach, Show Attach, Label.	Limits the type of control to show in the dropdown.	
DropdownPrompt		Optional text to show on the dropdown if no user is selected	
Style	To set the style (using any CSS style).		

### Example

```
{ControlPicker:ControlName, DropdownPrompt="User Prompt to Display"}
```

### Date

This Form control is a date picker control.

## Properties

PROPERTY NAME	DESCRIPTION	DEFAULT VALUE
BlockControl	If set to true, will surround the control within an HTML block element.	true
Style	To set the style (using any CSS style).	

### Example

```
{Date:ControlName}
```

### DateTime

This Form control is a date/time combination picker control. Unlike the standard UI control, this tag enables you to customize both:

- The intervals displayed as the time element of a **DateTime** control. The UI control only displays 1-hour increments.
- The start and end times displayed. The UI control displays 24 hourly increments.

## Properties

PROPERTY NAME	DESCRIPTION	DEFAULT VALUE
EndTime	(Optional) Sets the maximum time (of day) for the preselected picker values. Must exceed the StartTime value. The time can be formatted in the 12-hour ("01:00 PM") or 24-hour ("13:00") time formats.	
Interval	(Optional) The amount of time (in minutes) between preselected picker values. For example, if the interval is set to 30, the control will allow users to select times such as 1:30, 2:00, 3:30, etc., but not times like 1:15, 2:45, 3:27, etc.	
StartTime	(Optional) Sets the beginning time (of day) for the preselected values available for the picker. The time can be formatted in the 12-hour ("01:00 PM") or 24-hour ("13:00") time formats.	

### Example

```
{DateTime:ControlName, Interval=15, StartTime="09:00 AM", EndTime="05:00 PM"}
```

```
{DateTime:ControlName, Interval=15, StartTime="09:00", EndTime="17:00"}
```

Using the above example, the DateTime control will display to the end user as shown below.

6/30/2022 12:00 AM

Time Picker		
9:00 AM	9:15 AM	9:30 AM
9:45 AM	10:00 AM	10:15 AM
10:30 AM	10:45 AM	11:00 AM
11:15 AM	11:30 AM	11:45 AM
12:00 PM	12:15 PM	12:30 PM
12:45 PM	1:00 PM	1:15 PM
1:30 PM	1:45 PM	2:00 PM
2:15 PM	2:30 PM	2:45 PM
3:00 PM	3:15 PM	3:30 PM
3:45 PM	4:00 PM	4:15 PM
4:30 PM	4:45 PM	

### DateDiff

This Form control is used to calculate the difference between 2 dates.

### Properties

PROPERTY NAME	DESCRIPTION	DEFAULT VALUE
Date1	The first date or date / time	
Date2	The second date or date / time	
Type	(Optional) The type of the difference Years	Days

PROPERTY NAME	DESCRIPTION	DEFAULT VALUE
	Months Days BusinessDays Hours BusinessHours Minutes Seconds	
IncludeEndDate	Specifies whether to include the end date as part of the date difference.	False

### Example

```
{DateDiff:ControlName, Date1= {form:my_date1}, Date2= {form:my_date2}, Type=Days, IncludeEndDate=true}
```

### DBConnectorPicker

This Form control will display a Database Connector Picker on the form.

### Properties

PROPERTY NAME	PROPERTY ATTRIBUTES	DESCRIPTION	DEFAULT VALUE
DropdownPrompt		Optional text to show on the dropdown if no connector is selected	

### Example

```
{DBConnectorPicker:ControlName}
```

### DropDown

This Form control puts a dropdown control on the form. For more information, see the ASP DropDownList control documentation.

### Properties

PROPERTY NAME	PROPERTY ATTRIBUTES	DESCRIPTION	DEFAULT VALUE
CssClass		To set the CSS class name for this control.	
Item		Displays an item in the dropdown.	

## Example

```
{DropDown:ControlName, Item="Desired PromptText", Item=DisplayText1:Value1, Item=DisplayText2:Value2}
```

## FormErrorStrings

This Form control is used to identify the area(s) where error messages are displayed on the Form. If this control isn't present on a Form, then the error messages are placed at the top and bottom of the form.

## Properties

None

## Example

```
{FormErrorStrings:ControlName}
```

## FormInfoStrings

This Form control is used to identify the area(s) where informational messages are displayed on the Form. If this control isn't present on a Form, then the informational messages are placed at the top and bottom of the form.

## Properties

None

## Example

```
{FormInfoStrings:ControlName}
```

## Group Picker

This Form control will display a Group Picker on the form.

## Properties

PROPERTY NAME	PROPERTY ATTRIBUTES	DESCRIPTION	DEFAULT VALUE
DropdownPrompt		Optional text to show on the dropdown if no group is selected	
Height		For ListBox PickerType only; sets the height of the ListBox control	
Multiple	True	Allow multiple groups to be selected.	
PickerType	Dropdown Popup ListBox	<b>Dropdown:</b> Use a dropdown control. This is the default option, and will display as a type-ahead dropdown control. <b>Popup:</b> Use a popup control.	

PROPERTY NAME	PROPERTY ATTRIBUTES	DESCRIPTION	DEFAULT VALUE
Width		<b>ListBox:</b> Use a ListBox control. For ListBox PickerType only; sets the width of the ListBox control	

### Example

```
{Group Picker:ControlName, DropdownPrompt="Prompt Text", PickerType="Popup", Multiple="false"}
```

### HotLink

This Form control will allow you to create a hot link on the form. Please note that system variables can be used in these properties.

### Properties

PROPERTY NAME	PROPERTY ATTRIBUTES	DESCRIPTION	DEFAULT VALUE
Style		To set the style for this control.	
Target		HTML target parameter.	
Text		The text that is displayed as a link. This can include System Variables enclosed in {} .	
URL		The URL address to link to. This can include System Variables enclosed in {} .	

### Example

Plain text URL **Hotlink:**

```
{HotLink, URL="https://servername.com/page.htm", Text="Text to display"}
```

System Variable **Hotlink:**

```
{HotLink, URL="{URL_SYSTEM_VARIABLE}", Text="Text to display"}
```

### Icon

This control will display a specified icon on a Form page.

### Properties

PROPERTY NAME	PROPERTY ATTRIBUTES	DESCRIPTION	DEFAULT VALUE
CSSClass		A custom CSS Class to apply	

PROPERTY NAME	PROPERTY ATTRIBUTES	DESCRIPTION	DEFAULT VALUE
		to the icon.	
IconBackColor		The HTML Hexadecimal color of the icon's background.	
IconColor		The HTML Hexadecimal color of the icon.	
IconNumber		The ID Number of the Icon, which is displayed as the icon's tooltip in the <a href="#">Icon Chooser</a> .	
IconRight		When set to true, places the icon on the right side of the control.	
IconSize		The number, in pixels, of the vertical size of the icon to be displayed.	
Tooltip		The tooltip text to display when the user hovers over the icon.	

### Example

```
{ICON:ControlName, IconNumber=0000, IconSize=00, IconColor=#000000, Backcolor=#000000, CSSClass=Classname, tooltip="Tooltip text"}
```

### Image

This Form control will allow you insert an image on the form from a URL address. Please note that system variables can be used in these properties.

### Properties

PROPERTY NAME	PROPERTY ATTRIBUTES	DESCRIPTION	DEFAULT VALUE
Height		(optional) Height of image.	
ImageURL		URL path to image.	
Style		To set the style for this control.	
Target		(optional) HTML target parameter	
URL		(optional) URL to make image a hotlink.	



PROPERTY NAME	PROPERTY ATTRIBUTES	DESCRIPTION	DEFAULT VALUE
Width		(optional) Width of image.	

### Example

```
{Image:ControlName, ImageURL="https://servername/image.jpg", Height="20", Width="30",URL="https://www.bplogix.com", Target="_blank"}
```

### Include

This Form control will allow you to include a file into the Form. This can be used to place common Form logic, controls, or script into shared libraries. We recommend placing custom shared files into the \Program Files\BP Logix\Process Director\website\custom folder, and using the application relative syntax (with the ~ character).

If you add/remove Form controls in the included file, you must Check Out / Check In any Form that included it to have it “see” the updated controls.

### Properties

PROPERTY NAME	PROPERTY ATTRIBUTES	DESCRIPTION	DEFAULT VALUE
file		Local URL to include file.	

### Example

```
{INCLUDE, file="~/Custom/SharedForms/MyForm.ascx"}
```

### Input

This Form control place an input field on the form.

### Properties

PROPERTY NAME	PROPERTY ATTRIBUTES	DESCRIPTION	DEFAULT VALUE
Columns		This will determine the width of the field.	
Rows		This will determine the height of the field. Please note using a height of 1 will create a single-line input field. Using more than 1 will create a multi-line input box.	

### Example

```
{INPUT:ControlName, rows="4", columns="60"}
```

### Label

This Form control places an Label field on the form.

## Properties

PROPERTY NAME	PROPERTY ATTRIBUTES	DESCRIPTION	DEFAULT VALUE
AssociatedControlId		The Control Name of the control to which the Label must be associated.	
Text		The text that will be displayed for the label.	

## Example

```
{LABEL:ControlName, AssociatedControlId="Control Name", Text="Label Display Text"}
```

## KView

This Form control is used to place a Knowledge View on the Form. You can use a button to open a Knowledge View or you can view a Knowledge View inline on the form. To configure the Knowledge View, set the “Default Value” of the control to a “Content Item” in the control's properties in the Form definition, and point to the appropriate Knowledge View.

## Properties

PROPERTY NAME	DESCRIPTION	DEFAULT VALUE
Height	Only for IFRAME tags – The height of the IFRAME.	300px
QS	Optional list of QueryString parameters to pass to the KView. The KView can, for example, use these QueryString parameters in its filters.  Ensure that you've created a filter in the Knowledge View corresponding to each QS filter. You must use the QueryString type on the right side in the Knowledge View filter.	
Text	Only for Popup – The text on the button.	View Knowledge View
Type	Iframe – Displays the KView in an inline IFRAME on the form.  Popup – A button will be shown on the form. When clicked, a pop window showing the KView will be launched	Iframe
Width	Only for IFrame – The width of the IFrame	100%

## Example

In this example, we will use a Knowledge View that has two filter variables, VendorName and VendorContact.

The syntax for the Form tag, including query strings, would be:

```
{KView:ControlName, Text="Text to display", Type=Popup, QS="QueryAttribute1=Value1", QS="QueryAttribute2>{SYSTEM_VARIABLE}"}
```

The syntax for using the same query strings in the **QueryString Params** field of the properties box for the Knowledge View control in the Form Builder is:

```
QueryAttribute1=Value1  
QueryAttribute2>{SYSTEM_VARIABLE}
```

Process Director v3.76 and higher encodes QueryString parameters as "URL encoded" by default when using System Variables.

## ListBox

This Form control is used to place a List Box control on a Form, allowing a user to select more than one entry in the list. This ListBox control can be populated via Custom Tasks, scripts, or with the asp:ListItem tag. Note that commas in a ListBox item value aren't valid. For any ListBox item with a comma in the value, the comma will become a semi-colon upon ListBox creation (this doesn't apply when programmatically adding items).

## Properties

PROPERTY NAME	DESCRIPTION	DEFAULT VALUE
Height	(optional) The height of the ListBox control (e.g., 100px, 15ex, 25%, etc.)	
Items	The collection of items in the ListBox - See ASP ListControl Items property for usage examples.	
Multiple	Allow multiple users to be selected.	True
SelectedValues	A list of selected values in the Items collection	
SelectedValuesString	A comma-separated string representation of the list of selected values in the Items collection	
Width	(optional) The width of the ListBox control (e.g., 200px, 20em, 30%, etc.)	

## Example

```
{ListBox:ControlName, Multiple=true, item=DisplayText1:Value1,
item=DisplayText2:Value2}
```

## Manage Process Users

This Form control is used to place a Manage Users button on a Form. When the Process is running, the **Manage Process Users** button will appear when the form is opened in the context of a task. Otherwise, the control won't display.

## Properties

PROPERTY NAME	DESCRIPTION	DEFAULT VALUE
Name	The Name of the control.	
Text	The Text that will displayed on the form button.	
Add	(Optional) Boolean value to determine whether the Add User function can be managed with the control. Setting this value to "true" will enable the function.	false
Cancel	(Optional) Boolean value to determine whether the Cancel User function can be managed with the control. Setting this value to "true" will enable the function.	false
Reassign	(Optional) Boolean value to determine whether the Reassign User function can be managed with the control. Setting this value to "true" will enable the function.	false
Complete	(Optional) Boolean value to determine whether the Complete User function can be managed with the control. Setting this value to "true" will enable the function.	false
Remove	(Optional) Boolean value to determine whether the Remove User function can be managed with the control. Setting this value to "true" will enable the function.	false
Restart	(Optional) Boolean value to determine whether the Restart User function can be managed with the control. Setting this value to "true" will enable the function.	false
UserGroup	(Optional) Boolean value to determine whether the User Group function can be managed with the control. Setting this value to "true" will enable the	false

PROPERTY NAME	DESCRIPTION	DEFAULT VALUE
	function.	
StepName	(Optional) String value to specify which Workflow Step can be managed with the control. Omitting this parameter will always enable managing the currently running Activity.	
ActivityName	(Optional) String value to specify which Process Timeline Activity can be managed with the control. Omitting this parameter will always enable managing the currently running Activity.	
StartPendingUsersOnly	(Optional) Boolean value to specify that only pending users can be started in the running task.	false

### Example

```
{ManageUsers:ControlName, Text="Button Text", Reassign=true}
```

### Print

This Form control puts a single print button control on the form.

### Properties

PROPERTY NAME	PROPERTY ATTRIBUTES	DESCRIPTION	DEFAULT VALUE
ConfirmText		Pops up a confirmation box when a user clicks the button with the specified text, allowing a user to cancel or confirm the action which the button will take.	
ImageURL		Sets an optional 24px height image for the button.	
SmallImage		Sets an optional 16px height image for the button.	
Style		To set the style for this control.	
Text		Displays the label of the button.	

### Example

```
{Print:ControlName, Text="Button text"}
```

### Radio

This Form control puts single Radio button control on the form.

## Properties

PROPERTY NAME	PROPERTY ATTRIBUTES	DESCRIPTION	DEFAULT VALUE
CssClass		To set the CSS class name for this control.	
Text		Displays the label of the button.	

## Example

```
{Radio:ControlName, Text="DisplayText"}
```

## RadioList

This Form control puts a Radio Group control on the form.

## Properties

PROPERTY NAME	PROPERTY ATTRIBUTES	DESCRIPTION	DEFAULT VALUE
Item		Displays an item in the Radio Group	
RadioBox	true false	When set to true, displays the RadioList as a block.	false
RepeatDirection	Horizontal Vertical	Displays the direction the list will go.	

## Example

```
{RadioList:ControlName, RepeatDirection=Horizontal, item=DisplayText1:Value1,  
item=DisplayText2:Value2, RadioBox=true}
```

## Reauth

In a process task, when the "Re-Authenticate when a user completes the task" option is checked, Process Director will add a user login to the Form. The Reauth tag enables you to choose the location where you'd like the login box to appear on the Form, rather than allowing Process Director to insert it into the default location.

## Example

```
{Reauth:ControlName}
```

## RemoveRow

This Form control will create a button a user can click to remove row(s) to an array.

## Properties

PROPERTY NAME	DESCRIPTION	DEFAULT VALUE
ArrayName	The name of the array this button is attached to.	
At	The location to remove the new row(s).	0 (end of array)
ConfirmText	Pops up a confirmation box when a user clicks the button with the specified text, allowing a user to cancel or confirm the action which the button will take.	
ImageUrl	Sets an optional 24px height image for the button.	
OnClickClient	Used to execute client-side JavaScript or call client JavaScript functions. To prevent the button from causing a Post-back, place a return false; at the end of the JavaScript string.	
Rows	The number of Rows to remove.	1
SmallImage	Sets an optional 16px height image for the button.	
Text	Sets the optional button text.	

### Example

```
{RemoveRow:ControlName,ArrayName="ArrayName"}
```

### RichText

This Form control places a rich text editor on a Form. This allows a user to enter text as well as format it and place links and other rich text controls within the Form control.

### Properties

PROPERTY NAME	DESCRIPTION	DEFAULT VALUE
Height	(optional) The height of the ListBox control (e.g., 100px)	
SpellCheck	(optional) Enables spell check for text box.	False
Width	(optional) The width of the ListBox control (e.g., 200px)	

### Example

This example will cause a Post-back to the server

```
{RichText:ControlName, SpellCheck=True, Height="000px", Width="000px"}
```

### RoutingSlip

Display the [Routing Slip](#) for the process on the form.

## Properties

PROPERTY NAME	DESCRIPTION	PROPERTY ATTRIBUTES	DEFAULT VALUE
ActiveActivityOnly	Should the <b>Routing Slip</b> only display the active Activity?	True/False	False
ActiveStepOnly	Should the <b>Routing Slip</b> only display the active step?	True / False	False
ActivityName	An optional comma-separated list of Activity names to show the <b>Routing Slip</b> in.		
MostRecentInstance	Should the <b>Routing Slip</b> display only the most recent step instance? If this is false, then the <b>Routing Slip</b> will show the users every time a step ran.	True / False	False
Process TimelineName	A partial match for a Process Timeline definition name, restricting the <b>Routing Slip</b> to display only in the matching Process Timelines.		
ReverseOrder	Displays the <b>Routing Slip</b> in reverse chronological order, with the most recent item at the top.		
ShowAllRelatedProcesses	When set to true, this modifier shows the routing for all sibling processes of the main process.	True/False	False
ShowCancelled	Should the <b>Routing Slip</b> display users that have been canceled?	True / False	True
ShowChildren	Show subprocesses in the <b>Routing Slip</b> .	True/False	False



PROPERTY NAME	DESCRIPTION	PROPERTY ATTRIBUTES	DEFAULT VALUE
ShowComments	Should the <b>Routing Slip</b> display the comments?	True / False	True
ShowCompleted	Should the <b>Routing Slip</b> display users that have completed the step?	True / False	True
ShowCompletedOn	Shows the date the task was completed for each participant in the <b>Routing Slip</b> .	True / False	False
ShowHeader	Should the <b>Routing Slip</b> display the header?	True / False	True
ShowInitiator	Should the Process initiator be shown on the <b>Routing Slip</b> ?	True / False	False
ShowNotifyTasks	When set to true, this shows notify-only tasks on the <b>Routing Slip</b> .	True/False	True
ShowNotNeeded	When set to true, hides all users that had a process task completion status of “not-needed” or did not finish.	True/False	True
ShowParents	Show the Parent Process in the <b>Routing Slip</b>	True/False	False
ShowParticipants	Shows the name of each participant. You can use this, for example, to hide the names of the participants, and only show the signature image.	True / False	True
ShowPending	Should the <b>Routing Slip</b> display users that have not began?	True / False	False
ShowReassigned	Should the <b>Routing Slip</b> display users that have been reassigned?	True / False	True

PROPERTY NAME	DESCRIPTION	PROPERTY ATTRIBUTES	DEFAULT VALUE
ShowResult	Shows the Result column in the <b>Routing Slip</b> .	True / False	True
ShowRunning	Should the <b>Routing Slip</b> display users currently running?	True / False	True
ShowSignatures	Should the <b>Routing Slip</b> display user's signatures?	True / False	True
ShowStatus	Shows the Status column in the <b>Routing Slip</b> .	True / False	True
ShowStep	Groups the users in the <b>Routing Slip</b> according to the step in which they participated.	True / False	True
ShowTimedOut	Should the <b>Routing Slip</b> display users that have timed out?	True / False	True
StepName	An optional comma-separated list of steps in which to show the <b>Routing Slip</b> .		
UseDateTime	Optionally shows Date and Time if true	True / False	False
WorkflowName	A partial match for a Workflow definition name, restricting the <b>Routing Slip</b> to display on in the matching Workflows.		

### Example

This example will show the **Routing Slip** for a single step:

```
{RoutingSlip:ControlName, StepName="Step Name"}
```

This example will show the **Routing Slip** for the active step and only show the active users:

```
{RoutingSlip:ControlName, ActiveStepOnly="true", ShowRunning="true", ShowCompleted="false", ShowCancelled="false", ShowTimedOut="false"}
```

### Save

The “save” (close=false) will only appear when the user is viewing the form in a process task. The “save and close” (close=true) will always appear. The “save and close” will add an entry to the users [Task List](#).

## Properties

PROPERTY NAME	DESCRIPTION	DEFAULT VALUE
Close	The Close parameter can be set to true or false.	True
ConfirmText	Pops up a confirmation box when a user clicks the button with the specified text, allowing a user to cancel or confirm the action which the button will take.	
ImageURL	Sets an optional 24px height image for the button.	
SmallImage	Sets an optional 16px height image for the button.	
Style	To set the style for this control.	
Text	Sets the optional button text.	

### Example

This example will cause a Post-back to the server

```
{Save:ControlName, Text="Button Text", Close="True"}
```

### Section

This Form control is used to create a group or section of controls and text on a Form. This section can be used to apply formatting, required settings, or enabling/visibility rules to all elements in a section.

## Properties

PROPERTY NAME	DESCRIPTION	DEFAULT VALUE
BodyCssClass	To set the CSS class name for the body of the section.	
BodyStyle	To set the style (using any CSS style) for the body of the section.	
CanCollapse	Set this property to “true” makes this section a collapsible section.	False
CollapseImageURL	Optional URL to image to use for Collapse	
Expanded	Set to true to have the control viewed in the Expanded state initially. Set to false to have the control viewed collapsed initially.	True
ExpandImageURL	Optional URL to image to use for Expand	
HeaderCssClass	To set the CSS class name for the header.	
HeaderStyle	To set the style (using any CSS style) for the header.	
Text	The text for the “header” of the CollapseSection	
WrapperTag	The section can optionally be enclosed with an HTML element such as a div or span. A div uses “block” formatting in HTML (so that the section appears on the next line), where the span uses “inline” formatting.	Div

## Example

This sample will create a section on a form to collect addresses. The section will be surrounded with an HTML DIV block.

```
{Section:ControlName, CanCollapse=True}
<!--Some Form controls...-->
{SectionEnd}
```

This sample will create a section that flows “inline” with the surrounding HTML.

```
<!--Some HTML content--> {Section:ControlName, WrapperTag="span"} <!--Some more Form
controls and HTML...--> {SectionEnd} <!--Some content after the inline section-->
```

## ShowAttach

This Form control will display a table showing the attachments that match the desired criteria.

## Properties

PROPERTY NAME	PROPERTY ATTRIBUTES	DESCRIPTION	DEFAULT VALUE
AttachType	Form Workflow Process Process	Timeline <b>Form:</b> Show object(s) attached to Form. <b>Workflow:</b> Show object (s) attached to the current Workflow instance as a Workflow reference. <b>Process Timeline:</b> Show object (s) attached to the current Process Timeline instance as a Process Timeline reference. <b>Process:</b> Show object (s) attached to the current process.	Process
CustomScript		A custom snippet of JavaScript code that you'd like to run when the form attachment is displayed. E.g.: CustomScript="alert ("Hi. This is an alert for attachment name {OBJ_NAME}")"	
CustomText		A custom text string that you'd like to show when the	

PROPERTY NAME	PROPERTY ATTRIBUTES	DESCRIPTION	DEFAULT VALUE
		form attachment is displayed. E.g.: <code>CustomText="Attachment Info: id= {OBJECT_ID}, group: {FORM_REF_GROUP}, name: {OBJ_NAME}, name: {OBJ_TYPE}, formdata: {SomeFieldName}"</code>	
GroupName		Optional name of the group to filter the shown objects	
NameAsView	True False	Allows the attachment name to be a hot link. Same functionality as the ShowView property.	False
ObjectType	Document Form NotSet	Only shows documents Only shows form instances Shows all types of objects	NotSet
PageTitle		If MFT support is used, you can specify the title of the upload dialog box with this property.	
ShowDate	True False	Show the date for each attachment?	True
ShowDescription	True False	Show the attachment description.	False
ShowDownload	True False	Show the Download link for each document attachment?	True
ShowEdit	True False	Show the Edit link for each document attachment?	False
ShowModifyDate	True False	Show the last update date for each attachment?	False
ShowModifyUser	True False	Show the last user that modified for each attachment?	False
ShowRemove	True	Show the Remove link for each attachment?	True

PROPERTY NAME	PROPERTY ATTRIBUTES	DESCRIPTION	DEFAULT VALUE
	False		
ShowUser	True False	Show the user for each attachment?	True
ShowView	True False	Show the View link for each attachment?	True
SortBy	Name UpdateTime CreateTime	Enables attached items to be sorted when shown on the Form.	Name
SortType	Ascending Descending	Determines the sort order to be applied to the list of attached items.	Ascending
Text		Optional text to display before the list of attachments	
UseMFT	1	Enables the use of multi-file asynchronous upload using the control. If used, this value will be set to "1".	
ViewInline	True False	Allows the user to select a document to view inline on the Form in an IFRAME	False
ViewInlineHeight		Optional parameter to set the height of the IFRAME for the selected document. Use any HTML compatible string such as 300px.	200px

### Example

```
{ShowAttach:ControlName, GroupName="Group", ShowEdit=True, UseMFT=1, PageTitle="Title Text"}
```

### ShowAttachKView

This Form control will add an IFRAME to the Form showing a Knowledge View of attached objects. To configure the Knowledge View, set the “Default Value” of the control to a “Content Item” in the control's properties in the Form definition, and point to the appropriate Knowledge View.

Default Value

## Properties

PROPERTY NAME	PROPERTY ATTRIBUTES	DESCRIPTION	DEFAULT VALUE
AttachType	Form Workflow Process Process	Timeline <b>Form:</b> Show object(s) attached to Form. <b>Workflow:</b> Show object (s) attached to the current Workflow instance as a Workflow reference. <b>Process Timeline:</b> Show object (s) attached to the current Process Timeline instance as a Process Timeline reference. <b>Process:</b> Show object (s) attached to the current process instance.	Process
GroupName		Specifies to which group objects shown on the Knowledge View should be restricted.	
Height		Height of the Iframe, in pixels.	
ShowParents		If set the 1, the Knowledge View will show the parents of the object.	Null (parent isn't shown)
Width		Width of the Iframe in pixels.	

### Example

```
{ShowAttachKView:ControlName, Height=000, Width=000}
```

### SignatureComments

This Form control will display a textbox on the form to enter task comments.

### Properties

All properties are optional.

PROPERTY NAME	DESCRIPTION	DEFAULT VALUE
Columns	Sets the width of the textbox.	70
InitiatorComments	Indicates whether tho allow the initiator/submitter to provide comments.	

PROPERTY NAME	DESCRIPTION	DEFAULT VALUE
Rows	Sets the height of the textbox.	4
Style	To set the style for this control.	
Text	Sets the optional default text in the textbox.	

### Example

```
{SignatureComments:ControlName, Rows=00, Columns=00, Style="CSS style directives", Text="Text to display", InitiatorComments=1}
```

### Sum

This Form control sums all items of a column in an array. Please note the ID is optional as it can be a system variable.

### Properties

PROPERTY NAME	DESCRIPTION
Column	The column in an array that you'd like to sum up.

### Example

```
{Sum:ControlName, Column=ColumnNameToSum}
```

### Switch

This Form control is a true/false control that operates much like a check box, though with a configurable appearance.

### Properties

PROPERTY NAME	DESCRIPTION
SwitchType	The type of switch presented. The following values are available: Flat, Flip, iOS, Light, and Skewed)
TagOff	The text displayed when switch type is Flip or Skewed and switch is off. The default text is "No".
TagOn	The text displayed when switch type is Flip or Skewed and switch is on. The default text is "Yes".

### Example

```
{Switch:ControlName, SwitchType=Flip, TagOff="OffText", TagOn="OnText"}
```

### TabStrip

This Form control is a **TabStrip** for items. The **TabStrip** must have an id. The **TabStrip** will be listed in the Form controls. Set the default value to the Tab ID to display first.



## Properties

PROPERTY NAME	DESCRIPTION
Tab	A list of names of the tabs in this TabStrip. This is typically a list of Tabs. These names correspond to the TabContent id. To create an id and a different tab name see the following example.
UseVertical	For Process Director v5.12 and higher, setting this property to "1" will array the tabs vertically, rather than horizontally. The default value for this property is "0".

### Example

```
{TabStrip:ControlName, Tab=Tab1:Tab Name 1, Tab=Tab2:Tab Name 2, UseVertical=1}
    {TabContent:Tab1}<!--tab1 form data here-->{TabContentEnd}
    {TabContent:Tab2}<!--tab2 form data here-->{TabContentEnd}
{TabStripEnd}
```

### TabContent

This Form control is one tab of the TabStrip. The TabContent must have an id. This id must match the "Tab" parameter of the TabStrip Form control.

### Example

```
{TabStrip:ControlName, Tab=Tab1:Tab Name 1, Tab=Tab2:Tab Name 2, UseVertical=1}
    {TabContent:Tab1}<!--tab1 form data here-->{TabContentEnd}
    {TabContent:Tab2}<!--tab2 form data here-->{TabContentEnd}
{TabStripEnd}
```

### Time

This Form control places a picker for selecting time values on a Form.

### Properties

PROPERTY NAME	DESCRIPTION	DEFAULT VALUE
EndTime	(Optional) Sets the maximum time (of day) for the preselected picker values. Must exceed the StartTime value.	
Interval	(Optional) The amount of time (in minutes) between preselected picker values.	
StartTime	(Optional) Sets the beginning time (of day) for the preselected values available for the picker.	

### Example

```
{Time:ControlName, StartTime="11AM", EndTime="3:45pm", Interval="15"}
```

### UserPicker

This Form control will display a User Picker on the form.

## Properties

PROPERTY NAME	PROPERTY ATTRIBUTES	DESCRIPTION	DEFAULT VALUE
DropdownPrompt		Optional text to show on the dropdown if no user is selected	
Height		For ListBox PickerType only; sets the height of the ListBox control	
InGroup		Optional filter to only show users that are members of the specified group. You can optionally use system variables in this property	
Multiple	True	Allow multiple users to be selected.	
PickerType	Dropdown PopupForce ListBox	Dropdown – use a dropdown control  PopupForce – force the use of the legacy popup control, instead of the default Typeahead.  ListBox – use a ListBox control.	
Width		For ListBox PickerType only; sets the width of the ListBox control	

### Example

```
{UserPicker:ControlName, DropdownPrompt="Prompt Text", PickerType="PopupForce", Multiple="true", InGroup="Group"}
```

### Other Control Tools

You can view the documentation for all tools available in the Online Form Designer by using the Table of Contents on the upper right corner of the page, or by clicking one of the links below.

**[Input Controls:](#)** Controls that are commonly used to collect data, but are a bit less widely used than the basic controls.

**[Other Input Controls:](#)** Additional Input controls, consisting mainly of the different content picker controls.

**[Actions:](#)** These controls enable you to control form actions, like placing buttons, or choosing objects via a picker,

**Other Controls:** Controls that perform miscellaneous tasks like adding HTML content, or labels.

**Layout:** Controls that are used to govern the control layout for the template, such as tabs and sections.

**Responsive Layout:** Controls that implement Bootstrap form layout objects.

**Arrays:** Controls that enable to you create and control arrays on the Form.

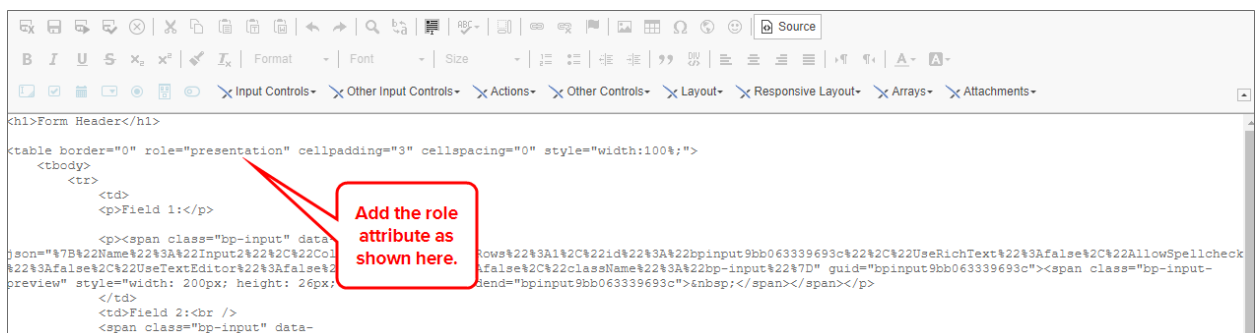
**Attachments:** Controls that enable you to add and show attachments, such as documents or images, to the Form.

## Using Tables in Forms

While there are several ways to organize controls on a page, many users, especially those not familiar with HTML/CSS, will choose to use a table to provide the layout of the form page. In most cases, this isn't an issue. Process Director will automatically refactor tables to display responsively as the width of the user's viewport changes. On screens with narrow widths, tables will be refactored to display in a single column. This responsiveness change happens automatically, and requires no special configuration.

This use of tables may, however, create an accessibility issue. For tables inserted manually into a Form, the container table markup & structure may be parsed by the browser before the Form field content, and some rows may be announced as 'table row blank' to the user, because the Form container hasn't yet been parsed. This issue can result in screen readers being unable to convey the table contents correctly to users of screen reader software/devices for the disabled.

To fix this, you must set the table as a "presentation" table to change its role from being parsed as a data table to merely serving as a layout container. To do this, click the **Source** button on the top toolbar to open the Form's HTML source. Find the TABLE tag in the HTML source, and add the attribute, `role="presentation"`, to the HTML tag.



Once you've done so, click the **Source** button again to return to the visual designer, then save the Form. This should resolve the accessibility issues that might arise from the table.

As the form designer, you must test tables for accessibility, and add the "presentation" role to tables where appropriate.

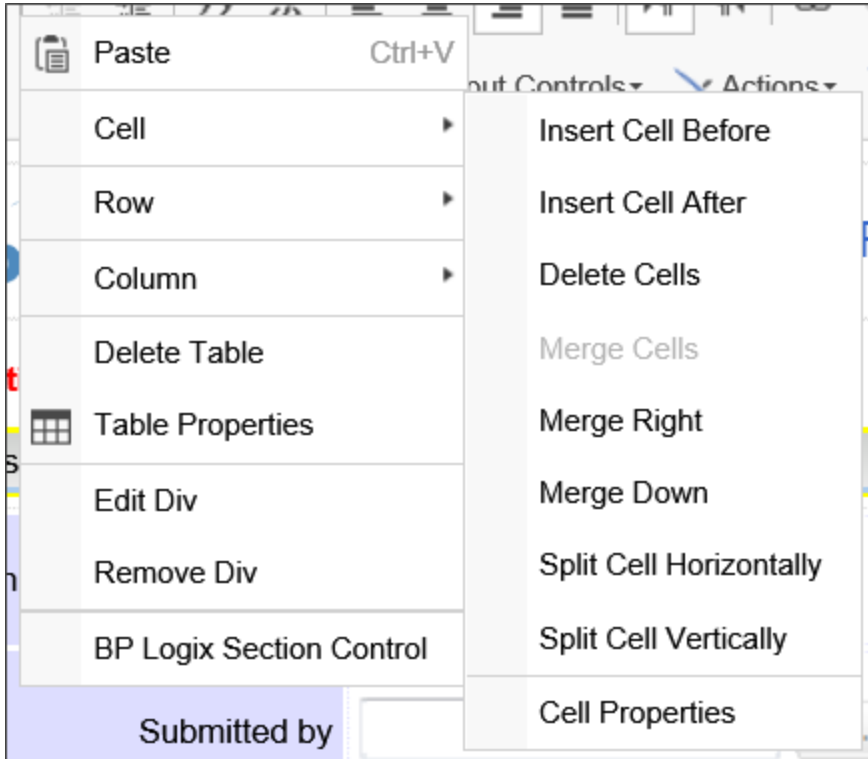
Process Director, in some cases, also inserts tables automatically, to control the layout of some controls. Process Director automatically adds the appropriate "role" attribute for tables it inserts into forms.

## Configuring Other Form Objects

In addition to the Form controls that you can place on a template, you can also insert other objects, such as images or tables, onto the Form's design surface. Many of these objects have their own properties that

can be configured. These properties can usually be accessed by right-clicking on the object, then selecting the properties you'd like to set from a pop-up menu that appears.

For instance, if you insert a table onto the template, right-clicking anywhere in the table presents you with a pop-up menu from which many different table or cell properties can be configured.



Selecting **Table Properties** or **Cell Properties**, for example, will open a **Properties** dialog box from which you can set properties such as borders, alignment, background colors, and others. You can also add or delete rows and columns, merge and split cells, etc.

Right-clicking on any object that has configurable properties will present you with the appropriate menus to configure the object.

## Word Form Builder (Legacy)




BP Logix **STRONGLY** recommends that **ALL** new forms be created with the Online Form Designer (OFD), and that customers migrate existing forms to the OFD format using the conversion tool provided on the Edit tab of the Form definition. The options to create a new Word-based Form were removed from the product in v5.44.900.

The [Online Form Designer \(OFD\)](#) is the primary tool for designing Forms in Process Director. Prior to Process Director v5.44.1000, the Word Form builder was available, along with an ActiveX plug-in for Microsoft Word. With the use of Internet Explorer, the Word Plug-in enabled you to use Microsoft Word as a Form-building tool, with round-trip editing via Internet Explorer. On 15 June 2022, Microsoft ended all support for both Internet Explorer and ActiveX, making it extremely difficult for BP Logix to continue

support Word as a Form design tool. Thus, in v5.44.900, the ability to create new Forms using the Word Form Builder was removed from the product. Additionally, all other references to the Word Form Builder were removed from the product's documentation. Please refer to the [archived legacy documentation](#) for the product to view these Word-based topics.

Word-based forms can be easily converted to the OFD format, using the **Convert to an HTML Form** button that appears on the **Edit** tab of the Form definition. BP Logix recommends that you convert your existing Word-Based forms to the OFD format, if feasible.

 Existing Word-based forms will continue work correctly in Process Director, so you won't lose the functionality of your existing forms. Those existing forms were already converted to HTML forms when they were checked in. Your users will still be able to access them normally.

Microsoft has eliminated the technologies that enable Word-based forms; therefore, creating, editing, and maintaining them will now be much more difficult. Creating and editing Word-based forms requires Internet Explorer and ActiveX running on both the server machine and any workstation used to edit the Word forms. Assuming that you're willing to follow some specific technical restrictions, you can, if you choose, continue maintaining Word forms in your installation.

The server on which Process Director is installed must:

1. Be running a Windows operating system that supports Internet Explorer.
2. Have Internet Explorer installed on the system.
3. Have Automatic Updates turned **OFF**, otherwise Microsoft will automatically remove Internet Explorer and ActiveX on every update. Any server updates you need will have to be installed *manually*, taking care to exclude updates that would remove Internet Explorer.

The client system on which you design forms must similarly:

1. Be running a Windows operating system that supports Internet Explorer.
2. Have Internet Explorer installed on the system.
3. Have Automatic Updates turned **OFF**, just as on the server, and for the same reasons.
4. Have only a 32-bit version of Microsoft Word 2016 or earlier installed.
5. Have the BP Logix Word Plug-in installed.

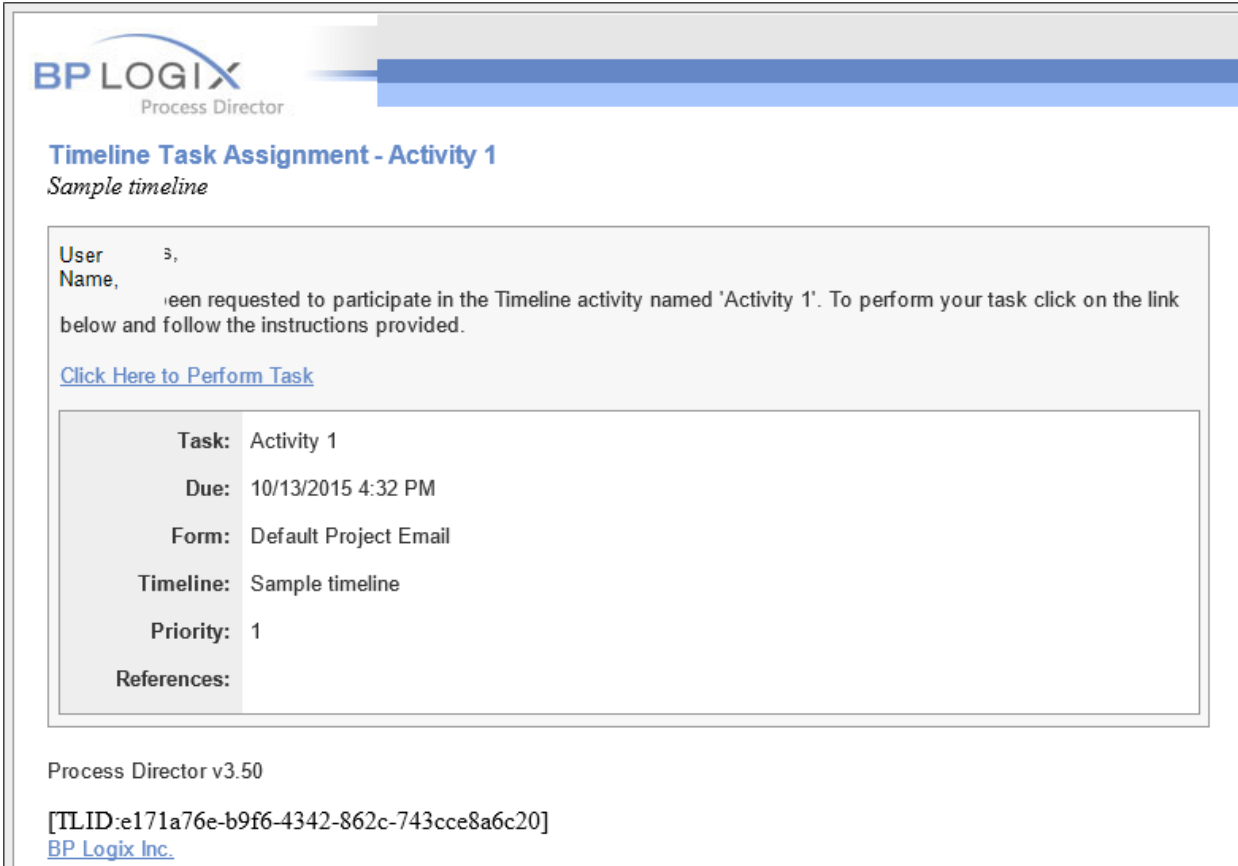
Turning off Automatic Updates may have security or other implications. Similarly, maintaining the appropriate server and client operating systems, along with older versions of Microsoft Word may also have implications for your network's administration. You should discuss these issues with the appropriate IT/IS personnel.

Microsoft's elimination of these technologies will also impact the ability of BP Logix to provide technical assistance with issues that arise from the use of Word-based forms. BP Logix will make every reasonable effort to provide technical support for Word-based Forms, but Microsoft's elimination of the Word Form Builder's supporting technologies imposes the same limits on BP Logix as it imposes on you.

## Email Templates

Process Director sends automated email notifications when certain process events occur. A process definition can specify a custom email template as the default email template for the process by referencing an email template stored in the [Content List](#). Each Process activity can also specify an email template for the individual activity, thus enabling a single process to use many email templates. You can, therefore, create as many email templates as you need for a process in the [Content List](#).

When an email needs to be sent, the system checks for the existence of the appropriate email file listed in the Email Template field of the step and if one isn't found, the default email file is used instead.



The screenshot shows an email template interface for 'Timeline Task Assignment - Activity 1'. It includes a header with the BP LOGIX logo and 'Process Director' text. The main content area contains a message to a user named 'S', asking them to participate in a task. A link 'Click Here to Perform Task' is provided. Below the message is a table with task details: Task: Activity 1, Due: 10/13/2015 4:32 PM, Form: Default Project Email, Timeline: Sample timeline, Priority: 1, and References. The footer includes 'Process Director v3.50', a unique identifier [TLID:e171a76e-b9f6-4342-862c-743cce8a6c20], and the BP Logix Inc. logo.

**BP LOGIX**  
Process Director

### Timeline Task Assignment - Activity 1

*Sample timeline*

User S,  
Name, have been requested to participate in the Timeline activity named 'Activity 1'. To perform your task click on the link below and follow the instructions provided.

[Click Here to Perform Task](#)

<b>Task:</b>	Activity 1
<b>Due:</b>	10/13/2015 4:32 PM
<b>Form:</b>	Default Project Email
<b>Timeline:</b>	Sample timeline
<b>Priority:</b>	1
<b>References:</b>	

Process Director v3.50  
[TLID:e171a76e-b9f6-4342-862c-743cce8a6c20]  
[BP Logix Inc.](#)

This default email template is built directly into the product, and can't be edited or modified in any way. The best choice, therefore, is to create custom email templates for your applications.

### See Also:

[Creating Email Templates](#)

[Using Multiple EmailData Controls in a Template](#)

[Task Completion Control via Email](#)

## Creating and Customizing Email Templates

Creating an email template requires creating a new Form definition within Process Director. In the [Options](#) section of the [Create Form](#) dialog, ensure that you check the [Email Template](#) checkbox. If this property

isn't checked, Process Director will create a normal Form definition, which *can't* be used as an email template.

The screenshot shows the 'Create Form' dialog box. At the top, there is a dropdown menu labeled 'Use Online Form Designer'. Below it are two text input fields: 'Form Name' and 'Description'. A section titled 'Options' contains four checkboxes: 'Form Custom Task', 'Workflow Custom Task', 'Timeline Custom Task', and 'Rule Custom Task', all of which are unchecked. The 'Email Template' checkbox is checked and is highlighted with a red circle. At the bottom right, there are two buttons: a green 'OK' button and a red 'Cancel' button.

You can also use a custom ASCX file for your email template, just like you can when creating a regular Form definition. Simply select the option you desire from the dropdown marked "Online Form Designer". In nearly all use cases, the default choice of the Online Form Designer is the simplest option, but you have the option to create a custom ASCX control in ASP.NET using Visual Studio (assuming you have this license option).

Your email template is now stored in the [Content List](#). You can edit the template by selecting the template to edit and then select the Edit tab. You can make changes to your template in your preferred editor.

## Email Templates and Controls #

While email templates are technically a type of Process Director Form, are generated by the Process Director server, and use the Online Form Designer to create their content and styling, email templates have limitations that normal data input Forms do not. An email template can only generate the plain text of an email message. Unlike a normal Form, an email template cannot use any data input controls. An email cannot display text, textarea, dropdown, or any other type of input control, nor can an email store form field information generated by end users in any way. Why is this?

**Email messages are plain text messages.** Even HTML email messages, with their styling and other HTML display elements, are simple text messages that just happen to include HTML markup as part of the plain text. Your email client, like Outlook, can convert the HTML markup to display the desired styling when you view the email message, but this is generated from the plain text of the email message itself. All email messages are nothing more than simple text files.

The only controls that can be used on an Email Template are controls that can be converted to plain text when the email is generated. This limitation enables only a limited set of controls like the [System Variable](#) control, which gets converted to plain text when the message is generated, or the [Hotlink](#) control that also gets converted to an HTML text hyperlink. A special control used **only** on email templates, the

**Email Data** control, also converts to plain text to provide the subject and return addresses for the email message at run-time. All of the other data entry controls, such as the **Input**, **Dropdown**, **Checkbox**, etc., cannot function in an Email template. Data entry controls will not work in an email template, and users cannot provide any data to Process Director from an email message.

Moreover, email messages have no memory or user context. They can't accept or store user input, and can't even tell whether the person who opens and reads the email is the originally intended recipient. Once sent, a email template is just a text message, and has no other content outside the actual text of a message.

**Use of images in email templates is not recommended**, as most email clients strip all images from email messages and prevent them from displaying by default. There are methods to override this behavior for different email clients. Please see [Microsoft's explanation of this behavior](#) on their web site, as well as the Apple's explanation for similar behavior on the default Mail client on both [iPhone/Ipad devices](#), and on [the Macintosh](#).

If, despite the default image blocking on email clients, you decide to use an image, such as a logo, on an email template, you must use a fully qualified URL as the image URL. Partial or relative URLs will not be able to properly access the image. E.g., a URL like <https://www.server.com/images/imagename.jpg> will work, but a partial or relative URL like </images/imagename.jpg> will not display the image.

## Email Variables #

Variable substitutions are performed when an email file is processed for a process request. System variables can be added to the email, either by typing them in plain text, or by using the **System Variable** control. To see the complete list of system variables, please refer to the System Variables guide. Note that not all system variables are available for every event that generates an email.

The **Subject**, **From Email**, **Mail Priority**, **From Display**, **Send As Attachment**, **Group Name** and **Cancel Email** properties are contained in each email template as properties of the **Email Data** control.

## EmailData Control #

The Email Data object enables you to customize various aspects of notification emails such as the email subject, reply-to addresses, etc. Inserting the Email Data control produces this object in your Form:



By double-clicking on it, you can open its properties dialog box , shown below, to edit its properties.



The screenshot shows a dialog box titled "Email Data Control" with a close button (X) in the top right corner. The dialog has three tabs: "Email Data Control" (which is selected and highlighted with a blue border), "Additional Properties", and "Comments". Below the tabs, there are several input fields and a dropdown menu:

- Name:** A text input field containing "Emaildata1" with a help icon (?) to its right.
- Subject:** An empty text input field.
- From Email:** An empty text input field.
- From Display:** An empty text input field.
- Group Name:** An empty text input field.
- Locale:** An empty text input field.
- CC Email:** An empty text input field.
- BCC Email:** An empty text input field.
- Priority:** A dropdown menu with "Normal" selected and a downward arrow on the right.

For information about this control's properties, please see the [EmailData Control](#) topic.

You can add multiple EmailData controls to an email template. Use the [Condition Builder](#) to optionally show and hide these by going to the control in the Form properties page and setting a condition on each EmailData tag created. Each tag has a unique ID.

The screenshot shows the 'EFORM CONTROLS' interface for a 'Notification Email Template'. It features a table with the following columns: Control Name, Control Type, Properties, and Commands. The 'Email Data' entries are highlighted with a red box.

Control Name	Control Type	Properties	Commands
ApprovalTaskAssigned	Email Data		Edit
POAssigned	Email Data		Edit
POCompleted	Email Data		Edit
PRDisapproved	Email Data		Edit
ProcURL	HTML Control		Edit
ProcURL2	HTML Control		Edit
SectionApprovalNotification	Section		Edit
SectionPOCompleted	Section		Edit
SectionPRDisapproved	Section		Edit
SectionPRGenerated	Section		Edit

Once you have multiple Email Data objects on the email template, you'll need to set up conditional visibility of the objects. On the Forms Controls tab of the email template file, you can set conditions of what email data or email sections should be used like this:

The screenshot shows the 'Form Field Properties: TaskAssigned' dialog box. The 'Field is Visible When ...' dropdown is set to '(Email Type = Task Assigned)' and the 'Otherwise Hidden' checkbox is checked.

This Email Data object will be visible when the email type is "Task Assigned". There is more about these Email types in the next section, "Email Template Conditional Processing", below. One more thing to note: when multiple Email Data objects' conditional statements will evaluate as "true", then PD will work through the separate Email Data objects in order, and information in properties input boxes will be replaced by non-blank properties boxes of the next Email Data object. If any of the "Cancel Email" boxes are checked in any displayed Email Data objects, the email won't be sent.

There is one email type that is sent when a user's task is completed due to a condition that completes the task without the user's involvement. For example, a task may assigned to a group of users, but be set to complete when the first user in the assigned group completes the task. In this case, the default behavior is for all users to receive the task completion email. Essentially, for any function that causes a user to be marked as "Not Required", all assigned users will receive email notifications that the task is complete. If

you don't want this behavior, add an **EmailData** tag or control to the email template with the "Cancel Email" option set to True. Be sure to place that Email Data control in a section that is visible when the email type is "not needed".

### ASPX Format

```
<bpX:EmailData ID="StepName" runat="server" Subject="Subject Line"
MailPriority="High"
    FromEmail="{CREATE_USER, format=email}" FromDisplay="{CREATE_USER}" SendAsAttachment="True"
    GroupName="Group" CancelEmail="True">
</bpX:EmailData>
```

### Email Template Conditional Processing #

The email templates can optionally contain Section tags that can control what portions of the email should be displayed under different conditions (e.g. value of a form field). You can optionally display any control that has an ID which will display in the list of controls on the Form properties page. Conditional processing allows you to display a type of email to use such as the reminder email type. You can conditionally display controls based on the type of email. The following email types are available in Process Director:

- Task Assigned
- Reminder Email
- Task Timeout
- Task Canceled
- Step Started
- Step Stopped

Refer to the chapter on Form Definitions for more information on conditional processing. To test email templates that contain Form tags, upload the email template to Process Director as a Form but marked as an Email Template to have the built-in parser validate the use of your Form tags. This will only allow the email template to be validated; it won't be a valid Form.

### Task Links in an Email #

The email templates can optionally contain links that allow the user to complete their task. This system tag will add the task URL to the link tag in the HTML email allowing the user to click on it to open in the browser and complete their task.

In the Online Form Designer:

```
{EMAIL_URL}
```

In .ascx:

```
<bpX:HTML ID="ControlName" runat="server" HTMLString="<a href='{EMAIL_URL}'>Click
here to view the process</a>">
</bpX:HTML>
```

This system tag will return the URL that can be placed within an <A> tag on the HTML email.

### Complete Links in an Email #

Email templates can contain links that allow the user to complete their task without having to view a Form.

```
{EMAIL_RESULT_LINKS, comments=1, confirm=1, separator="&lt;br>", icon=1}
```

This system tag will return the possible branches to take. It creates an HTML link with the URLs for all of the possible results in a Process Timeline Activity assigned to the user.

ARGUMENT	TYPE	DEFINITION
comments	true/false(default value)	The user is prompted to comment if true
Confirm	true(default value)/false	The user is required to confirm the task has been completed
Separator	HTML tags	characters to separate the list of links
Icon	true(default)/false	Optional icons/images from the Timeline Activity results appear next to each of the links.  NOTE: Process Timeline results don't have image icons.

### Example

Possible Branches to Take:

```
{EMAIL_RESULT_LINKS, comments=1, confirm=1, separator="&lt;br>", icon=1}
```

This system tag will return a comma separated list of all of the possible results from a result in a Process Timeline Activity assigned to the user. This tag is useful for reply options the recipient must use. The next process may be scanning to match text in the sent Email.

### Example

Please enter one of these keywords in responding to this Email:

```
{Email_                               Result_                               List}
{EmailCompleteLink, text=Approve, comments=1, confirm=1, action=Approve}
```

This System Tag is a link to a web page allowing their user to complete their task.

Each **EmailCompleteLink** System Tag with its arguments corresponds to one result in a Process Timeline Activity. Remember to name this Email Template as the "Default Email template" in the Process Timeline definition.

ARGUMENT	TYPE	DEFINITION
Text	string	text to display on the link (e.g. Approve, Resubmit, Reject)
Comments	true/false(default value)	The user is prompted to comment if true
Confirm	true(default value)/false	The user is required to confirm the task has been completed

ARGUMENT	TYPE	DEFINITION
Action	string	The Process Timeline Activity result (e.g. Approve, Resubmit, Reject)

### Example


```
{EmailCompleteLink, text=APPROVE, comments=true, confirm=true, action=APPROVED}
{EmailCompleteLink, text=RESUBMIT, comments=true, confirm=true, action=RESUBMIT}
{EMAIL_COMPLETE_URL, action=Approve, comments=1, confirm=1}
```

This system tag will return the URL that can be placed within an <A> tag on the HTML email.

ARGUMENT	TYPE	DEFINITION
Text	string	text to display on the link (e.g. Approve, Resubmit, Reject)
Comments	true/false(default value)	The user is prompted to comment if true
Confirm	true(default value)/false	The user is required to confirm the task has been completed
Action	string	The Process Timeline Activity result (e.g. Approve, Resubmit, Reject)

### URL Access to the Process Instance #

#### Workflow (wd.aspx)



The Workflow object is the legacy process model used in early versions of Process Director. BP Logix recommends the use of the [Process Timeline](#) object, and not the Workflow object. The Workflow object remains in the product for backwards compatibility, but doesn't receive any new functionality updates, other than required bug fixes. No new features have been added to this object since Process Director v4.5. All new process-based functionality is solely added to the Process Timeline.

#### Description

This URL can be used for in Workflow emails. It is used to construct a hotlink back to the originating Workflow.

#### Parameters

PARAMETER NAME	DESCRIPTION
forminstid	The ID of the form instance to display.

## Examples

This example will create a hotlink in an ASCX Email template which points back to the Workflow's Form

```
<bpx:HTML runat="server" HTMLString="<a href=' {INTERFACE_ URL}wd.aspx?forminstid=
{FORM_
INSTANCE_
ID}'>
Click Here to View the Form Instance</a>">
</bpx:HTML>
```

## Process Timeline (pd.aspx)

### Description

This URL can be used for in Process Timeline emails. It is used to construct a hotlink back to the originating Process Timeline.

### Parameters

PARAMETER NAME	DESCRIPTION
forminstid	The ID of the form instance to display.

### Examples

This example will create a hotlink in an ASCX Email template which points back to the Process Timeline Form

```
<bpx:HTML runat="server" HTMLString="<a href=' {INTERFACE_ URL}pd.aspx?forminstid=
{FORM_
INSTANCE_
ID}'>
Click Here to View the Form Instance</a>">
</bpx:HTML>
```

## Using Multiple Email Data Controls in a Template

When you create an email notification template, you aren't limited to creating one template for each notification. Instead, you can use that same template for many different notifications by using multiple **Email Data** controls in each template. It's quite possible to use one template for all notifications associated with a process.

The keys to using a single template for multiple notification are:

1. Divide the email template into different sections, each of which has its own Email Data control, and
2. Determine the conditions under which only one **Email Data** template is visible for each notification.

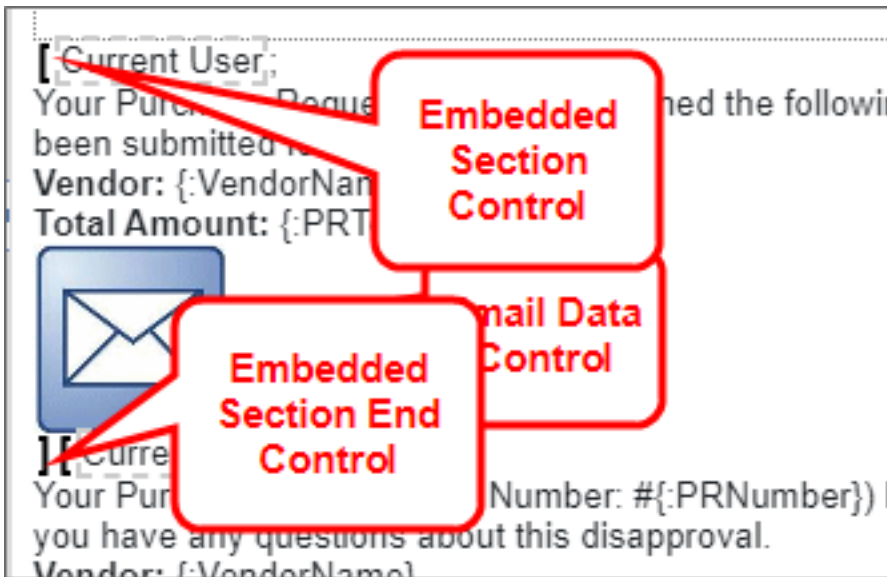
Process Director includes a type of section control called the **Embedded Section**. Unlike the **Section** control, the **Embedded Section** is hidden from the user. The trait makes it useful for creating multiple sections on a Form that may be hidden or displayed as necessary. Section controls are all container or parent controls that house individual Form controls or other content. If a section control is hidden, all of the child controls in the section are hidden as well.

By placing an **Email Data** control and other relevant content in each section, you can hide all of the sections except the one containing the necessary content for each notification.

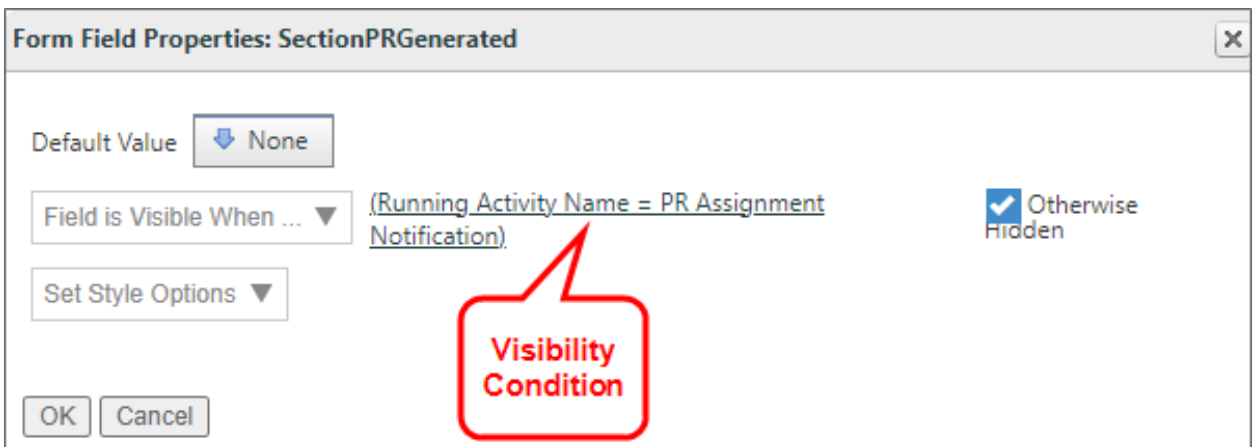
### Determining Section Visibility

The most important part of making a multiple-notification email template work properly is ensuring that you can create a set of conditions that results in only one valid **Email Data** control being active for every

notification, with each notification being presented in it's own section of the template.



Obviously, different processes may require different conditions to be applied to the **Visibility** property of each section. One place to start, however, is setting the section to become visible based on the Running Activity Name.



In the case of the example above, the **Embedded Section** control has the **Visibility** property set in the **Form Controls** tab of the Form definition, so that the section is visible when the Running Activity name is "PR Assignment Notification", and is otherwise hidden. With this setting, the section, and the **EmailData** control it contains will only be visible when the notification is sent from the "PR Assignment Notification" task.

Other conditions might be applied for various circumstances, such as when the email is sent to a particular type of participant (like a manager), or when a Due Date has been reached. The important thing to remember is to ensure that only one section, with one **EmailData** control, is visible at any given time the notification is sent.

You always have the choice, of course, to use a single email template for a notification, rather than using a single template with multiple **EmailData** controls. So, the question is, when should you use a particular method?

First, if it isn't possible to derive a set of conditions that ensure only one **EmailData** control is visible at a given time, a separate notification template will be necessary. Second, you may also have a special template for notifying managers of an event, and you may wish to have a completely separate template for making this managerial notification. In those cases, a single template isn't desirable.

Additionally if you have many different task notifications, as well as additional notifications for non-task users, you may wish to have a template for task notifications, and a separate template for non-task notifications.

## Task Completion Control via Email

There are several ways to control task completion via email, each of which prompts different responses in Process Director.

### ***Email Completion Links #***

There are three types of email links that can be inserted into the email template for completion of a task, each of which direct users to a completion form in Process Director, and not to the original Form:

- Email Result Links
- Email Complete Link
- Email Complete URL

All of the completion links do essentially the same thing, that is, they direct the user to a task completion page, bypassing the original Form. They each have a slightly different implementation on the email template, however.

The **Email Result Links** provide the user with a list of all of the possible results for a task, while the **Email Completion Link** provides a result link for a single possible result. So, if you want the user to see all of the possible task results, you'd use the **Email Result Links**, while if you only want to show a result for one or two possible results, you'd configure an **Email Completion Link** for each of the results you desire to show the user.

As an example, let's say that you have three possible results for a Process Timeline Activity: Approve, Reject, and Resubmit. For some users, you might want to show the **Email Result Links** to display all of the possible results:

```
{EMAIL_RESULT_LINKS, comments=1, confirm=1, separator=" | ", icon=1}
```

In this example, the **Email Result Links** control tag is configured to require comments (`comments=1`), require a confirmation dialog for the result (`confirm=1`), insert a spaced bar character between each result (`separator=" | "`), and display the icons for each result that was configured in the Process Timeline Activity (`icon=1`).

For another user, you might only wish to show the Approve or Resubmit results but not the Reject result:

```
{EmailCompleteLink, text=APPROVE, comments=false, confirm=true, action=APPROVED}  
{EmailCompleteLink, text=RESUBMIT, comments=true, confirm=true, action=RESUBMIT}
```

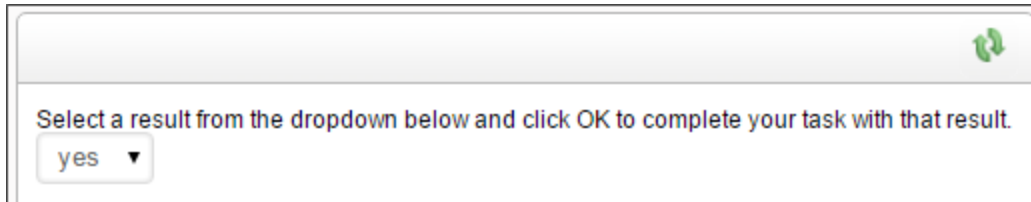


In this example, the **Email Complete Links** are configured to specify the text to display for the result (text=APPROVE), require comments for the Resubmit result (comments=true), require a confirmation dialog for the results (confirm=true), and specifies the task result, or action, for each result (action=RESUBMIT).

The **Email Complete URL** control tag, at its simplest, provides a link to the task completion page.

```
{EMAIL_COMPLETE_URL}
```

In this example, the user is provided with a link to the completion page, but, since it has no indication of what the result should be, the completion page will present the user with a dropdown control that displays all of the possible results.



It is possible, however, to fully configure the **Email Complete URL** control tag, so that it acts just like an Email Complete Link control:

```
{EMAIL_COMPLETE_URL, action=Approve, comments=1, confirm=1}
```

Using any of the three completion links work best if the process participant doesn't need to edit any information on the Form, and the only action is to provide a result. If you do wish the user to go to the Form to edit or add information, then you can't use the result links.

### ***Offline Completion #***

Allowing a task to be completed from email is called offline completion. Offline completion enables processes to parse an email message, and, based on the contents of the email, complete a task. Offline email completion requires several specific configuration elements:

1. A properly configured Timeline Activity.
2. A properly configured Email notification template.
3. An email import process.
4. A properly configured import folder into which emails can be imported to prompt the email import process.
5. A method of importing the email messages into the specified import folder.

We'll discuss each of these methods in more detail below.

## **1. Configuring the Process Timeline Activity**

For offline completion of an Activity to work, the Process Timeline Activity must be properly configured. First, in the **Advanced Options** tab of the Process Timeline Activity, you must select the **Allow task to be completed from email** property.

Activity Participants Results Start When

- Is activity required to complete parent
- Set Activity result to a list of all children or user results
- Re-authenticate users
- Prompt for Signature Comments
- Allow task to be completed from email
- Enable email invitations
- Require at least one user to be assigned and complete this task
- Assign task to first user to accept
- Allow anonymous users to be assigned tasks by email address
- Switch the user in Case mode when opening this task

When this property is checked, Process Director will search an imported email message for the task for which the result must be set, and the text of the configured result. By default, Process Director will search for the text configured on the **Results** tab as the **Result Name** for all of the configured results. You can also add additional strings to search for to meet the result by using the **Optional list of strings for email complete** property to enter a text string or comma-separated list of strings that, if found, will also constitute a valid result.

Activity Participants **Results** Start When Completed When Needed When Due Date Notifications Advance

List the possible results of this activity.

Add Result

Approve

Result Name: Approve

Description:

Conditions Options

Button Order on Form: 1

Button Icon on Form:

Button Background Color: Green

Button Text Color: White

Optional list of strings for email complete: Approved, Approves, Yes, OK

Signature Comments Required  Allow this result to be completed from the users Task List  Skip Validation on Form when Selected

In the example shown above, in addition to the "Approve" result, the result is also configured to accept "Approved", "Approves", "Yes", or "OK" as a valid "Approve" result for the Activity. Any email that contains these optional text strings instead of "Approve" will still complete properly, and the result "Approve" will be applied to the activity.

You must repeat this configuration for all of the results of the Timeline activity. For example, the "Deny" result in this example is configured as:

## 2. Configuring the Email Notification Template

In order for Process Director to correctly associate the email message with the task that needs to be completed, the email notification must use the [Email Task ID System Variable](#) ({EMAIL\_TASK\_ID}) somewhere in the body of the email notification.

In this example, a **System Variable** control has been placed on the notification message template to invoke the **Email Task ID** system variable at run time, and place the GUID identifier for the task as text in the email message.

The value placed in the email by this system variable should *never* be edited, or the parsing and completion process will fail, so it's important that end users, when replying to the email notification, do *not* edit this value.

Finally, the **Email Data** control that you place on the email notification template must use, as the **Reply To** address, the email inbox that will be used to store email responses until they're imported into Process Director. When the user replies to the notification email, the message will be routed to this email inbox, where it will wait until imported.

### 3. Configuring the Email Import Process

You'll need to create a Process Timeline that has a single Custom Task activity that invokes the **Email Action** Custom Task. In this example, the Process Timeline has been named **Email Action**, which you'll need later when configuring the import folder.

Index	Activity	Type	Actions	Timeline (Days)
1	Import Email	Custom Task	[Icons]	1 2 3 4 5 6 7 8 9 10 11 12

This is the Process Timeline that will import the email, parse it for the Task ID and completion text string, then apply the parsed result to the specified user task/activity.

The **Email Action** Custom Task must be configured with the following settings.

First, the **Set Container Form** property for the Custom Task must be configured with the main process Form for the Timeline activity you wish to complete, which, in this example, is named **Process Form**.

Name: Import Email

Activity: Custom Task

Set Container Form (optional): Process Form

E-mail Action Configured On 6/14/2023

Buttons: Close, Cancel

Once set, you can then configure the Custom Task, primarily its **Data From Message** section, which specifies how to parse the email response.

**Data from Message**

Validate that email was sent from an email address belonging to a user to whom the task was assigned

Destination Location	Line	Value
Results	Line: ▼	1
Comments	Line: ▼	2

Ignore Text part of multipart message

Append Data to an Array     Append Data to Form Field

In this example, the Custom Task is configured to look for the result text in the first line of the email message, and any signature comments in the second line of the message, e.g.:

Approve —————

This is the signature comment for this task

---

**From:** Dale Franks <dale.franks@bplogix.com>  
**Sent:** Thursday, June 15, 2023 9:07 AM  
**To:** Dale Franks <Dale.Franks@bplogix.com>  
**Subject:** Email completion

To approve, hit reply. Type the approval text on the first line and any comments you'd like to add on the second line of the reply email.

=====DANGER!!!!!!!!=====

**You CANNOT edit this. EVER. Don't touch it: [TLID:994730f6-1741-4814-b55e-df67d71f6e33]**

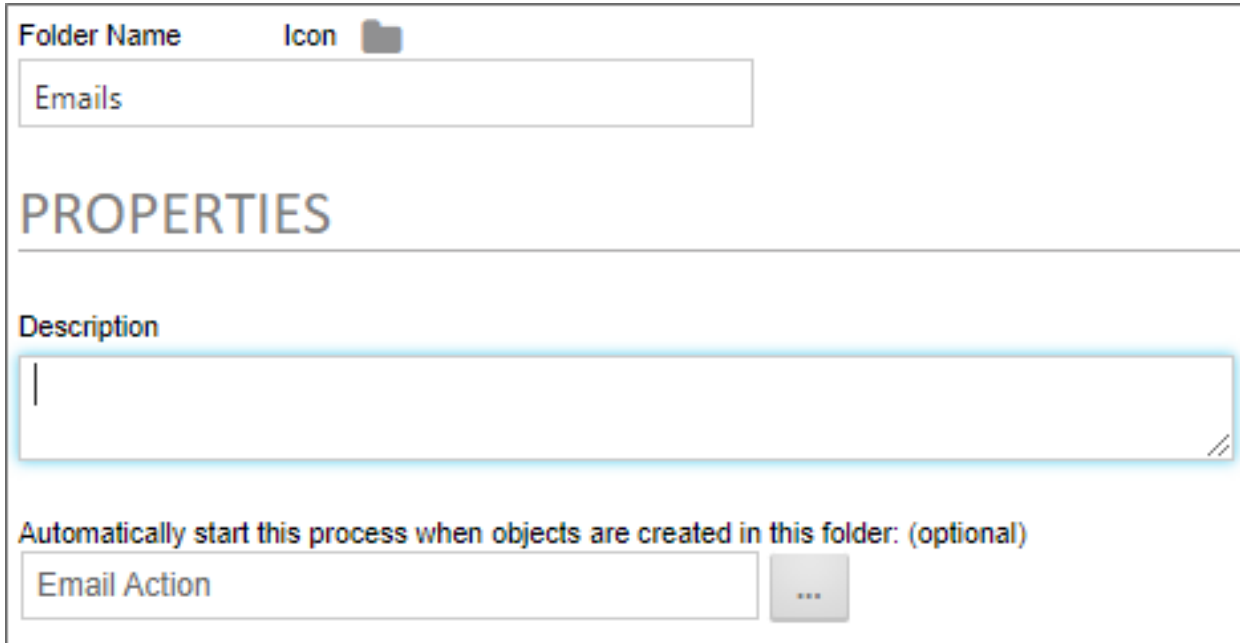
This is probably the easiest method to use for parsing the imported email message.

While this basic configuration is sufficient, you may wish to also configure whether to include the parsed email as a process attachment, or configure simple notifications for success or failure of the email import, to let the user know the result of the import. Please see the [Email Action Custom task documentation](#) for more details on configuring this Custom Task.

When properly configured, the Custom Task will parse the email messages in a specified email inbox for the appropriate result, and send an email to the user for unparseable or ambiguous emails, to prompt the user to resend the completion email.

## 4. Configuring the Import Folder

Email messages must be imported from the specified email inbox into a Process Director [Content List](#) folder for processing. This folder, named [Emails](#) in the example below, must be configured with the [Automatically start this process when objects are created in this folder](#) property set to invoke the import process you configured previously.



The screenshot shows a configuration window for a folder. At the top, there is a table with two columns: 'Folder Name' and 'Icon'. The 'Folder Name' column contains a text input field with the value 'Emails'. Below this is a section titled 'PROPERTIES'. Under 'PROPERTIES', there is a 'Description' label followed by a large, empty text area. At the bottom, there is a label 'Automatically start this process when objects are created in this folder: (optional)' followed by a dropdown menu showing 'Email Action' and a button with three dots.

In this example, the [Email Action](#) process Timeline that we configured in Step 3 will be invoked every time an email message is imported (or uploaded) to the [Emails](#) folder.

## 5. Importing the Email Messages

Offline completion generally uses the [bpImportEmail utility](#) to retrieve emails from the email inbox on a scheduled basis, and import them into the specified folder for automatic parsing. You should know, however, that you can also manually upload email messages into the folder as well, if needed. Irrespective of how the email message is placed in the folder, the [Email Action](#) Process Timeline will run for each email message when it's placed there.

Keep in mind that scheduling the email import imposes some delay on the processing of the email message, since the email message will be stored in the [Reply To](#) inbox until the next scheduled run of the bpE-mailImport utility. For example, if you run bpEmailImport every 8 hours, there will be a delay of up to 8 hours before the email message is imported into the folder and parsed.

## Goals




Goals are used to set a global state in the Process Director system, and to perform actions when a global state change occurs. A Goal returns a value, or set of values, based on conditions that the Goal regularly evaluates, on a schedule that is specified in the Goal definition. For instance, you can set a Goal to execute daily, hourly, etc., and set a global system state when the execution occurs.

Goals incorporate two completely distinct components: execution and examination. **Execution** occurs when the Goal evaluates the conditions and sets the state or value that results from the evaluation. **Examination** occurs when a another Process Director object references the Goal to retrieve the value that was set during the execution. Referencing a Goal (the examination) will always produce the result from the last time that Goal was executed. The Goal will return a null string if it has never evaluated, otherwise the examination will always produce the value resulting from the most recent execution.

Scheduled execution makes the Goal uniquely suited for evaluating conditions that are processor-intensive. Unlike a system variable that is executed when referenced (e.g., when called from a Form or Process Timeline), Goals are only executed on their configured schedule. So, you can schedule a processor-intensive Goal execution at a time when usage is low. When another object references the Goal, the Goal does *not* re-execute. It merely sends the value that was set during the last scheduled execution. Many different processes, therefore, may use the same Goal without placing a strain on the server, because the Goal will only execute when scheduled.

Goals also can initiate actions, either explicitly, or implicitly. If a Goal changes state after a scheduled execution, the Goal can initiate a process or a run a Knowledge View automatically when the state changes, which we refer to as an **explicit action**. Alternatively, a running Timeline instance may use a Goal's value to determine which Activities to run, or which users to assign to an Activity, which we refer to as an **implicit action**.

Let's examine a scenario in which a Goal might be useful. Imagine a call center that takes customer support requests. This customer center normally takes 50 or fewer callers per hour. If the number of calls per hour exceed 50 calls, you might want to assign additional users to take support calls. If there are more than 75 calls per hour, you might wish to notify managers of heavy call volume. In this scenario, you could create a Goal that checks the number of calls per hour every fifteen minutes. When then Goal evaluates, it sets a status of "Green" when there are 50 calls or less per hour, "Yellow" when there are 50-75 calls per hour, and "Red" when there are more than 75 calls per hour. In the Goal, you could also automatically start a new process that sends notifications to managers if the system status changes to "Red". In your process, you could add additional users to whom to assign tasks when the status is "Yellow" or "Red". In this example, using a Goal both alters how your process is performed and starts a new process when conditions require it.

Conditions and Results (3 Results)		
When Condition met...	Set this Result	... and perform this Action
 <code>{(VAR.Calls) &lt;= 50}</code>	Green	Set Result Only
 <code>{(VAR.Calls) &gt;= 51 AND (VAR.Calls) &lt;= 74}</code>	Yellow	Set Result Only
 <code>{(VAR.Calls) &gt;= 75}</code>	Red	Run Process Documentation

The ability to start a process based on a Goal, along with the ability to schedule Goal evaluation, enables Process Director to schedule processes without having to use the Windows Scheduler.

In addition, results for a Goal can be evaluated from a [Business Value](#), giving you the ability to schedule processes based on information from your outside CRM, ERP, or other systems. For example, you can use a Business Value to return any unprocessed Invoices or POs from your ERP system. The Goal can evaluate the Business Value at a desired interval, and, if there are unprocessed items, automatically start the PO or

Invoice process to process the items. Moreover, you can return any required data to update your ERP system during the processing or after the processing is complete. In effect, simply entering a new invoice and PO into the ERP system would automatically start the PO or Invoice process—and ultimately update your external ERP system—when the Goal is evaluated. In this example, you'd no longer need to manually start the process by filling out and submitting a Form.

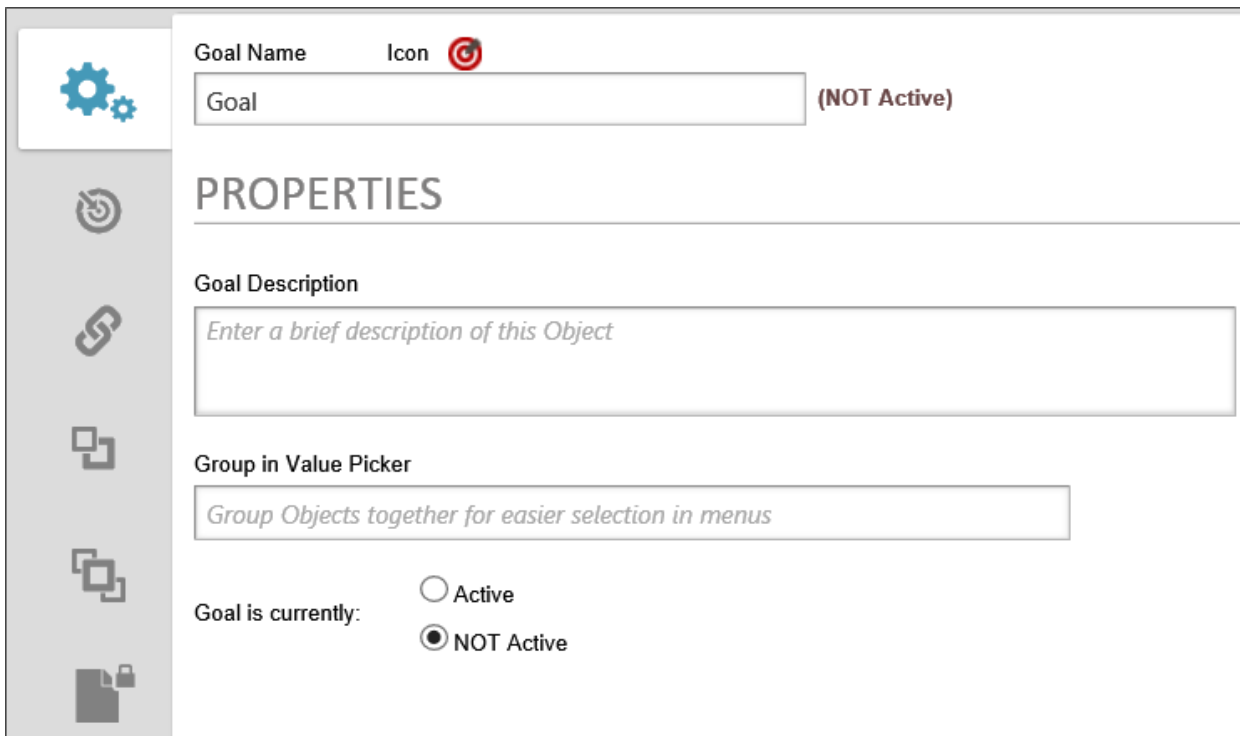
Combined with the access to external data available through the Business Value feature, Process Director is a powerful process automation system, enabling automatic process initialization in response to changes to external systems.

To see how to configure a Goal object, please continue to the [Configuring Goals topic](#).

## Configuring Goals

Goals, like all other Process Director objects, are configured via a tabbed interface. The interface tabs for the Goal definition, and the properties for each tab, are specified below.

### Properties Tab #



The screenshot displays the configuration interface for a Goal object in the Properties tab. On the left is a vertical sidebar with icons for settings, goal, link, group, and document. The main area contains the following fields:

- Goal Name:** A text input field containing "Goal" and a status indicator "(NOT Active)".
- Icon:** A red target icon.
- Goal Description:** A large text area with the placeholder text "Enter a brief description of this Object".
- Group in Value Picker:** A text input field with the placeholder text "Group Objects together for easier selection in menus".
- Goal is currently:** Radio buttons for "Active" (unselected) and "NOT Active" (selected).

The following properties can be configured in the **Properties** tab of a Goal definition.

#### Name

The name of the Goal, which will be displayed in the Process Director Interface.

#### Icon

Click the Icon to open the [Icon Chooser](#) and pick the desired Icon for the Goal.

#### Goal Description




A brief description of the Goal's purpose.

### Group in Value Picker

The Group name for the Goal, which will appear in all dropdown menus for selecting system variable values or in the [Condition Builder](#).

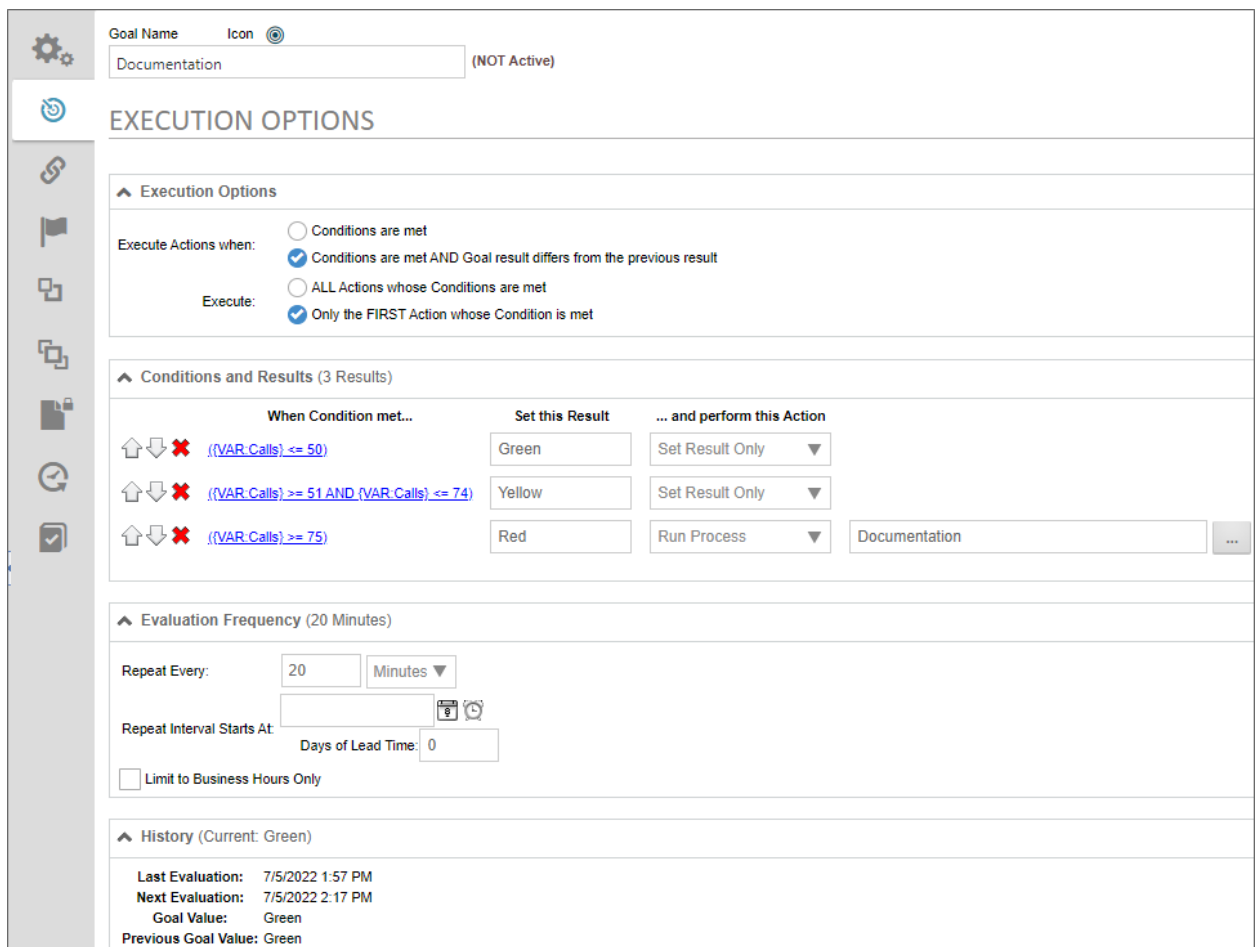
### Goal is Active/NOT Active

Radio button group to enable or disable a Goal's scheduled evaluation. Disabled Goals won't be evaluated on the configured schedule until they are enabled.

 The Active/Inactive designation only refers to the frequency of execution. Any object that references the Goal will return the Goal's current value, irrespective of whether the Goal is active or not.

## Execution Options Tab #

Goals are configured by configuring the **Execution Options** tab of the Goal definition.



The screenshot shows the 'EXECUTION OPTIONS' configuration screen for a goal named 'Documentation'. The goal is currently '(NOT Active)'. The interface includes a sidebar with various icons and a main configuration area with the following sections:

- EXECUTION OPTIONS**
  - Execution Options**
    - Execute Actions when:
      - Conditions are met
      - Conditions are met AND Goal result differs from the previous result
    - Execute:
      - ALL Actions whose Conditions are met
      - Only the FIRST Action whose Condition is met
  - Conditions and Results (3 Results)**

When Condition met...	Set this Result	... and perform this Action
<code>{(VAR.Calls) &lt;= 50}</code>	Green	Set Result Only
<code>{(VAR.Calls) &gt;= 51 AND (VAR.Calls) &lt;= 74}</code>	Yellow	Set Result Only
<code>{(VAR.Calls) &gt;= 75}</code>	Red	Run Process <span style="float: right;">Documentation</span>
  - Evaluation Frequency (20 Minutes)**
    - Repeat Every: 20 Minutes
    - Repeat Interval Starts At: [Calendar icon]
    - Days of Lead Time: 0
    - Limit to Business Hours Only
  - History (Current: Green)**
    - Last Evaluation: 7/5/2022 1:57 PM
    - Next Evaluation: 7/5/2022 2:17 PM
    - Goal Value: Green
    - Previous Goal Value: Green

Goals are configured to automatically evaluate on a schedule configured by the Goal creator. Actual Goal evaluation frequency, however, depends on several factors, including the frequency with which Activity checks are being run within Process Director in general. When saving a Goal definition, Process Director will try to determine if the Goal is being scheduled for evaluation more frequently than Activity checks, and will display a warning if that is the case. However, note that Goal evaluation frequency is always approximate.

The **Execution Options** tab is divided into four sections: **Execution Options**, **Condition and Results**, **Evaluation Frequency**, and **History**.

The **Execution Options** section defines how the Goal conditions are evaluated, and how actions are performed on the Goal result.

### Execute Actions when

- **Conditions are met:** The Goal's actions will run every time the Goal is evaluated and the conditions are met.
- **conditions are met AND Goal result differs from the previous result:** The Goal's actions will only run when the result of the Goal evaluation *changes*, and the conditions are met.

### Execute


- **ALL Actions whose Conditions are met:** The Goal will run all of the possible actions in the Actions list that meet the conditions.
- **Only the FIRST Action whose Condition is met:** The Goal will only run the first action in the Actions list that meets the condition. If you elect to take only the first matching condition, then the conditions will be evaluated in the order you specify in this section, i.e., from top to bottom. If no conditions for the Goal evaluate to true, the Goal will return a blank value.

The **Condition and Results** section defines the conditions to be evaluated, and the results to be returned.

### Add Action

To add an action to the action list, click the **Add Action** button to display a new, blank Action.

### Order Buttons

The order in which evaluations occur can be controlled by using the Arrow Icons at the end of each condition row. A Goal result can be deleted by clicking the  icon.



### When Condition Met...

A condition you specify using the **Condition Builder** to determine whether to return the Evaluation Result.

### Set this Result

The value that will be returned if the specified condition evaluates as true. In most use cases, you'll likely specify a text value, but you may enter a system variable using the curly brackets syntax, e.g., {CURR\_DATE}.

### ...and perform this Action

The action to take if the evaluation returns true. You have the following options:

- **Run Process:** Start a process if the evaluation returns true. If this option is selected, a Picker control will appear in the Action column that enables you to choose the process to begin.
- **Run Knowledge View:** Run a Knowledge View if the evaluation returns true. If this option is selected, a **Content Picker** control will appear in the **Action** column that enables you to choose the Knowledge View to run. In addition, a **User Picker** will appear that enables you to choose the user context under which the Knowledge View will run. The selected user *must* have the appropriate permissions to run the specified Knowledge View, or the Knowledge View won't run.
- **Set Result Only:** Only return the Evaluation Result without taking any other action. If the "Take ALL Actions that match Condition" option is specified, and more than one condition evaluates as true, the Goal will return a comma-separated list of all matching results.

### Action

This column will display the **Object Picker** control for a Process or Knowledge View, or will be blank, depending on the selection you make in the **...and perform this Action** column.

The **Evaluation Options** tab enables you to define the Goal's evaluation schedule.

^ Evaluation Frequency (20 Minutes)

Repeat Every: 20 Minutes ▼

Repeat Interval Starts At: [Calendar Icon] [Clock Icon]

Days of Lead Time: 0

Limit to Business Hours Only

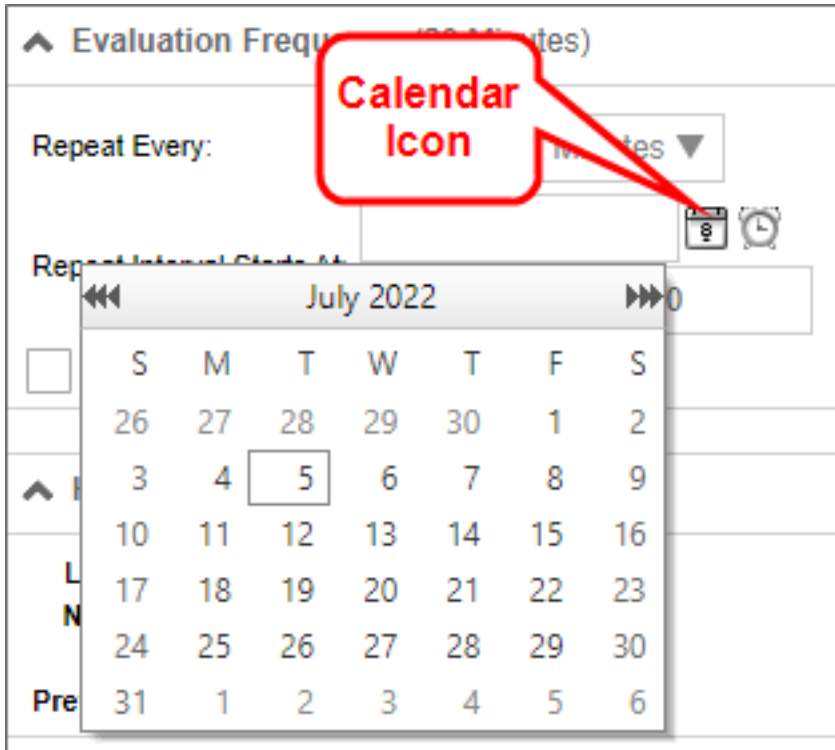
### Repeat Every

The frequency, in time periods from minutes to years, with which the evaluation is performed. Setting this value requires entering the number of time periods in the text box, then picking the desired time period from the adjacent dropdown. The options for the time units dropdown are:

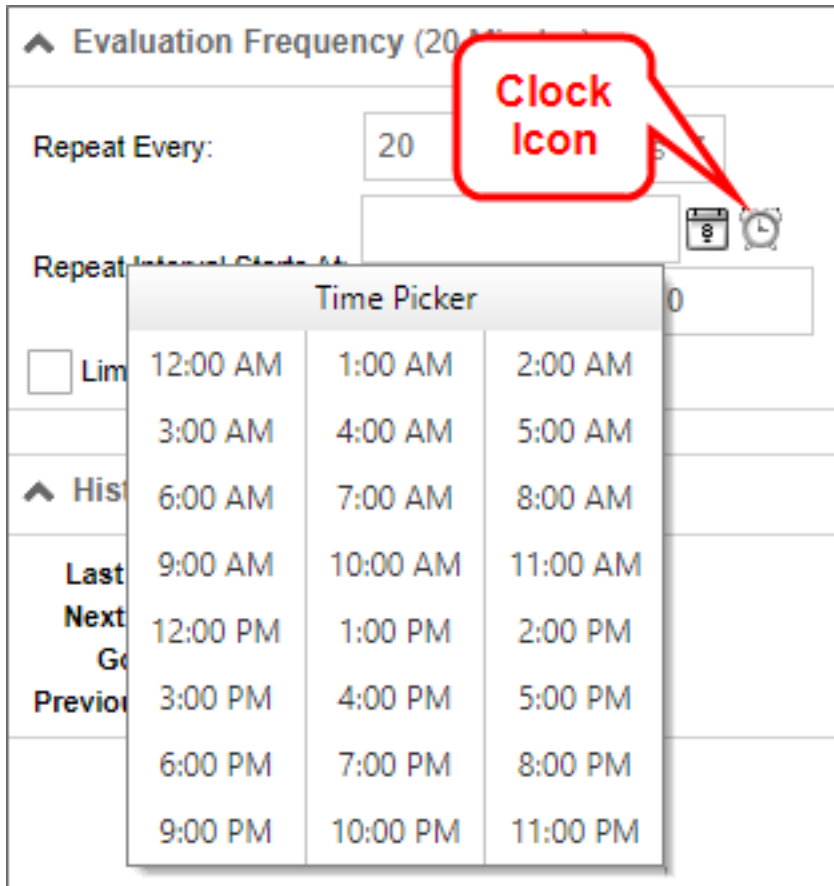
- Minutes
- Hours
- Days
- Weeks
- Months
- Years

### Repeat Interval Starts At

The date and time to begin scheduling evaluations. To select a date, click the **Calendar** icon located to the left of the text box control to open a calendar you can use to select the date.



Similarly, to select a time, click the **Clock** icon located to the left of the text box control to open a time selector you can use to select the time.



### Days of Lead Time

Sometimes, you want to begin a process prior to the due date. For example, a month-end report may need to be submitted on the first day of each month, covering the activities of the prior month. If we assume this report takes a couple of days to compile and generate, we might want to have a lead time of 2 days. In this case, the Goal will be evaluated two days early, so you have adequate lead time to generate the report.

### Limit to Business Hours Only

Check the box to evaluate the Goal only during Business Hours.

### Evaluate Now

Click the **Evaluate Now** button to evaluate the Goal immediately.

The **History** Section displays the evaluation run times, current result, and last result. It has no configuration settings.

^ History (Current: Green)

**Last Evaluation:** 7/5/2022 1:57 PM  
**Next Evaluation:** 7/5/2022 2:17 PM  
**Goal Value:** Green  
**Previous Goal Value:** Green

## Goal Evaluation History #

While technically not a configuration tab, since there's nothing on it to configure, the [Goal Execution History](#) tab is still important.

Goal Name: Books Due (NOT Active) OK

### GOAL EVALUATION HISTORY

Event Time	Object Name	Messages	Message Type
12/7/2023 11:22:38 AM	Books Due	The GOAL's new value (Nothing Due), the goals previous result (Books Due)	Info
12/7/2023 11:21:59 AM			
12/7/2023 11:21:10 AM			
12/7/2023 11:19:08 AM			
12/7/2023 11:13:36 AM			
12/6/2023 4:17:04 PM			
12/6/2023 4:16:59 PM			
12/6/2023 4:16:20 PM			
12/6/2023 4:15:48 PM			
12/6/2023 1:56:49 PM			
12/6/2023 1:56:44 PM			
12/6/2023 1:55:30 PM			
12/6/2023 12:03:56 PM			
12/6/2023 12:03:50 PM			
12/6/2023 12:01:09 PM			
12/6/2023 12:00:44 PM			
12/6/2023 11:58:16 AM			

This tab tracks all of the Goals evaluations to provide a record of when the Goal was evaluated, the results of each evaluation, and the results of the previous evaluation. This history enables you to track a Goal's evaluation frequency and results without having to refer to an external source like the Log files. It's a useful tool for ensuring the Goal is running at the dates/times expected. It can also be a valuable tracking tool for matching the condition results in other objects, such as Form or process Timelines, are accurately reflecting the goal's result at any given time.

## Knowledge Views

Process Director provides you with a convenient way to find and list information called Knowledge Views, or KViews, for short. The Knowledge View can display all types of information stored in Process Director, such as Forms, Process Timelines, Attachments, documents, etc. Knowledge Views only display information about Process Director objects, and can't return information from external data sources, such as

those generated by Business Values. To report on data from external sources, you'll need to use the [Advanced Reporting Component](#).

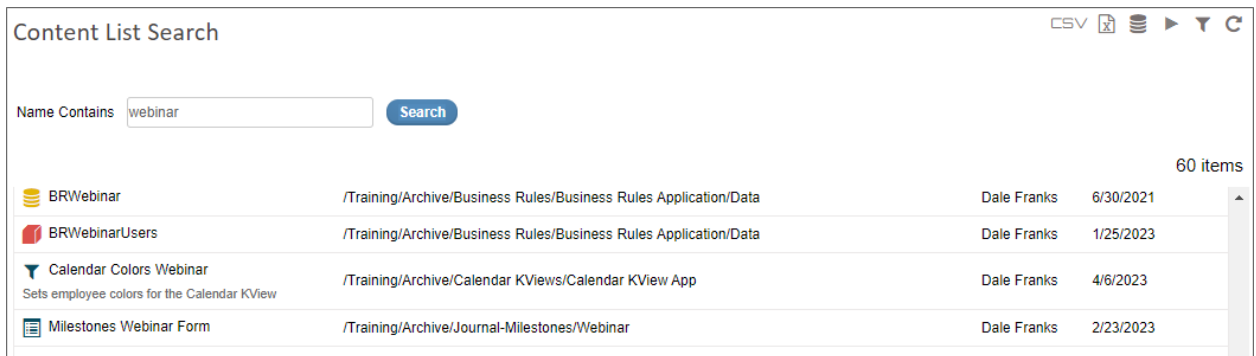
Knowledge View definitions are stored in Process Director and provide a “window” into the database showing related information. These are similar to predefined searches. Knowledge Views form the foundation for intelligent navigation, searching, reporting, retrieval, and ultimately delivery of information to the user community. A Knowledge View allows authenticated or anonymous users to navigate and retrieve related information based on the data classification.

Knowledge Views will display all matching documents, Forms, processes, etc. based on the criteria specified. Knowledge View definitions can also be created that will prompt users for input information to locate information faster.

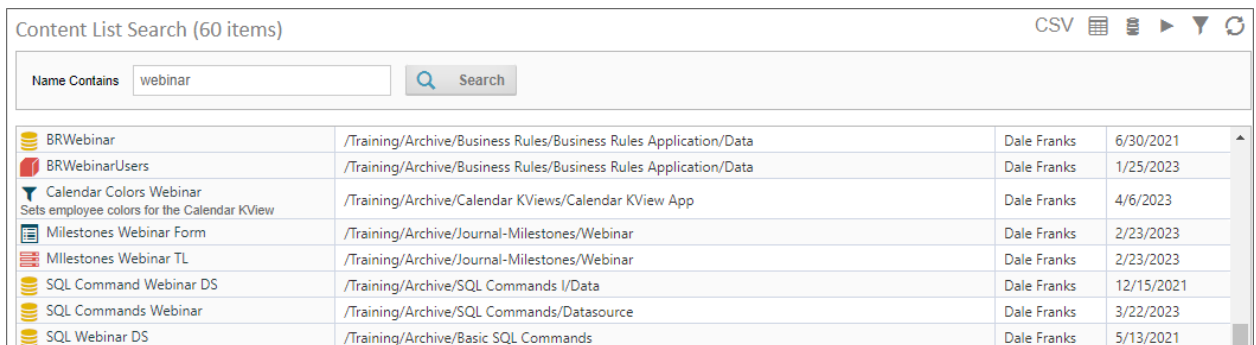
Knowledge Views also provide the foundation for the reporting component. The results of a search can:

- be displayed in the browser
- create and run an Excel report template
- be downloaded as a CSV file
- [run a process on each of the results](#)

For Process Director v6.0 and higher, running Knowledge Views are presented with a "flat" UI design by default:



This UI can be reverted to the UI displayed in previous versions of Process Director by unchecking the **Use Process Director 6.0 style** property on the **Configure** tab of the Knowledge View definition. This property is checked by default, but when unchecked, the UI displayed will be the same as it appeared in Process Director 5.X and below.



This change is purely visual, and has no effect on the operation of the Knowledge View.

## How Knowledge Views Work

To understand how Knowledge Views work, we must first understand what Process Director does with definition objects like Form definitions, Process Timeline definitions, etc. The definition object contains no actual data, but merely defines what data will be stored. For example, the Form definition describes the data fields that the user will fill out to enter data. Similarly, the Process Timeline definition describes when activities will start and end, or when a user task will become due, based on when the process begins.

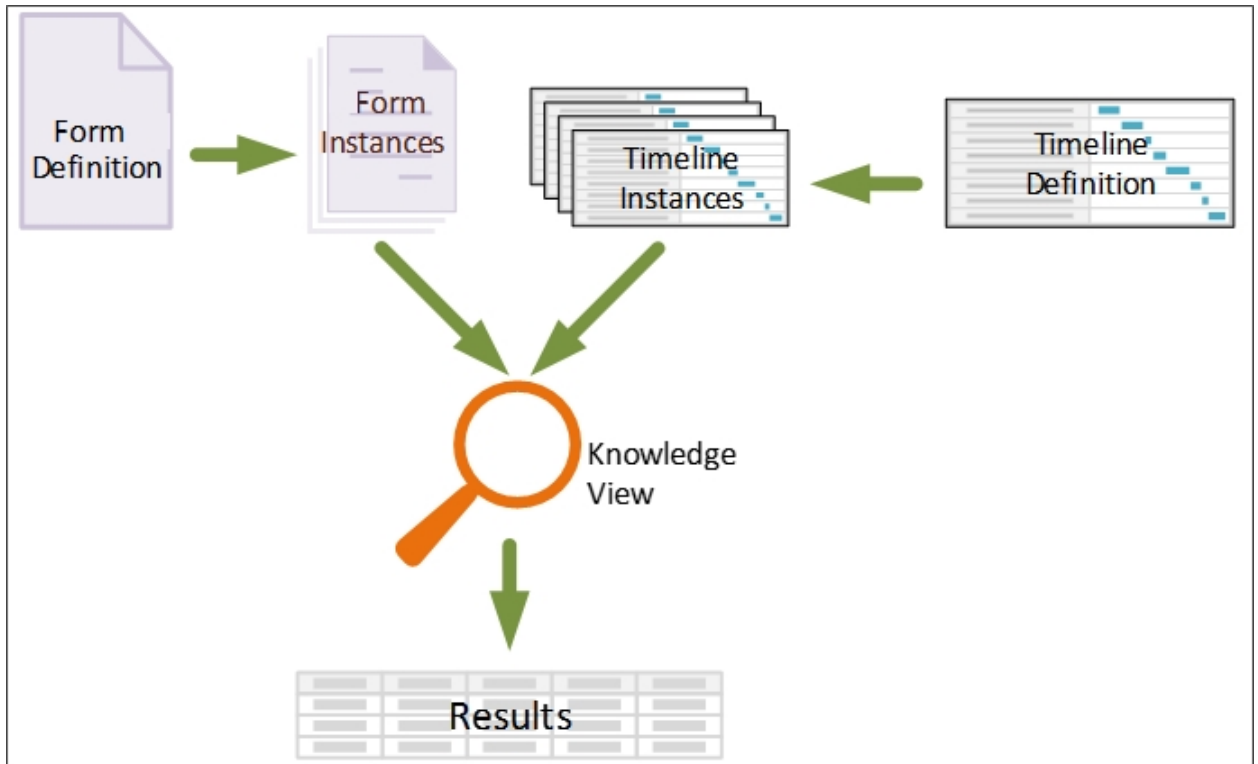
Regular, non-administrative users never interact with any of the definition objects directly. Instead, when a user wants to, say, fill out a Form, The Form definition makes a copy of itself, called an **instance**, that the user will fill with data in the fields specified in the Form definition. This instance object is what actually contains the data entered by the user. If ten different people fill out a Form, Process Director creates an instance for each person, so that there are ten instances of the Form, each of which contains the data filled out by one of the ten users.

By the same token, when a user submits a Form that initiates a Process Timeline, a new instance of the Process Timeline definition is created that stores the information about that specific run of the process. The whole purpose of the definition object is to describe the instance objects that are created to interact with the user.

Let's take a closer look at an example of the Process Timeline definition and a Process Timeline instance. When the definition is created, we might define a user Activity as being due three days after the Activity starts. When we create the definition, we obviously don't know when the Activity will start. All we know is that, whenever the Activity starts, we want the Activity to become due in three days. When a user submits the Form that starts the process, Process Director creates an instance of the Process Timeline that contains the actual Activity dates for that specific run of the process. So, in our example, if the Activity begins on January 1<sup>st</sup>, the Process Timeline instance that is created will set the due date for the Activity as January 4<sup>th</sup>.

When we search for information in Process Director, we really don't care much about the definition objects, because they don't contain the data. While there are other, more complicated scenarios for using Knowledge Views, we are usually interested in searching through all of the existing instances of a Form or process to find specific data. We might, for instance, want to search for all timeline activities that are due in the next week, or for all Forms that were submitted by a specific person.





Knowledge Views display information in a tabular format, much like a spreadsheet. So much so, in fact, that the Knowledge View results can be exported directly to CSV or Excel format.

When constructing a Knowledge View, you select the columns that you wish the Knowledge View to display. The columns can consist of any system variable, such as Form fields, Business Rule results, Process Timeline values, and much more.

Knowledge View Name:  Icon:

## COLUMNS

This section of the Knowledge View creation allows you to define which columns are displayed in the view.

Second line for each entry:

Default Column Sort Order:

Column Name	Column Data	When Displayed	
<input type="text" value="Name"/>	<input type="text" value="Name"/> ...	Always <input type="button" value="v"/>	<input type="button" value="up"/> <input type="button" value="down"/> <input type="button" value="X"/>
<input type="text" value="Size"/>	<input type="text" value="Size"/> ...	Always <input type="button" value="v"/>	<input type="button" value="up"/> <input type="button" value="down"/> <input type="button" value="X"/>
<input type="text" value="Author"/>	<input type="text" value="Create User"/> ...	Always <input type="button" value="v"/>	<input type="button" value="up"/> <input type="button" value="down"/> <input type="button" value="X"/>
<input type="text" value="Last Updated"/>	<input type="text" value="Update Date"/> ...	Always <input type="button" value="v"/>	<input type="button" value="up"/> <input type="button" value="down"/> <input type="button" value="X"/>
<input type="text" value="Version"/>	<input type="text" value="Version"/> ...	Always <input type="button" value="v"/>	<input type="button" value="up"/> <input type="button" value="down"/> <input type="button" value="X"/>

You can also choose to see different types of objects in the results. You can search for Forms, Process Timeline instances, Business Rules, documents, or any other type of Process Director object. You can select any number of object types to return, as well. The ability to return results containing any system variable from any type of Process Director object makes Knowledge Views very powerful.

Adding to that power is the filtering capability of Knowledge Views. Just as you can make a Knowledge View column out of any system variable, you can also filter every column to return only specific results.

While Knowledge Views are very powerful, you should also be aware that they can degrade server performance in some cases:

- KViews that have multiple filters that run against both Form and Process data can be complex queries that take a long time to run.
- When KView rows are returned, each column has to be filled with data. Sometimes this data is spread across many tables. For some Knowledge Views, it takes a substantial amount of server resources to fill the columns for each row returned.

These problems are exacerbated when the KView returns many rows. In Process Director v3.21 and above, Knowledge Views default to return a maximum 100 avoid performance issues. There is also an additional option that enables the user to get a message if the number of rows returned will be greater than the maximum. BP Logix also implements logic to improve the performance of the SQL generation of some Knowledge Views, to use information about the data returned to speed the query.

For users of Process Director v4.06 and higher, a performance improvement can be made to a Knowledge View when searching for form data on systems with millions of rows. There is an optional custom variable, [nLimitSearchToChars](#), that limits the number of characters to be searched in a form field.

In general, you should structure Knowledge Views to use as few columns, and return as few rows as possible to accomplish your purpose. In most cases, there is little to be gained in returning hundreds of rows in a "catch-all" Knowledge View. Such a Knowledge View simply has too much information to be useful in most cases, takes too long to generate, and takes too long for the user to parse through to find specific information.

Process Director is usually installed with Microsoft SQL Server as the back-end database. SQL Server is capable of returning hundreds of thousands or even millions of records in an instant; however, returning a large volume of rows resulting from a single query isn't what Knowledge Views are designed to do. Knowledge Views generate information for human consumption; humans don't ordinarily want to view that many rows. Instead, Knowledge Views are optimized to make it very easy to build or modify searches and tabular reports without having to code any SQL at all. Knowledge Views are, in that sense, general purpose wizards that automatically generate sets of potentially highly complex queries while hiding the nasty details of how those queries are actually constructed.

As a result, the performance of a Knowledge View isn't only related to the number of rows being returned, but even more by the number of queries required to construct the columns displayed in each row. So while the database might, for example, easily handle a single join resulting in a lot of rows, what the KView is actually constructing is one query to generate the underlying rows, and then another X queries per row to fill in the additional information configured by the user for each column in that Knowledge View. The total number of queries puts a larger load on the database than the single complex query that obtains the initial list of matching objects/rows.

And so, Knowledge Views represent a purposeful trade-off between design-and-build costs and execution time. In the vast majority of cases this is an optimal approach, making it very easy to generate tables of a few dozen or a few hundred rows for human consumption. This approach may be less than optimal, however, for returning a large number of Knowledge View rows.

There are definitely business cases for returning a large number of rows through Knowledge Views, and at BP Logix, we generally respond to such requests as follows:

“ What we want to do is identify the places where a large amount of rows are needed to be returned. For each of these Knowledge Views we should understand the reasons for the number of records being returned and then we can recommend other approaches. For example, some KViews may be replaced with the reports (which are capable of many more rows because we have control over the SQL sent to

“the DB), others may be needed because they are exporting to XLS and that could be a different KView or something scheduled nightly and email the XLS to the user. There are different approaches and steps to take, but the first step is to understand what the business reason is behind each KView returning so many records. As part of this it would also be important to know close to real-time this data has to be. If we are finding that a lot of this data is for reporting or exporting to something like XLS and it can be slightly old (e.g. 1 day or less), then we have more advanced configuration options with a dedicated DB for certain KViews and a Process Director [rendering server](#).”

Reports represent the inverse trade-off: design-and-build cost is higher, while execution time is lower. Reports enable you to hand-craft precisely the query you need, optimizing it as best you can. But in instances demanding that a large volume of results be produced on a relatively frequent basis, the extra up-front cost might be worth the effort.

Additionally, customers can run queries against the Process Director database. The schema for the database is [published and available in our documentation](#). To simplify such access further, the product also provides pre-built SQL views on tables containing process-related information, and enables you to easily generate SQL views on form information. It's not at all unusual for our customers to take advantage of these features to construct their own reports using third party report builders, for example.

## Other Knowledge View Topics

To see the other Knowledge View topics, navigate to them using the Table of Contents displayed on the upper right corner of the page, or use the links below.

[Knowledge View Definitions](#): Documentation for configuring a Knowledge View definition.

[Knowledge View Operations](#): Documentation for Knowledge View access methods, automation, and formatting.

[Knowledge View Exporting/Reporting](#): Exporting Knowledge View results to an external file for more detailed analysis.

## Knowledge View Definitions

A Knowledge View definition identifies what type of [Content List](#) objects will be returned, and how a user will navigate through them. The sample Knowledge View definitions that are created during installation are stored in a folder named [\[Knowledge Views\]](#), which is located in the Partition root folder. Knowledge View definitions can be stored anywhere in the Content List, under any folder structure.

To create a new Knowledge View, navigate to the folder into which you wish the Knowledge View to be created, then select the **Create New** dropdown and choose the **Knowledge View** menu entry.

To configure or view the properties of an existing Knowledge View, find the object in the [Content List](#), then click on its name, or click the check box next to the name and select the **Properties** hotlink.

Each Knowledge View definition contains several configurable tabs.

### Properties Tab #

The **Properties** tab defines the basic properties of the Knowledge View, and the objects you wish to associate with it.

Knowledge View Name      Icon

Processes

## PROPERTIES

**Description**

*Enter a brief description of this Object*

**Knowledge View Instance Name**

{KV\_DEF\_NAME} {{KV\_NUM\_ROWS}} items

eForms Associated with Knowledge View (optional)  ...

Workflows Associated with Knowledge View (optional)  ...

Timelines Associated with Knowledge View (optional)  ...

Case Definitions Associated with Knowledge View (optional)  ...

Script Associated with Knowledge View (optional)  ...

### Knowledge View Name and Description

The Knowledge View name and description are important because this is what a user will see when selecting a Knowledge View to run. A custom icon can also be configured for this Knowledge View definition. To change the icon used for this Knowledge View definition, click on the icon.

### Knowledge View Instance Name

This option enables you to specify a name for the Knowledge View instance when the Knowledge View runs.

### Forms Associated with Knowledge View

You can optionally link a Knowledge View to a specific Form definition in the [Content List](#). When this is configured:

- The Knowledge View will return only return objects or instances of the selected Form definition, and
- The selected Form will become the default Form for any Form-related filter selections. For instance, if you want to create a filter on a form field, only form fields from the selected Form will be available for selection. This is convenient if you have only one Form that needs to be associated with the KView.

### Workflows Associated with Knowledge View

You can optionally link a Knowledge View to a specific Workflow definition in the [Content List](#). When this is configured:

- The Knowledge View will return only return objects or instances of the selected Workflow definition, and
- The selected Workflow will become the default Workflow for any Workflow-related filter selections.

### Timelines Associated with Knowledge View

You can optionally link a Knowledge View to a specific Process Timeline definition in the [Content List](#). When this is configured:

- The Knowledge View will return only return objects or instances of the selected Process Timeline definition, and
- The selected Process Timeline will become the default Process Timeline for any Process Timeline-related filter selections.

### Case Definitions Associated with Knowledge View

You can optionally link a Knowledge View to a specific Case definition in the [Content List](#).

### Script Associated with Knowledge View

You can optionally link a Knowledge View to a custom script in the Content List.

### Select Excel Template (.xls, .xlsx, .xlsm)

When a Knowledge View is configured to export to Excel on the **Configure** Tab, this option will appear. You must use the **Object Picker** to specify an Excel file that exists in the [Content List](#) to use as the Excel template that will be used for the Export. Please see the [Knowledge View Exporting/Reporting](#) topic for details on exporting to Excel.

### Select Destination Folder for CSV/XLS (optional)

When a Knowledge View is exported to a CSV or Excel file, you can use the **Object Picker** to specify a [Content List](#) destination folder for the exported file. This setting is the equivalent to the `contentFolderPath` URL parameter for Knowledge View URLs. Please see the [Knowledge View Exporting/Reporting](#) topic for details on exporting to CSV.

### Select Database Data Source

When a Knowledge View is exported to a database table, you can use the **Object Picker** to select the Data-source for the database into which you desire to export the Knowledge View results as a table. This option is only available in Process Director v5.21 and higher.

### Configure Tab <#>

Additional configuration options are available in the **Configure** tab.

## Knowledge View Type

This defines the type of Knowledge View you want to display. When a type is selected, different options will display allowing you to configure your Knowledge View for the selected type.

**!** Changing this value will change the options available in the Perform this action when entry clicked property. Be sure to check the property to be sure it is correct after changing the Knowledge View Type property.

The following options are available:

- **Content List:** This is the default selection, and can return all [Content List](#) objects available in the system.
- **Task List:** This selection will return only user task items, and will open the items in task context.
- **Items I can Run:** This selection returns only [Content List](#) objects that the user has Run permission to execute, and will return a subset of the [Items I can Run](#) Global Knowledge View.

## Do not show this object in 'Items I Can Run' Knowledge Views

For a Knowledge View definition to not appear in the [Items I can Run](#) Global Knowledge View, you must uncheck the [Do not show this object in 'Items I Can Run' Knowledge Views](#) checkbox.

## Maximum number of items displayed in a result list

This controls the maximum number of items returned. If a larger number of items that match the filter criteria are returned, a message will be displayed to the user indicating that more records exist but were not displayed because of this configuration setting.

When exporting data and using the results in a Report, ensure this value is large enough to accommodate all of the possible items that will be returned, otherwise, some data will be missing.

An associated option is the checkbox labeled **Do not return any records if more than the maximum will be returned**. When this option is selected, Process Director will check to see if greater than the max number of results would be returned. If so, users will see a message informing them that the number of records returned is too large, and won't return any results. Using this option is advised and can result in improved overall system performance.



For process Director v4.04 and above, filter conditions will take precedence over this value for Task Lists. A Task List that contains both filter data and limits the number of the rows returned will correctly return all Task List entries that match the filter. Previously, it could return a smaller number of tasks than the limit defined in the Knowledge View filters, when the Maximum number of items was set to display a smaller number of records than the filter would return.

## Automatically refresh results (in seconds)

This option will automatically refresh the Knowledge View after the configured number of seconds. This option is useful for Knowledge Views that display data that changes in near real-time.

## Navigation Section

The **Navigation** section determines what type of a navigation tree should be presented to the end users. The navigation tree is optional. It is displayed as a vertical tree structure on the left side of the Knowledge View window. If a navigation tree is desired, it can be based on the folder structure in the **Content List** or it can be derived from your Meta Data schema's hierarchy.

Navigation	
<input checked="" type="radio"/>	No Navigation
<input type="radio"/>	No Navigation, start from folder
<input type="radio"/>	Folder Navigation
<input type="radio"/>	Category Navigation

### Folder navigation from “folder”

This indicates that a navigation window will appear in the left side of the running Knowledge View. The navigation tree that is displayed will be based on the folder structure in the **Content List**, and the folder structure will start from, and only include subfolders for, the folder you specify in this setting.

### Category navigation from “category”



This indicates that a navigation window will appear in the left side of the running Knowledge View. The navigation tree that is displayed will be based on the category tree hierarchy defined in the Meta Data taxonomy, and the category structure will start from, and only include subcategories for, the Category you specify in this setting.

### Folder navigation

This indicates that a navigation window will appear in the left side of the running Knowledge View. The navigation tree that is displayed will be based on the folder structure in the [Content List](#). When a user selects an item in the folder navigation, it will use that as filter criteria and only display items that are located in that folder and meet the defined filter criteria. The Content List, by the way, is nothing more than a Knowledge View that uses Folder Navigation.


### Category navigation


This indicates that a navigation window will appear in the left side of the running Knowledge View. The navigation tree that is displayed will be based on the category tree hierarchy defined in the Meta Data taxonomy. When a user selects a category in the navigation window, it will use that category as filter criteria and only display items that are assigned that category and meet the defined filter criteria.

### Knowledge View Results

This property controls where the results will display. The following options are available:

OPTION	DESCRIPTION
Displayed In Grid	Displays the results in a standard grid view.
Exported to Excel	Automatically exports the results to an Excel file. Selecting this option will add the <a href="#">Select Excel Template (.xls, .xlsx, .xlsm) (optional)</a> object picker to the list of associated objects for this Knowledge View on the <a href="#">Properties</a> tab. You must use this object picker to select the Excel Template to use for the export from the <a href="#">Content List</a> .
Exported to CSV	Automatically exports the results to a CSV file.
Run a Process	Automatically runs a process for each row in the results. Selecting this option will add the <a href="#">Select process to run on each object (optional)</a> object picker to the list of associated objects for this Knowledge View at the top of the <a href="#">Properties</a> tab. You must use this object picker to select the process you wish to run on each row of the Knowledge View.

 **When starting a process from a Knowledge View against returned form instances, the system will copy all attachment references from the original process associated with the form**

OPTION	DESCRIPTION
	<p>instance to the process instance that is being started by the Knowledge View. This behavior can be disabled using the <a href="#">fCopyRefsFromRealProcessToKViewProcess</a> Custom Variable.</p> <p> For Process Director v4.04 and higher, you can select either a Workflow or Process Timeline to run on each row. Earlier versions of Process Director only allow you to run a Workflow on each row of the Knowledge View.</p>
Calendar	<p>Date-based Knowledge View results can be displayed in a Calendar interface. This is useful for viewing the duration of a Process Timeline, Timeline Activity, etc, by using the start and end dates of the result to fill out the calendar view. You must configure the <b>Calendar</b> Tab to display the calendar properly. For more information, see the information below in the <b>Calendar</b> Tab section.</p>
Export to Database	<p>Automatically exports the results to a database table. Selecting this option will add the <b>Select Database Data Source</b> object picker to the list of associated objects for this Knowledge View on the <b>Properties</b> tab. You must use this object picker to select the Data-source object to use to export the data to a database. The option is available for Process Director v5.21 and higher.</p>

You can also optionally display button icons to export the Knowledge View results to Excel or CSV, and/or run a process based on the Knowledge View results. Process Director v5.21 and higher includes an option to show a database export icon.

### Export Database Table Name

When the **Show Export to Database Icon** option is selected, this option enables you to name the table that will be created to store the Knowledge View Results. When the Knowledge View runs, it will perform a DROP operation for any existing table with the same name, then perform a CREATE operation to create the table with the current data. The table name can accept system variables. You must also select a Data-source object, using the **Select Database Data Source** object picker on the **Properties** tab of the Knowledge View Definition, to specify the database into which the table will be created.

### URL to Call When Complete

When a Knowledge View's **Knowledge View Result** property is set to export the data to a different format, or to run a process, this property will appear. This property specifies a URL to call if

- The Knowledge View is run asynchronously, and
- The result is successful.

Once the processing is complete, the URL specified in this property will be called. Setting this property can be used to serialize multiple Knowledge Views. So, for example, KView1 can call the URL for KView2, which will in turn, call the URL for KView3, thus running all three Knowledge Views in serial.

When using this property for a Knowledge View that exports to CSV or Excel files, you must specify [the `dorun=1` URL parameter](#) as part of the URL that is called.

This property can also be used to enter the URL of a process to start when the Knowledge View Completes.

### Windows file path name to write the CSV/XLS to when running with URL parms

Enables you to specify a Windows file path for the exported Excel/CSV file. This is generally for running the Knowledge View via URL with URL parameters as described in the [Knowledge View Exporting and Reporting](#) topic.

### File "save as" name when downloading a CSV/XLS file

Enables you to specify a file name when exporting an Excel/CSV file.

### Export file name for CSV/XLS

This option will specify the file name of a Knowledge View exported to the [Content List](#), and must be used in conjunction with the [Select Destination Folder for CSV/XLS](#) option on the [Properties](#) tab.

### Export all Form Fields to CSV

This option will export all form fields for Forms returned in the results to a CSV file. This option appears only if the Export to CSV option is selected.

### Ignore Case context when running this Knowledge View

If the Knowledge View returns Case objects, the Knowledge View will ignore case mode when displaying the results. Normally a Knowledge View always filters results by case when it is run with case context. This option will direct the Knowledge View to ignore case context



This option shouldn't be set unless there is a specific reason to do so.

### When form instance is opened treat like it was opened from a Task List

By default, Knowledge Views do **not** open forms in task context, so, even if the user has a task assigned for that form, no task completion buttons will be displayed, and the user will be unable to complete the task by opening the form from a Knowledge View.

When this option is selected, opening a Form instance from the Knowledge View will open it in task context. If you have a task active for the Form, you can complete the task by using the task completion buttons on the form.

### This Knowledge View will only use cached Form Instance Data

This option is only available when a Knowledge View returns data from a form that has Form Data Caching enabled. When this box is checked, the Knowledge View will display only the form data that is stored in

the Form cache. As a result, the Knowledge View may run significantly faster, but may not contain the most current data. For more information, please see the [Form Data Caching](#) topic.

### Column Header Style

This option gives you the choice between selecting a fixed header that won't scroll when you scroll down the page, or a scrolling header.

### Filter Type Style

This option enables you to choose whether to allow users to expand or collapse the Filters for the Knowledge View, and determine the initial view state of the filter section, i.e., expanded or collapsed.

### Style

The dropdown for this property contains a list of preset visual styles that can be applied to the Knowledge View results. Selecting a style from this dropdown will replace the default color scheme with the selected style.

### Use Process Director 6.0 style

For Process Director v6.0 and higher, this property enables/disables the display of the v6.0 UI for the Knowledge View results.


### Render Mode

There are two options for this property: Classic and Lightweight.

Classic rendering is the long-standing rendering mode of the Knowledge View, with extensive use of table elements for the layout and images for backgrounds, icons, etc.

Lightweight rendering leverages HTML5 and CSS3, which may be more appropriate for mobile devices, but the control may lose its rounded corners, gradients and shadows in non-modern browsers.

### Perform this action when entry clicked



Perform this action when entry clicked

This property enables you to specify what action is taken when an entry is clicked in the result set. Depending on the option you select, additional options will appear in the user interface.

### Available Options

- **Open:** The item will be opened in the browser. For instance, if the Knowledge View returns Form Instances, clicking on the item when this option is selected will open the Form.
- **Properties:** The item's properties will be displayed in the browser.
- **Download:** The item will be downloaded to the user's computer.
- **Run:** The item will run.
- **Attach Link:** Usually used in conjunction with an [AttachKView](#) control on a Form, this option will attach the Knowledge View to the Form.
- **Case Folder View and Open:** Enters case mode and open the item.
- **Case Folder View Only:** Enter case mode for the item's case.

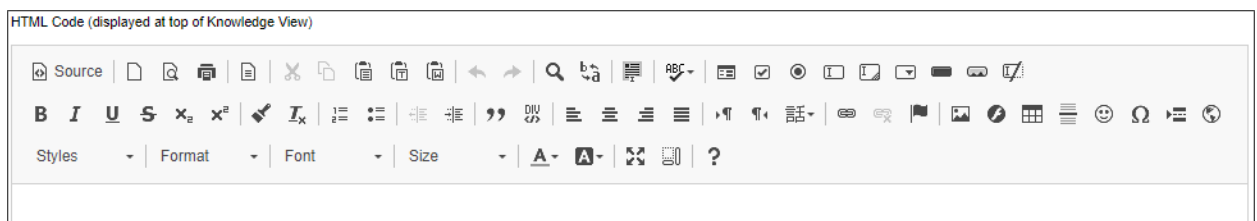
## Additional Options

If you select the [Open](#), [Properties](#), [Run](#), or [Case Folder View and Open](#) options from the Available Options above, an additional dropdown will appear that enables you to select how you want the item displayed to the user.

- **Inline In the Browser:** Opens the item directly in the browser. This is the default choice, and is generally the best option. (**NOTE:** Forms will always open in a new browser window.)
- **In a Popup Window:** A new popup window will appear that contains the selected item.
- **and replace entire Window:** The entire current window will be replaced by the selected item.
- **and replace entire Portlet:** If the Knowledge View appears in a Workspace portlet, the selected item will replace the Knowledge View inside the portlet.
- **and display in target Portlet:** If the Knowledge View appears in a Workspace portlet, the selected item will appear in the selected target portlet. When this option is selected, an additional dropdown, labeled [in workspace portlet #](#), will appear, and you can select the desired Portlet number in which to display the item from this dropdown control.

## HTML Code

This option enables you to enter arbitrary HTML text that will appear at the very top of the Knowledge View when it is displayed to users. For users of v5.23 and higher, this field will use the same text editor as the Online Form Designer to make formatting the text easier without having to know the actual HTML code to apply formatting.



## Options Tab #

The **Options** tab defines the object types the Knowledge View will return, and determines which user interface controls or icons will be available to the user. Providing these controls will give the end user additional actions to enhance reporting and provide more control of objects in Process Director. These controls are used throughout the system (e.g. [Task List](#), [Content List](#), etc.).

The options available on this tab are dependent upon the [Knowledge View Type](#) you select on the **Configure** tab, i.e., [Content List](#), [Task List](#), or [Items I can Run](#).

### Content List Knowledge View Options

The [Content List Knowledge View Type](#) can return all Content List items. This type requires the user to have Read permissions on the objects that are returned. Most Knowledge Views will be of this type.

Content List Knowledge Views have the following options.

OPTIONS				
Allow These Actions on Objects <input type="checkbox"/>				
<input type="checkbox"/> Copy / Move	<input type="checkbox"/> Delete	<input type="checkbox"/> Properties	<input type="checkbox"/> Meta Data	
<input type="checkbox"/> Permissions	<input type="checkbox"/> Export	<input type="checkbox"/> Run	<input type="checkbox"/> Attach Link	<input type="checkbox"/> Cancel Process
Return These Object Types <input type="checkbox"/>				
<input type="checkbox"/> Form Definitions	<input type="checkbox"/> Workflow Definitions	<input type="checkbox"/> Timeline Definitions	<input type="checkbox"/> Knowledge View Definitions	<input type="checkbox"/> Case Definition
<input type="checkbox"/> Form Instances	<input type="checkbox"/> Workflow Instances	<input type="checkbox"/> Timeline Instances	<input type="checkbox"/> Case Instances	<input type="checkbox"/> Documents
<input type="checkbox"/> Business Rules	<input type="checkbox"/> Data Sources	<input type="checkbox"/> DropDown Objects	<input type="checkbox"/> Folders	
<input type="checkbox"/> Reports	<input type="checkbox"/> Charts	<input type="checkbox"/> Dashboards	<input type="checkbox"/> Data Lists	
<input type="checkbox"/> Business Values	<input type="checkbox"/> Goals	<input type="checkbox"/> Machine Learning Definitions	<input type="checkbox"/> Stream Actions	
Display Action Icons for <input type="checkbox"/>				
<input type="checkbox"/> Children	<input type="checkbox"/> References	<input type="checkbox"/> Properties	<input type="checkbox"/> Attach Link	<input type="checkbox"/> View
<input type="checkbox"/> Download	<input type="checkbox"/> View Processes	<input type="checkbox"/> Manage Process Users	<input type="checkbox"/> Case Folder Link	
Users Will Have Permission to Create <input type="checkbox"/>				
<input type="checkbox"/> Folder	<input type="checkbox"/> Form Definition	<input type="checkbox"/> Workflow Definition	<input type="checkbox"/> Process Timeline	<input type="checkbox"/> Document
<input type="checkbox"/> Knowledge View Definition	<input type="checkbox"/> Dashboard	<input type="checkbox"/> Import Objects	<input type="checkbox"/> Chart	<input type="checkbox"/> Report
<input type="checkbox"/> DropDown	<input type="checkbox"/> Case	<input type="checkbox"/> Business Rule	<input type="checkbox"/> Data Source	<input type="checkbox"/> Goal
<input type="checkbox"/> Business Value	<input type="checkbox"/> Machine Learning Definition	<input type="checkbox"/> Stream Action	<input type="checkbox"/> Data List	<input type="checkbox"/> Template

For users of Process Director v5.08 and higher, new functionality enables Knowledge Views that return documents to also return the form context for the form used to upload the document. With this context, the Knowledge View can return Form field data from the in-context forms.

### Allow These Actions on Objects

This option will allow users to run actions on a result item if checked. In cases where the Knowledge View will be displayed to end users, these options should all be unchecked.

Allow These Actions on Objects <input type="checkbox"/>				
<input type="checkbox"/> Copy / Move	<input type="checkbox"/> Delete	<input type="checkbox"/> Properties	<input type="checkbox"/> Meta Data	
<input type="checkbox"/> Permissions	<input type="checkbox"/> Export	<input type="checkbox"/> Run	<input type="checkbox"/> Attach Link	<input type="checkbox"/> Cancel Process

### Return These Objects Types

This setting determines the types of objects that will be returned by the Knowledge View. In most cases, Knowledge Views that will be displayed to end users will only be configured to return Form Instances, with all other objects unchecked.

Return These Object Types <input type="checkbox"/>				
<input type="checkbox"/> Form Definitions	<input type="checkbox"/> Workflow Definitions	<input type="checkbox"/> Timeline Definitions	<input type="checkbox"/> Knowledge View Definitions	<input type="checkbox"/> Case Definition
<input type="checkbox"/> Form Instances	<input type="checkbox"/> Workflow Instances	<input type="checkbox"/> Timeline Instances	<input type="checkbox"/> Case Instances	<input type="checkbox"/> Documents
<input type="checkbox"/> Business Rules	<input type="checkbox"/> Data Sources	<input type="checkbox"/> DropDown Objects	<input type="checkbox"/> Folders	
<input type="checkbox"/> Reports	<input type="checkbox"/> Charts	<input type="checkbox"/> Dashboards	<input type="checkbox"/> Data Lists	
<input type="checkbox"/> Business Values	<input type="checkbox"/> Goals	<input type="checkbox"/> Machine Learning Definitions	<input type="checkbox"/> Stream Actions	

### Display Action Icons for

This option allows you to display Action icons for each result displayed. In cases where the Knowledge View will be displayed to end users, these options should all be unchecked.

Display Action Icons for <input type="checkbox"/>				
<input type="checkbox"/> Children	<input type="checkbox"/> References	<input type="checkbox"/> Properties	<input type="checkbox"/> Attach Link	<input type="checkbox"/> View
<input type="checkbox"/> Download	<input type="checkbox"/> View Processes	<input type="checkbox"/> Manage Process Users	<input type="checkbox"/> Case Folder Link	

### Users Will Have Permission to Create

This enables users with the appropriate permission the ability to create new objects in the [Content List](#). In cases where the Knowledge View will be displayed to end users, these options should all be unchecked.

Users Will Have Permission to Create <input type="checkbox"/>				
<input type="checkbox"/> Folder	<input type="checkbox"/> Form Definition	<input type="checkbox"/> Workflow Definition	<input type="checkbox"/> Process Timeline	<input type="checkbox"/> Document
<input type="checkbox"/> Knowledge View Definition	<input type="checkbox"/> Dashboard	<input type="checkbox"/> Import Objects	<input type="checkbox"/> Chart	<input type="checkbox"/> Report
<input type="checkbox"/> DropDown	<input type="checkbox"/> Case	<input type="checkbox"/> Business Rule	<input type="checkbox"/> Data Source	<input type="checkbox"/> Goal
<input type="checkbox"/> Business Value	<input type="checkbox"/> Machine Learning Definition	<input type="checkbox"/> Stream Action	<input type="checkbox"/> Data List	<input type="checkbox"/> Template

### Task List Knowledge View Options

The *Task List Knowledge View Type* will only display actionable items for the current user. Using this type will allow the system to dynamically refresh the [Task List](#) to allow items to be added, updated, or removed.

Task List Knowledge Views also incorporate the same *Perform this action when entry clicked* settings as the Content List Knowledge View type, as described above.

### Show These Task Types

This option enables you to display specific task types.

Show These Task Types <input type="checkbox"/>			
<input type="checkbox"/> Checked Out Objects	<input type="checkbox"/> Incomplete Forms	<input type="checkbox"/> Workflow Tasks	<input type="checkbox"/> Timeline Tasks

**i** Please note that if you select *Incomplete Forms* as a task type to return, the Knowledge View will return ALL incomplete Forms the user has created in the system, *irrespective of any filters you might configure on the Filter tab of the Knowledge View definition*. The behavior is implemented by design.

## Items I Can Run Knowledge View Options




The *Items I Can Run Knowledge View Type* will only display definitions that can be run or submitted by users with Run permission on the objects returned.

### Return These Object Types

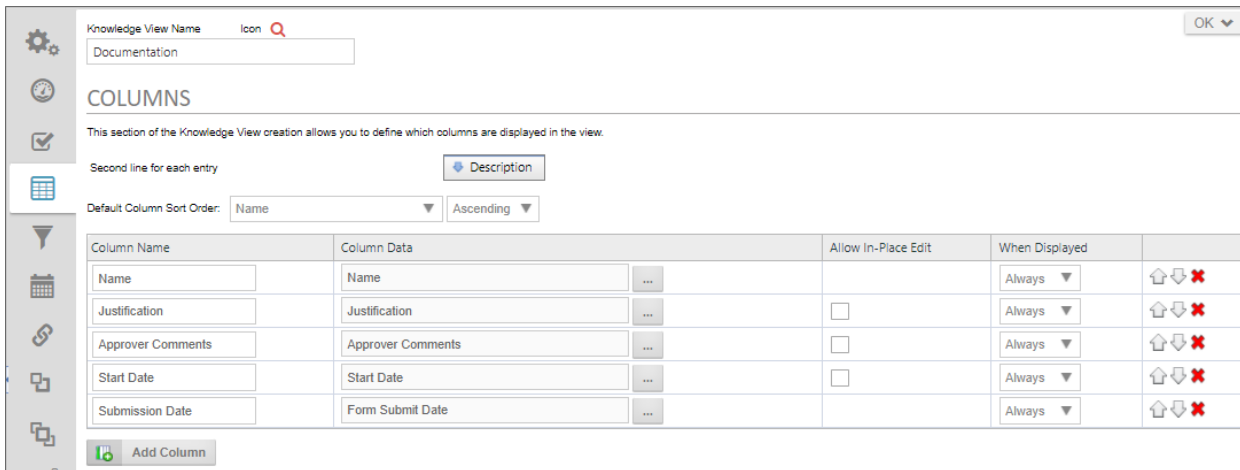
This option enables you to specify what definition types to display.

Return These Object Types				
<input type="checkbox"/> Form Definitions	<input type="checkbox"/> Workflow Definitions	<input type="checkbox"/> Timeline Definitions	<input type="checkbox"/> Knowledge View Definitions	<input type="checkbox"/> Case Definition
<input type="checkbox"/> Reports	<input type="checkbox"/> Charts	<input type="checkbox"/> Dashboards	<input type="checkbox"/> Data Lists	

## Columns Tab #

The **Columns** tab defines the columns to display in the Knowledge View. The column data can include form fields, process data, Meta Data, or various object properties. The order of the columns can be changed using the up arrow  and down arrow  to the right of the column. Columns can be removed using the  icon.

The sort order can be configured in the Knowledge View; however the end user can override this by clicking on the column names in the Knowledge View's display. The sort order is **not** used when exporting the results to a CSV or an Excel Report, it is only used when the results are displayed in the browser.



Column Name	Column Data	Allow In-Place Edit	When Displayed	
Name	Name	<input type="checkbox"/>	Always	↑ ↓ ✖
Justification	Justification	<input type="checkbox"/>	Always	↑ ↓ ✖
Approver Comments	Approver Comments	<input type="checkbox"/>	Always	↑ ↓ ✖
Start Date	Start Date	<input type="checkbox"/>	Always	↑ ↓ ✖
Submission Date	Form Submit Date	<input type="checkbox"/>	Always	↑ ↓ ✖

### Second line for each entry

This defines optional data that should be used on the second line of every item returned. The default is the description of the item.

### Default column sort order

This defines the default sort order of the data displayed in the list. The columns that are configured will appear in this dropdown list of possible fields to sort on. The user can override the sorting by clicking on a column.

The sort order isn't used when exporting the results to a CSV or an Excel Report.

### Column Name



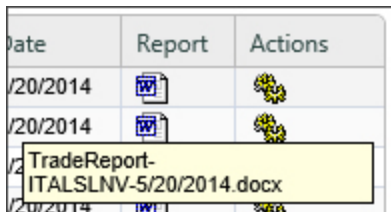
This is the column name that will be displayed in the Knowledge View. This is important when using the reporting options. For more information refer to the [Knowledge View Exporting and Reporting](#) topic.

### Column Data

This defines the data to display. The data can be form data, process status, or other information about the item being returned. When displaying form data, a dropdown will be displayed with the list for form fields if the optional [Forms Associated with Knowledge View](#) link has been configured in the [Properties](#) tab.

Array fields can also be configured as a column to be returned by the Knowledge View. When configured in this way, a separate Knowledge View row will be returned for every row in the array. That array field expansion will only occur if you have configured the [Forms Associated with Knowledge View](#) link to on the [Properties](#) tab.

When a column is set to display attachments, an icon for each attachment will appear in the column. The icons provide a link at which you can download or view the attachment, and mousing over them will display the name of the file or reference,



If the Knowledge View returns data from two different Form definitions, you can mix the use of the Form Field Picker and FORM system variables. Process Director will correctly parse the system variables to return the appropriate data.







Column Name	Column Data	Allow In-Place Edit	When Displayed	
Name	Name		Always	↑↓✖
Field From Form 1	Justification		Always	↑↓✖
Field from Form 2	{FORM:Author}		Always	↑↓✖
Last Updated	Update Date		Always	↑↓✖
Version	Version		Always	↑↓✖

Field from Form 2, identified via System Variable

This behavior can be turned off using the [fEnableMultFormFieldsInCols](#) custom variable.

**!** Note that, when using a Knowledge View to return process instances, and you wish to display columns returning form data from a non-default form instance, Process Director v.3.45 and above supports getting form data from the non-default form instance in the process using the system variable syntax `{FORM:NonDefaultFormFieldName}`. You should remember, however, that, if multiple forms are included in a Knowledge View, *the Knowledge View will only return data from a single form instance in a single row*. You can't have form fields from two different form instances, such as the default and the non-default Form, display as fields in a single row in the Knowledge View.

You have the option to run a process for the row by choosing **Run Process** from the **Object Information** menu item, then selecting the process you which to run for that row. When you do so, an icon for the process will appear in every row of the Knowledge View, like so:

Name ▲	Run a Process
Original Form 123	
Second Form 126	
Webinar 20210929 Original Form 130	
Webinar 20210929 Original Form 146	
Webinar 20210929 Second Form 133	
Webinar 20210929 Second Form 149	

Clicking one of the icons in the **Run a Process** column will start the selected process, and attach the row object to the process. This is a different feature from the **Run a Process** selection in the Knowledge View results dropdown in the options tab, which is explained above.

Finally, for Task List KViews, you have an option to display **Task Completion Icons**. When this option is selected, be sure that the **Allow this result to be completed from the users Task List** property is selected from the **Options** tab of *each* result in the Timeline Activity's **Results** tab. Otherwise, the icons you've configured for the results won't appear properly.

### Allow In-Place Edit

For Process Director v5.44.500 and higher, Knowledge View that return Form Instance data can be configured to allow end users to perform in-place editing of Form field data directly from the Knowledge View's grid display. Checking the **Allow In-Place Edit** check box, turns on in-place editing for the selected field. Designers can configure none, some, or all Form fields for in-place editing.

In addition to checking the checkbox for the **Allow In-Place Editing** property, the **Options** tab must be configured to ensure that the **Allow These Actions on Objects** section has the **Properties** box checked, as well as the **Children** box checked in the **Display Action Icons** for section.



For some versions later than v5.44.500 but earlier than v6.0.0 (Beta), issues with in-Place Editing could cause this feature to operate incorrectly. BP Logix recommends an upgrade to v6.0.1 or higher to fully implement this feature.

### When Displayed

When configuring columns in a Knowledge View, you have the ability to control whether the columns will be visible when the results are shown in the browser, whether the column is visible when exporting to CSV, when the column is visible when exporting to an Excel report, or whether the column is always visible.

This feature is also partially controlled by the **Knowledge View Results** property setting you configure on the **Configure** tab:

- When "Display the results of the Knowledge View in the browser" is chosen, the default is all columns will be displayed for all destinations.

- When “Export the results to a Microsoft Excel report” is chosen, the default operation is to display the first column only to the web browser, and to display all columns in the Excel report.
- When “Export the results to a CSV file” is chosen, the default operation is to display the first column only to the web browser, and to display all columns in the CSV export.

You can change this by selecting from the **When Displayed** dropdown. The following options are available in the dropdown:

- **Never:** This column will never display.
- **Always:** This column is always visible.
- **Desktop:** This column will only visible in a desktop browser.
- **Excel:** This column will only be visible in the Excel Report.
- **CSV:** This column will only be visible in the CSV export.
- **Script:** This column will only be visible in script APIs.
- **Report:** This column will only be visible in Reports.
- **Desktop:** This column will only visible on a mobile device.

For example, use the column name “Process Timeline Name” and set to Always display the “Process Timeline Name” column. Use the column name “Start Date” and set to Excel to only display the “Start Date” column in the Excel report.

For Process Director v5.1 and higher, Knowledge Views will **not** evaluate columns that are configured to display in a different output format than the current display, in order to reduce resource usage. For instance, columns that are only displayed in an export won't be evaluated when the Knowledge View is displayed in the desktop browser. For legacy applications that use Knowledge View Scripts, upgrading to this or later versions will affect those scripts. To accommodate this behavior, you'll need to add a column with the type "Script" that will always be evaluated and sent to a script regardless of the destination output.

## Filter Tab #

The **Filter** tab identifies filter criteria that enables you to restrict the records the Knowledge View returns to those that meet specific filter criteria. The primary use for the **Filter** tab is to create searches for specific data that may be contained in a large set of records. The filter criteria can include form field data, categories, attribute values, or various object properties (e.g. author, document type, published). The filter criteria can identify what document versions to display – the latest document version or only the published version.

The end user can optionally be given the ability to override any of the filter values. This allows the field name and the input box to be displayed on the running Knowledge View so the user can input the additional filter information.

The screenshot displays the configuration interface for a Knowledge View. On the left is a vertical sidebar with icons for settings, a gauge, a checkmark, a calendar, and a funnel. The main area contains a 'Knowledge View Name' field with the text 'Processes' and an 'Icon' field with a document icon. Below this is a 'FILTER' section with a checked checkbox labeled 'Perform search immediately when Knowledge View is loaded' and an 'Add Condition' button with a plus icon.

### Perform search immediately when Knowledge View is loaded

This indicates that the Knowledge View search should occur immediately when it is run (opened). If this check box isn't turned on, the Knowledge View will display with an empty list until the **Search** button is pressed to run the search.

### Filter Conditions

The **Filter Conditions** identify what items are available to this Knowledge View according to their properties. Only items with matching conditional property values will be available to this Knowledge View.

Refer to the [Condition Builder section](#) of this guide to create your Knowledge View filter conditions.

### Prompt user for value?

This checkbox can be set for any of the filter criteria. This allows the user to override the filter value to locate objects. For each filter field that has this box checked, an input field will be displayed on the top portion of the user Knowledge View, when the user can use to filter the data the Knowledge View returns.

If you are creating filter criteria based on the result of a dropdown or radio button list, you can limit the values used to filter the search to the values available in the dropdown. This way, neither you nor a user can enter a filter value that isn't on the dropdown. This requires that the dropdown be linked with a Drop-down Object.

## Complex Filter Considerations

In general, when creating filters, you should bear in mind the type of object the Knowledge View returns. For instance, if the Knowledge View returns Form instances, filters work best when filtering on Form data. Conversely, trying to filter Form instances based on data stored in other objects, like Process Timelines, may return unexpected results. For instance, if a Form is used in more than one Process Timeline, and you try to filter the Form instances based on Timeline data, Process Director will evaluate whether the **any** Timeline associated with the Form meets the filter criterion you've designated. In that case, the Knowledge View may return multiple rows that contain the same Form instance, because more than one process meets the criteria.

For a Form associated with multiple processes, when the Form is returned by a Knowledge View, additional logic runs in the background of the Knowledge View to associate the Form with the process that started first, in addition to any running processes. This logic, when considering what to return in the Knowledge View, makes the following decisions about which Form to return in the Knowledge View:

- If the form is affiliated with a single running process, always use that process.
- If the form is affiliated with more than one running processes, use the oldest.
- If the form is affiliated with no running processes, but is affiliated with completed processes, use the oldest.

This decision tree ensures that, when returning the possible Form Instances, Forms that are associated with a running process are always displayed.

Filtering applied to array fields is complex as well. When Process Director returns values from an array, it returns all of the values in each column as a comma-separated list. If you apply a filter to an array field, and **any** value in the array column matches the filter, then the Knowledge View will count the filter condition as being met. The filter doesn't parse every row in the array, and make an individual "match" decision for each row.

Moreover, when you return Form instances that are filtered on an array, Process Director will return a Form Instance *for each row in the array*. In other words, if your Knowledge View returns Form instances, and a Form has three rows in the array you're attempting to filter, then three copies of the Form instance will be returned. As a result, filtering on an array often requires custom Knowledge View scripting to parse the array and filter each row individually. Additionally, users of the Advanced Reporting Component could use that component to perform the filtering, and present the filtered results in a Report, instead of using a Knowledge View.

A more comprehensive look at array conditions is presented in the [Using Arrays in Conditions](#) section of the [Condition Builder](#) topic.

## Calendar Tab #

The **Calendar** Tab enables users to view date-based Knowledge View results in a calendar interface. This is useful for viewing the duration of an Activity or Process Timeline, for example.

### CALENDAR

The Calendar mode of the Knowledge View will display the results in a table representing the specified Calendar View.

Initial Calendar View:  Day  Week  Month  Timeline

Initial Calendar View Date:  Use a properly formatted date, system variable, or blank for current date

Column for Subject:  ▼

Column for Start Date:  ▼

Column for End Date:  ▼

Color:  ▼

In order for the Calendar view to work properly you need to supply the Calendar interface with Knowledge View columns that supply the name you'd like to display in the calendar entry, the start date, and the end date for the calendar item. The following properties are configurable.

### Initial Calendar View

This setting determines whether you'd like to initially display the Knowledge View results calendar in a daily, weekly, or monthly view. Please note that, while this does control the initial display of the calendar, users can select a different view after the calendar is displayed.

### Initial Calendar View Date

This setting was added in Process Director v6.0, to enable you to specify the initial start date to display for the calendar. This can be set to a fixed, properly formatted date, or can use a System Variable like {CURR\_DATE} to dynamically set the **Initial Calendar View Date** when the Knowledge View is run.

### Column for Subject

The Knowledge View column to display as the subject of the calendar entry.

### Column for Start Date

The Knowledge View column containing the start date for the calendar entry.

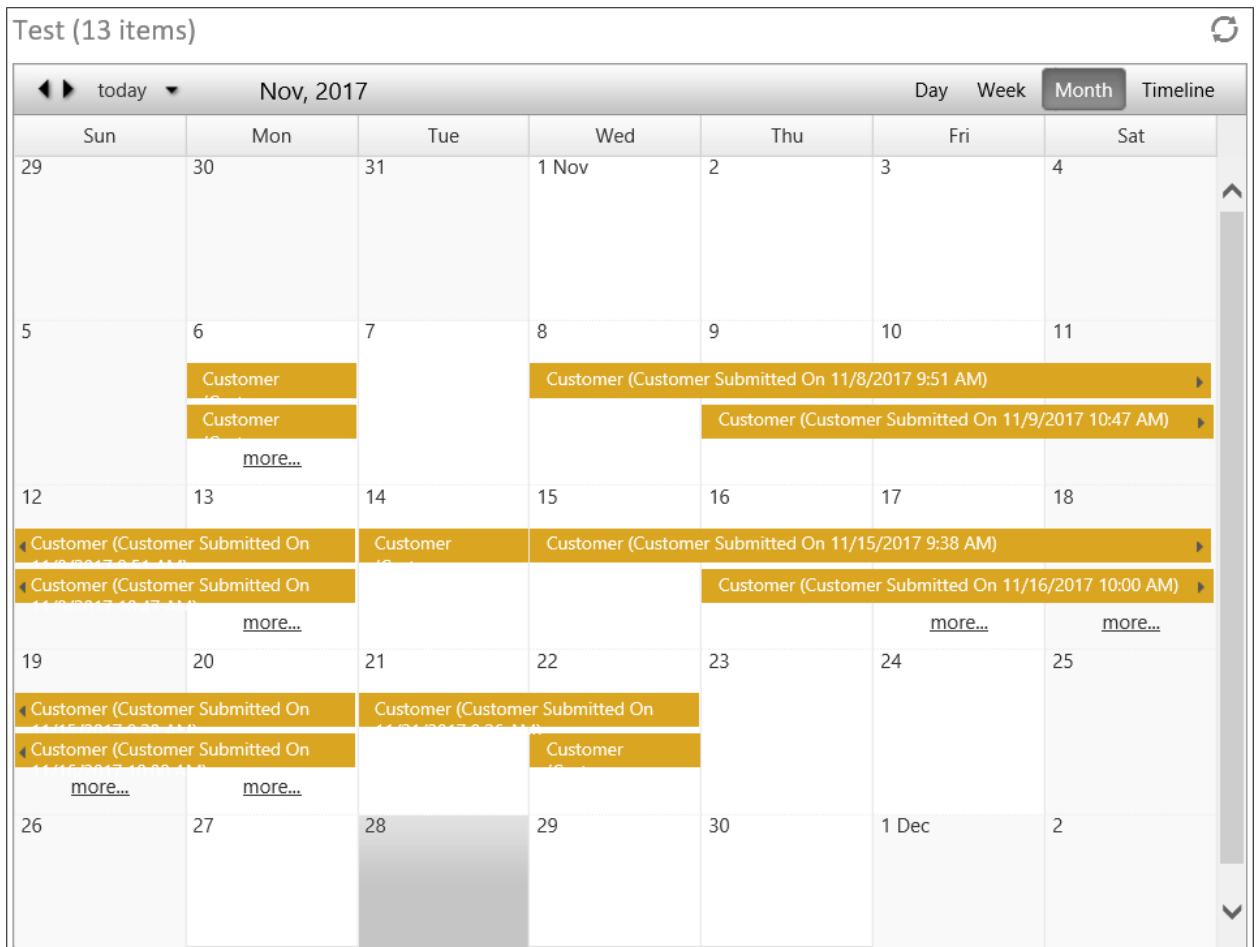
### Column for End Date

The Knowledge View column containing the end date for the calendar entry. The calendar entry will span the appropriate dates between the start date and end date.

### Color

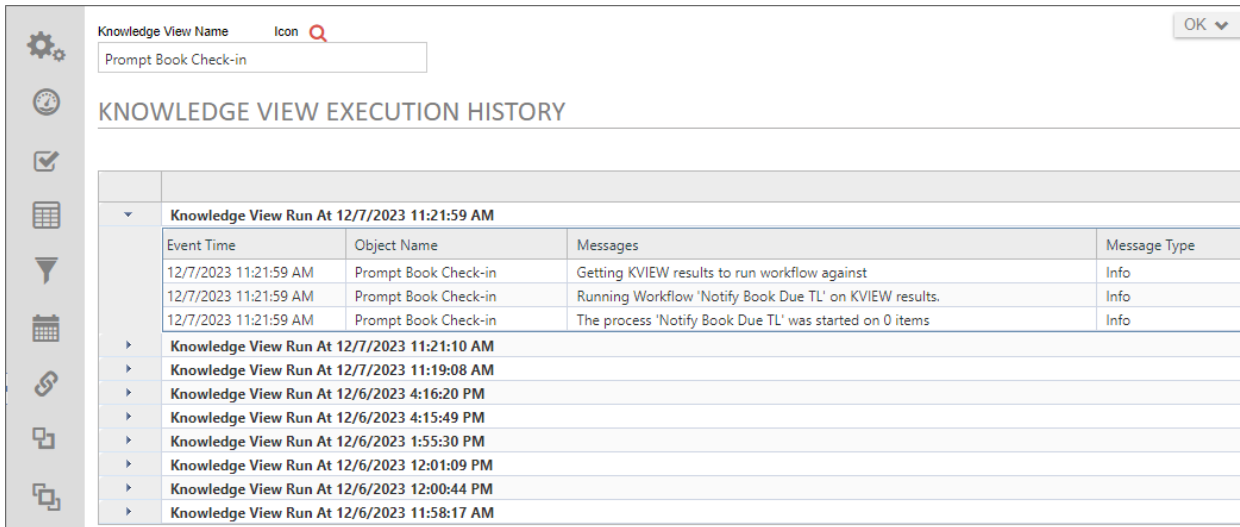
A Knowledge View column that displays the color to use to display the calendar entry. You can manually set the color value for the column, or use a Business Rule or system variable to determine this column's value. Whichever method you use to supply the column's value, the color value must be either a hexadecimal HTML color, e.g., "#FF0000", or a named HTML color, e.g., "AliceBlue".

Once the values have been supplied, the Calendar view will appear similar to the example below.



### Knowledge View Execution History #

The **Knowledge View Execution History** tab enables you to see a list of executions that specify when a Knowledge View that invokes a process was run.



Event Time	Object Name	Messages	Message Type
Knowledge View Run At 12/7/2023 11:21:59 AM			
12/7/2023 11:21:59 AM	Prompt Book Check-in	Getting KVIEW results to run workflow against	Info
12/7/2023 11:21:59 AM	Prompt Book Check-in	Running Workflow 'Notify Book Due TL' on KVIEW results.	Info
12/7/2023 11:21:59 AM	Prompt Book Check-in	The process 'Notify Book Due TL' was started on 0 items	Info
Knowledge View Run At 12/7/2023 11:21:10 AM			
Knowledge View Run At 12/7/2023 11:19:08 AM			
Knowledge View Run At 12/6/2023 4:16:20 PM			
Knowledge View Run At 12/6/2023 4:15:49 PM			
Knowledge View Run At 12/6/2023 1:55:30 PM			
Knowledge View Run At 12/6/2023 12:01:09 PM			
Knowledge View Run At 12/6/2023 12:00:44 PM			
Knowledge View Run At 12/6/2023 11:58:17 AM			

Most Knowledge Views do not display an execution history. Knowledge views that simply return data don't execute anything. They merely display a list of records when run.

Knowledge views that start a process, on the other hand, do perform executions, and this tab enables you to track all of the instances in which the Knowledge View ran, and the results of each execution cycle.

## More Information

Continue to the [Knowledge View Exporting and Reporting](#) topic for more information.

## Knowledge View Operations

Knowledge Views are a collection of related objects based on categorization or other criteria. They can be made available to either authenticated or anonymous users depending on the permissions. In addition to displaying Knowledge Views in the UI, such in a workspace or Dashboard portlet, Knowledge Views can be accessed directly via a URL, and can be used to prompt processes on a manual or scheduled basis.

## Knowledge View URLs for External Access #

A Knowledge View is represented by a URL that can be displayed outside of the BP Logix environment. The URL for a Knowledge View can be sent in an email, displayed in a web portal, or added to an existing web site. The URL format for an individual Knowledge View is:

```
https://ServerName.com/kv.aspx?ext=1&id=NN
```

Where [ServerName.com](#) is the hostname of BP Logix and [NN](#) is the internal ID displayed in the Knowledge View definition you want to use (e.g. ID=ea1f7d30-2c44-4971-ac8a-4654ae2521ff).

An additional URL parameter, "gkview", can be used to enable a Global Knowledge View to be displayed using the Knowledge View name instead of the ID, but it can only be used to return Global Knowledge Views that are [Task Lists](#), e.g. kv.aspx?gkview=Task List.

For user of Process Director v5.37 and higher, a URL Parameter, "canceltext", can be used on a [Task List](#) URL and will display the configured text in an alert box when a user clicks on the **Cancel** button on the form.



[https://ServerName.com/kv.aspx?canceltext="You clicked the Cancel button."](https://ServerName.com/kv.aspx?canceltext=)

## Passing Filter Values in the URL #

The Knowledge View can be opened using a URL that is placed on a web site. If the Knowledge View is configured with Filter, you can pass this filter information to the Knowledge View on the URL. Simply set the desired filter result as a variable, then enter the variable name you'd like to use in the URL parameter like the example below:

**FILTER**

Perform search immediately when Knowledge View is loaded

InputState	...	Contains	▼	Variable	
				Var1	<input type="checkbox"/> <span style="color: red;">✖</span> <span style="color: green;">↻</span>
[ AND ]					
InputZip	...	Contains	▼	Variable	
				Var2	<input checked="" type="checkbox"/> <span style="color: red;">✖</span> <span style="color: green;">↻</span>

+ Add Condition

Alternatively, you can use a String for the filter, and use the syntax `{VAR:VarName, modifier=ModifierValue}` to use additional formatters/modifiers for the variable.

Once the variables have been configured, you can pass the desired values via the URL parameters using this syntax with the query string parameters:

<https://ServerName.com/kv.aspx?ext=1&id=NN&var1=abd&var2=123>

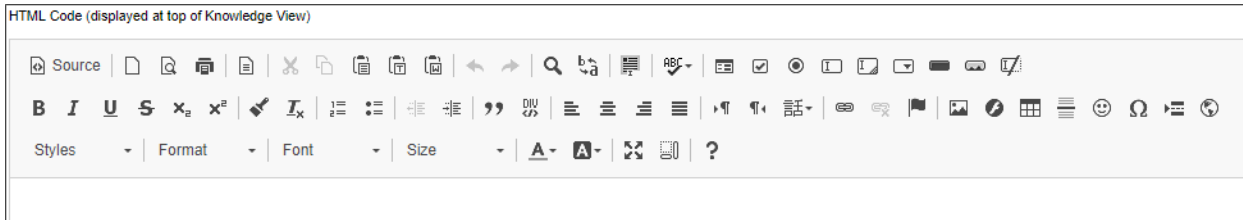
This configuration uses the Variable object in the same way as [Form Definition URLs](#).

**!** Sometimes, variables must be passed through multiple objects before reaching a Knowledge View, or other object. For instance, you may have a Form that calls a Chart, which, in turn, calls a Knowledge View via a drill-down. The data on the Form is passed to the Chart via a variable that is accessed using the normal `{VAR:ParmName}` syntax. However, when the chart does a drill-down to a Knowledge View, the Knowledge View does not have access to the original `{VAR:ParmName}` variable. Instead you need to append two underscores and the number "2" (`__2`) to the variable name, e.g. `{VAR:ParmName__2}`. When you do so, Process Director treats the `"{VAR:ParmName__2}"` in the Knowledge View as a pointer to the original value for `{VAR:ParmName}`, so that the proper value is automatically available to the Knowledge View.

## Knowledge View Formatting #

Each Knowledge View can implement a different look and feel by using the **HTML Code** property in the **Configure** tab of the Knowledge View definition. This property enables HTML formatting to be applied to the top of the running Knowledge View. The Knowledge View window can also be framed in HTML frames

and use a custom style sheet (CSS). You can also display the Knowledge View in a form by using the **Knowledge View** control.



## Running Processes on Knowledge View Results #

Processes can be run on the results of Knowledge View using two different approaches. One approach will automatically run a process on every item returned, the other approach allows a user to click on a special column to run a process on the selected item.

### Approach 1: Run a Process on all the results

**!** For Process Director v4.04 and above, you can select either a Workflow or Process Timeline to run on each row. Earlier versions of Process Director *only allow you to run a Workflow* on each row of the Knowledge View.

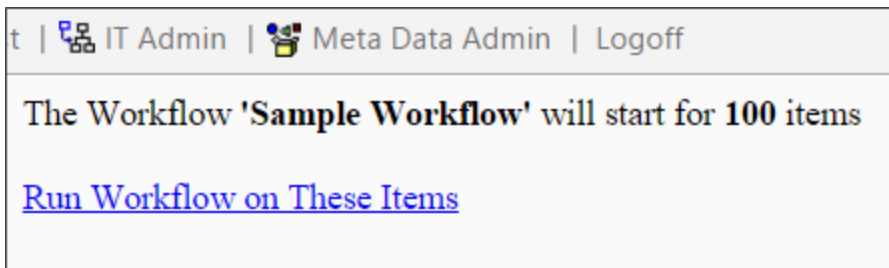
On the Knowledge View's **Configure** tab, set the **Knowledge View Results** property dropdown to "Run a Process".

Once you have set the **Knowledge View Results** property, click the **Update** menu button to save the configuration. When you do so, a new option will appear in the **Properties** tab, named **Select process to run on each object**.



Use the **Object Picker** for this option to select the process definition, usually a Process Timeline, that you want to start for each item returned by the Knowledge View. Finally, click the **Update** menu button again to save your configuration.

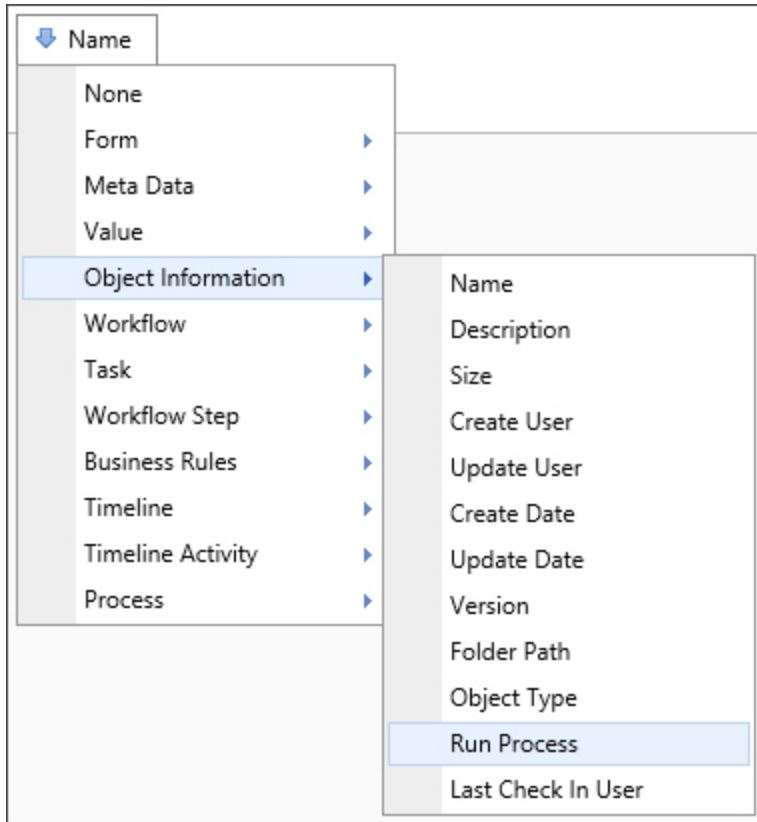
Now, when you run the Knowledge View, Process Director will display a confirmation message, with a link to click if you wish to run the Process.



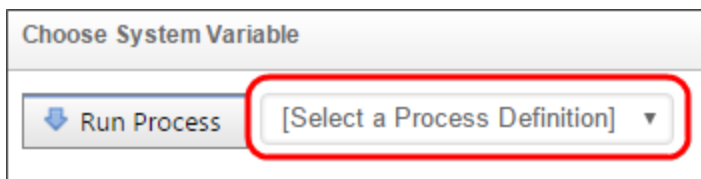
**i** If you schedule this Knowledge View to run automatically, be sure to put the "dorun=1" URL parameter in the Knowledge View URL, as described in the *Scheduling a Knowledge View* section below.

### Approach 2: Run a process on certain items

To enable a user to run a process on specific items in the Knowledge View results, add a column to the Knowledge View, and set the **Column Data** to "Run Process" in the **Choose System Variable** dialog box.









A dropdown will appear in the **Choose System Variable** dialog box, from which you can choose the process to start.



Choose the process you'd like to start, then click the **OK** button to close the **Choose System Variable** dialog box and save your configuration changes.

Column Name	Column Data
Name	Name ...
Run a Process	Run Process [01. User] ...

Now, when the Knowledge View executes, the results will display a "Run" column that shows an icon on each row that, when clicked, will launch the specified process using the Knowledge View result from that row.

Name	Run a Process
Original Form 123	
Second Form 126	
Webinar 20210929 Original Form 130	
Webinar 20210929 Original Form 146	
Webinar 20210929 Second Form 133	
Webinar 20210929 Second Form 149	

Run Process Icons

## Scheduling a Knowledge View #

Knowledge Views can be used to run a process on each of the objects returned by the filter criteria. This can be used, for example, to archive or delete forms or documents that match a certain criteria. Or it can be used to schedule a document review for documents that meet certain criteria. The criteria can be any Knowledge View filter criteria including last updated time, creation date, or any form field data. The actual process run on each object can perform any action or task.

Knowledge Views can also be used to export results into Excel, CSV, or PDF format. By scheduling this type of Knowledge View, a report will automatically be generated and can be written to a specific path on the server. This folder, for example, can be monitored by the BP Logix import utility, to automatically run a process on the reports. This process can, for example, automatically distribute the report to a pre-defined group of users.

These Knowledge Views can be run manually, or scheduled to perform an automatic actions. To schedule these Knowledge Views, a web page is run.

### Manual Knowledge View

Knowledge Views can be viewed or exported via URL from any web browser, using the kv.aspx page contained in Process Director. Below is a sample URL to the kv.aspx page:

```
https://-  
loc-  
alhost/k-  
v.aspx?id=KVID&dorun=1&bpUserID=USERID&bpPassword=PASSWORD&exportname=EXPORTNAME
```

The kv.aspx page can be used to:

- View a Knowledge View.
- Export a Knowledge View to a CSV or Excel file on your local machine's file system.
- Export a Knowledge View to a CSV or Excel file in the [Content List](#) of the Process Director server.

The kv.aspx command can take the following parameters as part of the QueryString in the URL:

URL PARAMETER	DESCRIPTION
id	The KVID of the Knowledge View to run.
dorun	This optional parameter is used only when the Knowledge View runs a Workflow or Process Timeline. Specify "1" to run the command immediately without a prompt.
exportname	This parameter is required only when exporting the Knowledge View to a CSV or Excel file, or to the <a href="#">Content List</a> . When exporting to the file system on a local machine, this parameter takes the desired local file path and file name of the exported file, e.g., "C:\filename.csv". When exporting to the Content List in Process Director, this parameter takes only the desired file name, e.g., "filename.csv".
BPUserID	This optional parameter is used only when the Knowledge View runs a Workflow or Process Timeline. The parameter takes a Process Director user ID with permission to run the Knowledge View.
bpPassword	This optional parameter is used only when the Knowledge View runs a Workflow or Process Timeline. The parameter takes the Process Director password for the user passed in the <i>userid</i> parameter.
contentfolderpath	This is an optional parameter used only when exporting the Knowledge View to a CSV or Excel file in the <a href="#">Content List</a> . The parameter takes the <a href="#">Content List</a> folder path to the Content List folder where the file will be stored, e.g., "/folder1/folder2". The <i>contentparentid</i> parameter can be used in lieu of this parameter if the Folder ID of the destination folder is known.
contentparentid	This is an optional parameter used only when exporting the Knowledge View to a CSV or Excel file in the <a href="#">Content List</a> . The parameter takes the Folder ID of the desired destination folder. If the Folder ID isn't known, the <i>contentfolderpath</i> parameter can be used, instead.

For example, the URL to run a Knowledge View which runs a process on each result would be:

```
https://  
ServerName.com/kv.aspx?id=KVID&dorun=1&bpUserID=USERID&bpPassword=PASSWORD
```

The URL to run a Knowledge View which writes the results into an XLS file (the KView must be defined to Export the results to Excel):

```
https://ServerName.com/kv.aspx?id=KVID&bpUserID=USERID&bpPassword=PASSWORD&exportname=c:\test.xls
```

The URL to run a Knowledge View which writes the results into a PDF file (the KView must be defined to Export the results to Excel):

```
https://ServerName.com/kv.aspx?id=KVID&bpUserID=USERID&bpPassword=PASSWORD&exportname=c:\test.pdf
```

The URL to run a Knowledge View which writes the results into a CSV file (the KView must be defined to Export the results to CSV):

```
https://ServerName.com/kv.aspx?id=KVID&bpUserID=USERID&bpPassword=PASSWORD&exportname=c:\test.csv
```



When exporting to a Windows file location, you can't export files via the `exportname` URL parameter to any folder that isn't specifically listed in the `AllowedExportLocations` custom variable. Attempts to export to a non-allowed location will fail.

## Scheduled Knowledge View

You may want to automatically schedule the Knowledge View to run at regular intervals (for example, every night at midnight). To do this, use the Microsoft Windows Scheduled Tasks utility. This utility enables you to schedule and test commands executed on a regular basis.

Do not schedule IEXPLORE.EXE because the web browser will never close. Rather, use the `bputil.exe` command to run the web page. For example, enter this command in the "Run" dialog box to schedule the synchronization:

```
"PATH\bputil.exe" SU  
"https://  
ServerName.com/kv.aspx?id=KVID&dorun=1&bpUserID=USERID&bpPassword=PASSWORD"
```

Where `PATH` is the installation directory for Process Director (e.g. `c:\Program Files\BP Logix\Process Director\`). Enter the appropriate credentials in the Windows Scheduler when prompted



Each of the above methods use the `bpUserID` and `bpPassword` parameters. Be advised that using the `bpUserID` or `bpPassword` parameter requires sending the User ID and Password in clear text, so be mindful of the security implications of transmitting these values.

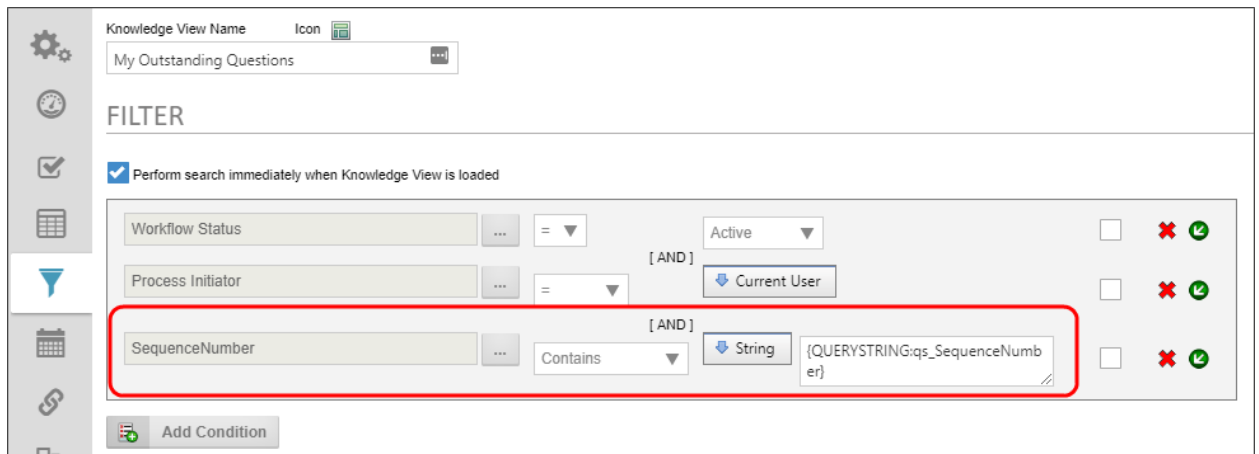
## Knowledge View Parameters

Knowledge Views can be [parametrized](#); which means configured to return only data that matches a filter value that is passed to the Knowledge View from an external object, like a form or from a URL Parameter.

Indeed, we saw an example of the latter in the [Knowledge View Operations](#) topic, in the section titled **Passing Filter Values in a URL**. In this topic, we will describe the required components for parametrizing a Knowledge View, and how to use a Business Value to call a parametrized Knowledge View, and only return the information that matches the filter we'd like to specify.

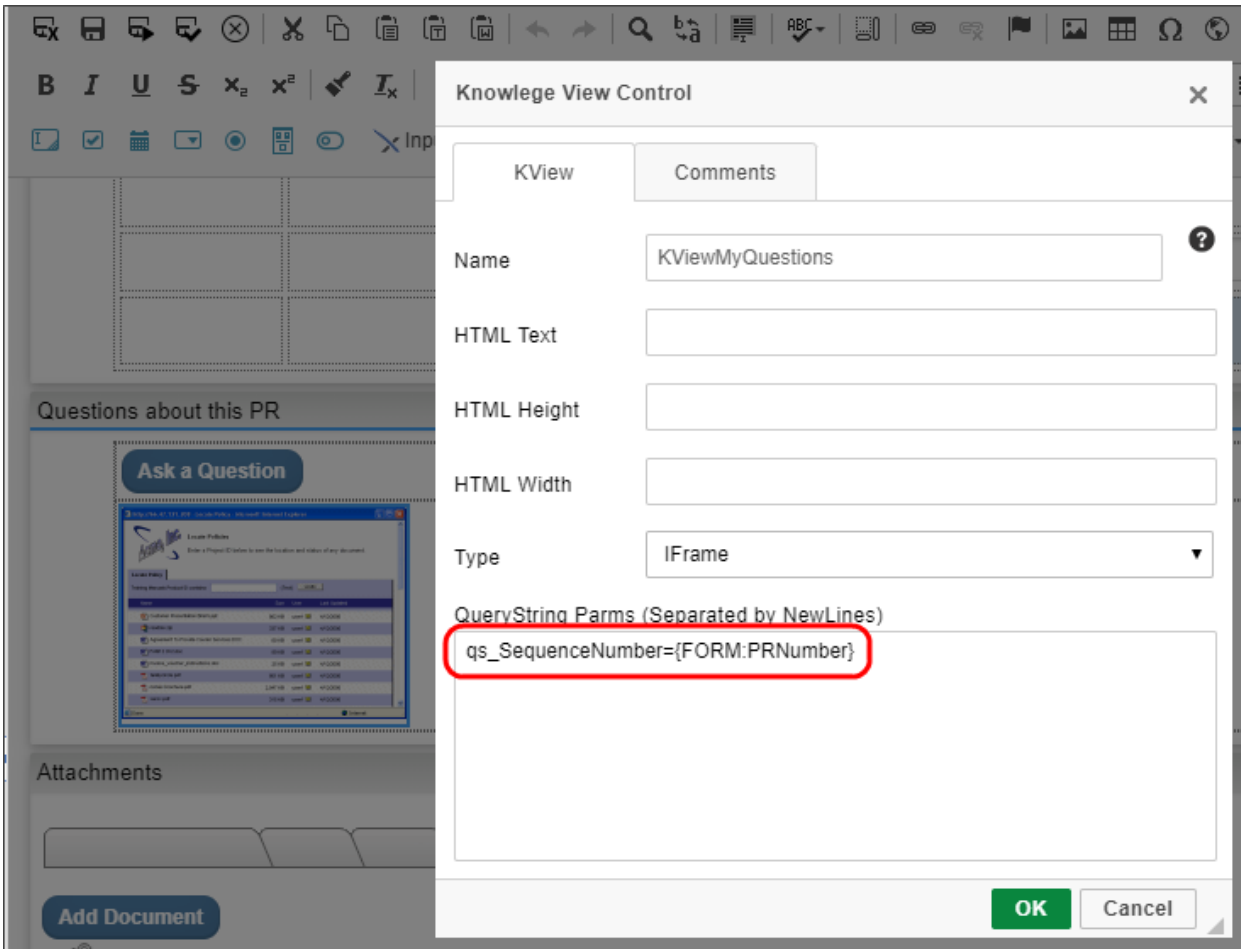
### Simple Parametrization #

The most common use case for implementing Knowledge View parameters via a Form is through the use of the Knowledge View control. To implement this use case, your Knowledge View must have a parameter filter. For example, we might need to pass a unique number to the Knowledge View to return items that match a specific form or request number. So, we need to pass that unique number through a query string variable. To do so, we'll configure the query string variable in the filter tab of the Knowledge View, to create a query string named "qs\_SequenceNumber".



In the form, once we place the Knowledge View Control on the form, we have to configure the query string in the Knowledge View control's **Properties** dialog box.

In the form, there is a form field called PRNumber that stores a unique request number. We'll pass this value to the Knowledge View through the "qs\_SequenceNumber" query string by assigning the value of PRNumber form field, using a form field system variable.



When the form runs, the Knowledge View Control will display only records that are related to the specific request that was created by that form instance.

### Parametrization with a Business Value #

For the purposes of this discussion, we have a simple Knowledge View that returns Form Instances containing some sales data.



Form Data - Filtered (400 items)

Name	Customer	Order Date ▲	Product	Quantity
Order #45 for Mollari LLC Submitted on 05/04/2017	Mollari LLC	5/4/2017	Widget	48
Order #86 for Computorama Submitted on 05/05/2017	Computorama	5/5/2017	Contraption	4
Order #206 for Chimera Submitted on 05/06/2017	Chimera	5/6/2017	Gadget	18
Order #183 for Clothiers Inc Submitted on 05/08/2017	Clothiers Inc	5/8/2017	Gizmo	17
Order #226 for Gilligan's Island Travel Submitted on 05/08/2017	Gilligan's Island Travel	5/8/2017	Implement	11
Order #342 for Cantata Inc Submitted on 05/10/2017	Cantata Inc	5/10/2017	Utensil	7
Order #44 for SuperSoft Submitted on 05/11/2017	SuperSoft	5/11/2017	Machine	7
Order #205 for Gilligan's Island Travel Submitted on 05/12/2017	Gilligan's Island Travel	5/12/2017	Implement	42
Order #299 for SuperSoft Submitted on 05/15/2017	SuperSoft	5/15/2017	Device	8
Order #259 for Pear Computer Submitted on 05/17/2017	Pear Computer	5/17/2017	Implement	33
Order #164 for SuperSoft Submitted on 05/20/2017	SuperSoft	5/20/2017	Device	34
Order #225 for The A Team Consultancy Submitted on 05/22/2017	The A Team Consultancy	5/22/2017	Device	44
Order #236 for Custodes Partners Submitted on 05/23/2017	Custodes Partners	5/23/2017	Implement	21
Order #163 for ID10T Submitted on 05/24/2017	ID10T	5/24/2017	Utensil	48
Order #21 for MuchMusic Submitted on 05/24/2017	MuchMusic	5/24/2017	Implement	19

We want to build a search that uses this Knowledge View to return only records for specified customers, so, we need to parametrize the Knowledge View to accept the customer name from an external source, like a form.

The first step to parametrizing a Knowledge View is to create the filter parameter you'd like to apply. In the **Filter** tab of the Knowledge View definition, you need to create a filter that uses a variable to filter the data.

One of the Form Fields in the set of Form Instances we are querying is called CustomerName, and this is the form field upon which we wish to filter.

On the left side of the filter condition, add the **CustomerName** form field, then set the Operator to "Contains". In cases like this, it's a good practice to use the "Contains" operator so users can a) perform a search using just a part of the customer name, without having to worry about matching the whole name exactly, and b) to return ALL of the records in the Knowledge View if the filter value is blank.

On the right side of the condition, set the value to a "Variable". When you do so, a text box will appear that will enable you to enter the variable name you want to create. This name is an arbitrary value, so we are going to name the variable, "Cust". Now that we've set up the variable parameter, we can save and close the Knowledge View.

With a Knowledge View that will accept a parameter, we can access filtered data by passing a value to the Knowledge View from another object, using a Query String. A **Query String** is a simple text phrase that passes the parameter value to the Knowledge View, and it's written with a specific syntax:

`VariableName=Value`

In the case of our example above, we might have a customer named "Pear Computer". If we wanted to return only sales orders from Pear Computer, therefore, our query string might be:

`Cust=Pear Computer`

Of course, in most cases, we won't be able—or want—to hard-code a customer name. Instead, we'll need to pass the customer name via some sort of system variable, based on the value of, say, a form field. So, to do that, let's continue with this sample Knowledge View, and build a more complicated implementation based on it.

Let's say that we'd like to create a search form that enables us to enter the customer's name—or, since we're using the Contains operator, the relevant part of a customer's name—and return all the orders from that customer. To build that form, we'll need an input box to serve as the search box into users will enter the name, and an array that contains the rows and fields that the Knowledge View return. Of course, to populate that Array, we'll also need a Business Value that we can bind to the Array to fill out the data.

First, let's build the Business Value. We need to create a Business Value that uses our Knowledge View as a data source.

Business Value Icon

Parameterized Form Data BV

## CONFIGURE

SQL Data Source    **Knowledge View Data Source**    REST Data Source

Choose Knowledge View  
Form Data - Filtered ...

Knowledge View Filter Data  
Cust={PARAM:Customer}

Execute Knowledge View under this user context  
Barb Stanley x

Test Knowledge View

**Properties (5 Properties)**

Property Name	Property Type	Property Value
Name	Knowledge View Column	Name    Column Data
Customer	Knowledge View Column	Customer    Column Data
Order Date	Knowledge View Column	Order Date    Column Data
Product	Knowledge View Column	Product    Column Data
Quantity	Knowledge View Column	Quantity    Column Data

Add Property

**Parameters (1 Parameter)**

Name	Test Value
Customer	

Add Parameter

Notice two things about this sample Business Value. First, in the **Parameters** section of the Business Value, There is a Parameter named "Customer". This is the parameter we will use to pass the customer name to the Knowledge View. Second, in the **Knowledge View Data Source** tab, we have specified the Query String to use in the **Knowledge View Filter Data** property as:

`Cust={PARAM:Customer}`

This Query String will apply the value of the **Customer** parameter of the Business Value to the **Cust** variable we created in the Knowledge View.

Finally, of course, we need to create the Business Value Properties that will store the Knowledge View Columns. Once we've done that, our Business Value is complete, so we can save and close it, and create our form, which is pretty simple.

The screenshot shows a web form with a rich text toolbar at the top. Below the toolbar is a text input field with the placeholder text "Enter part of Customer Name:". Below this is a table with four columns: "Customer", "Order Date", "Product", and "Quantity". Each column contains an empty text input field. The table is enclosed in square brackets.

At the top of our form, we have a text box named **CustFilter**, where we will enter the customer name. Below that, we have the array that will display the rows and fields that will be returned by our Business Value.

In the **Form Controls** Tab of the Form definition, we will make the **CustFilter** field an Event field, so that the Business Value gets called and changes the data in the Array each time we change the value of the **CustFilter** field.

The screenshot shows the "FORM CONTROLS" configuration window for the "CustFilter" field. The window has a "Control Name" list on the left and a "Form Field Properties" panel on the right. The "CustFilter" control is selected in the list. The properties panel includes:

- Event Field
- Synchronize Field Within Process
- Encrypt
- ToolTip: [Empty text box]
- Friendly Name: [Empty text box]
- Data Type: Text (dropdown), Min: [Empty text box], Max: [Empty text box]
- Default Value: None (dropdown)
- Link to Attribute: [Empty text box] with a "..." button
- Case Property: None (dropdown)
- Set Readonly Options: [Dropdown menu]
- Set Display Options: [Dropdown menu]
- Set Required Options: [Dropdown menu]
- Set Style Options: [Dropdown menu]

At the bottom of the panel are "OK" and "Cancel" buttons.

As an aside, all of the array's Input controls are disabled from the **Form Controls** tab as well. Also, the form is set to **Show Disabled Fields as Text** since we don't want to do any data entry on this form, so we don't need to show the actual Input controls inside the array, just the data they'll contain.

To bind the array to our Business Value, so that it displays the data we want to see, we need to go to the **Set Form Data** Tab of the Form definition, and do two things, using the **CustFilter** event field.

The screenshot shows the 'SET FORM DATA' configuration interface for a 'Sales Search Form'. It features a sidebar with navigation icons and a main configuration area. The 'CustFilter' event is selected, and its value is set to '0'. Below this, a 'Business Value: Parameterized Form Data BV' section is visible, which contains a table of fields to be populated from the Business Value. The table has two columns: 'Customer' and 'CustFilter'. The fields listed are CustomerName, OrderDate, Product, and Quantity, each with a dropdown menu and a red 'X' icon indicating a binding or action.

First, every time the **CustFilter** event fires, we want to empty the array. Otherwise, Process Director will keep appending rows to the array every time we change the **CustFilter** field. So, the first action that takes place when the **CustFilter** event fires is to set the value of the Array to a Number, and enter "0" as the number. The value of an array is the number of rows in the array, e.g., an array with a value of "2" will display two rows. By setting the value to 0, we are removing all of the rows in the array, so that the empty array can be refilled with fresh data.

The second thing we need to do is to set the values of the array fields to the properties contained in the Business Value. When we do so, the configuration screen will prompt us for the value we want to use as the **Customer** parameter. In this case, we want to use the **CustFilter** field to supply the parameter.

So, let's take a step back and look at the parametrized architecture we've created.

1. When we change the value of the **CustFilter** field in the form, the array will be emptied and the Business Value will be called.
2. When the Business Value is called, the form will pass the value of the **CustFilter** field to the **Customer** parameter in the Business Value.
3. The Business Value will pass the value of the **Customer** parameter, in turn, to the Knowledge View's **Cust** variable in the Knowledge View Filter.

4. The Knowledge View will create a dataset that is filtered to contain only records where the **CustomerName** matches the **Cust** variable's value.
5. The Business Value will fill the form array with the data returned by the Knowledge View.

Assuming that we've configured all of this correctly, we now have a working search form that uses a Business Value and parametrized Knowledge View to display our search data.

Enter part of Customer Name:

Customer	Order Date	Product	Quantity
Pear Computer	5/17/2017	Implement	33
Pear Computer	5/30/2017	Contraption	49
Pear Computer	6/17/2017	Widget	26
Pear Computer	9/11/2017	Gizmo	8
Pear Computer	9/20/2017	Gizmo	25
Pear Computer	10/31/2017	Apparatus	24
Pear Computer	1/25/2018	Contraption	38
Pear Computer	3/10/2018	Implement	50
Pear Computer	5/16/2018	Gizmo	46
Pear Computer	5/20/2018	Implement	40
Pear Computer	7/3/2018	Apparatus	45
Pear Computer	10/21/2018	Widget	31
Pear Computer	11/10/2018	Contraption	28
Pear Computer	12/9/2018	Machine	43
Pear Computer	2/14/2019	Machine	6
Pear Computer	3/20/2019	Machine	17
Pear Computer	4/28/2019	Utensil	28

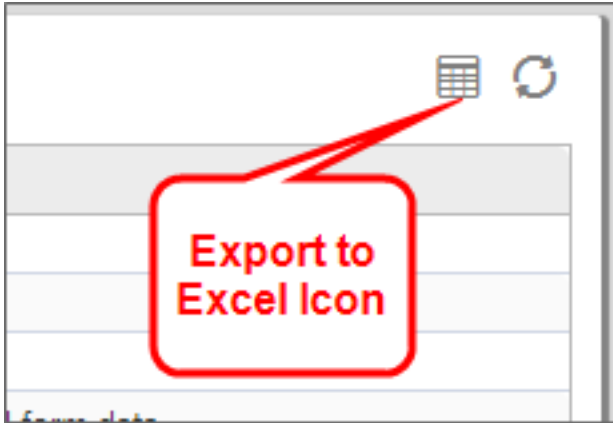
## Knowledge View Exporting and Reporting

The default behavior of a Knowledge View is to display the result list in the browser. For more detailed analysis, though, you may need to export the Knowledge View into a more accessible format, so, Knowledge View results can be exported to a CSV file (which can be opened by Excel). If you have a Cloud installation, Subscription license, or the older Tiered license with Office Integration component, you can export directly to Excel, and use Excel templates to modify or style the exported file.

When exporting to either CSV or Excel, exported Date/Time fields will contain both the date and the time.

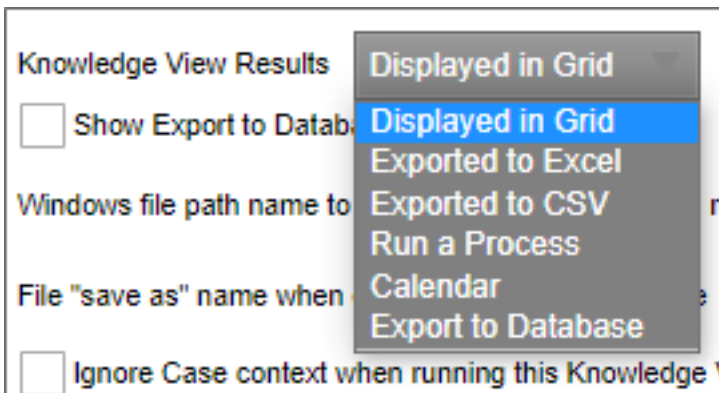
### Exporting the Results to a CSV File

To export the results to a CSV file select the **Show Export to CSV Icon** check box in the Knowledge View definition's **Configure** tab. Doing so will display an export icon at the upper right corner of the Knowledge View. This option is selected by default when a new Knowledge View is created.

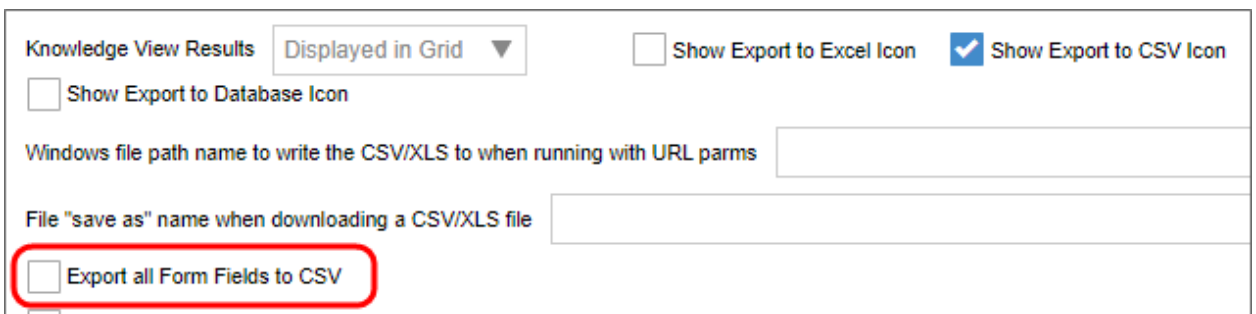


There are two options when exporting the results to a CSV file. First, you can set the **Knowledge View Results** property to "Displayed in Grid", which is the default option. Second, you can set the **Knowledge View Results** property to "Exported to CSV", which will automatically save the data to a file on your local computer.

If you want to prompt the user for additional filter data on this Knowledge View when running the report, choose the option "Displayed in Grid" to allow the user to click on the icon to export the results after the filter data is entered.



Additionally, selecting the **Export all Form Fields to CSV** option will export all form data, regardless of what is defined on the columns. When this option is selected, all form fields will be exported after the defined columns. This will also cause the default display options for the columns to export only (only the first column will appear on the screen).



## Running an Excel Report on the Results

The results of the Knowledge View can also be streamed to an Excel template and have a report run automatically. This will result in Excel opening and running a report based on the data just retrieved from the server.

### Configure the Knowledge View to Reference the Excel Template

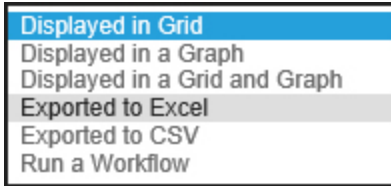
To enable the export of Knowledge View results to an Excel file, check the [Show Export to Excel Icon](#) checkbox in the [Configure](#) tab of the Knowledge View definition, then click the [Update](#) button to see more Excel options on the [Properties](#) tab of the definition. These additional properties won't appear on the [Properties](#) tab unless one of the export to Excel options is set on the [Configure](#) tab and the definition is updated.

The screenshot shows the 'Properties' tab of a Knowledge View configuration window. At the top, there is a 'Knowledge View Name' field containing 'Test' and an 'Icon' field with a search icon. Below this is a large 'PROPERTIES' section. The first field is 'Description' with a text area containing 'Enter a brief description of this Object'. The second field is 'Knowledge View Instance Name' with a text area containing '{KV\_DEF\_NAME} ({KV\_NUM\_ROWS} items)'. Below these are several rows of fields for associated forms and workflows, each with a three-dot menu icon to its right. The last two rows are 'Select Excel Template (.xls, .xlsx, .xlsm)' and 'Select Destination Folder for CSV/XLS (optional)', both of which are highlighted with a red rectangular box.

Before a Knowledge View can be exported to an Excel file, an Excel template file located in the [Content List](#) must be linked to the Knowledge View. To link to an Excel template file, set the [Select Excel Template](#) property.

Knowledge Views can immediately export their results to an Excel file when opened. To activate this option, select the “Export to Excel” option in the [Knowledge View Results](#) dropdown.





Please refer to the topic on [configuring Knowledge Views](#) to configure the use of the **Selected Destination Folder...** and **Export File Name...** properties to use when exporting the Excel file to the **Content List**.

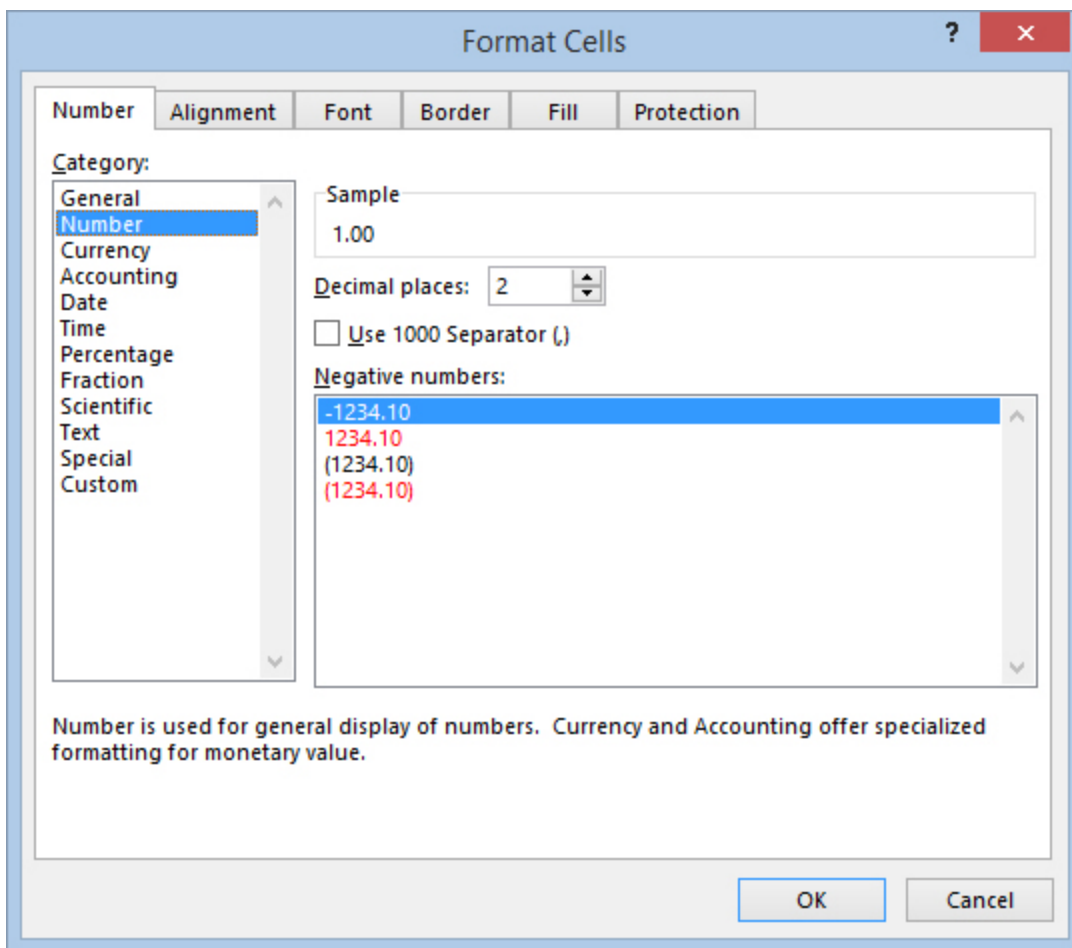
If the “Displayed in Grid” option is selected, then the Knowledge View results will display in the browser, along with an icon button giving the user the option to export the results to an Excel file.

## Excel Template File

The Excel Template Report file defines how the data in a Knowledge View will be displayed in an Excel report.

### Format the Columns in Excel

Apply formatting to the columns in the Template Excel file such as date/time and number formatting, using the Format Cells command in Excel.



## Smart Markers

In the template XLS document, you must place Smart Markers where you want the results of the Knowledge View to be imported. The smart markers take this format:

`&=BP.col1`

Where col1 is the name of the Knowledge View column. For example, the image below defines 3 columns in a template XLS sheet.

	A	B	C
1	<b>Contact Date</b>	<b>Customer name</b>	<b>Customer Type</b>
2	<code>&amp;=BP.Contact Date</code>	<code>&amp;=BP.Customer</code>	<code>&amp;=BP.Customer Type</code>

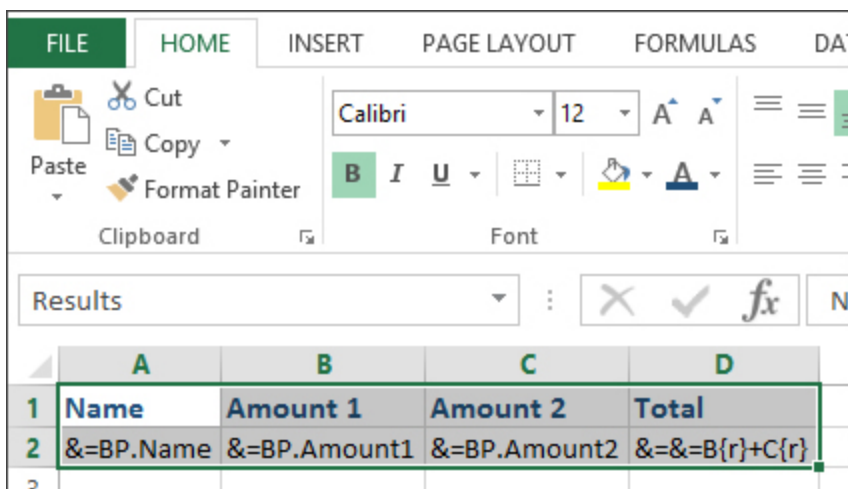
These columns will be filled with data from the corresponding columns (Contract Date, Customer, Customer Type) in the Knowledge View. The column names defined in the Smart Markers must match exactly the column names defined in the Knowledge View.

You can also place formulas in columns using the syntax described below, in the Excel Formulas section.

	A	B	C	D
1	<b>Name</b>	<b>Amount 1</b>	<b>Amount 2</b>	<b>Total</b>
2	<code>&amp;=BP.Name</code>	<code>&amp;=BP.Amount1</code>	<code>&amp;=BP.Amount2</code>	<code>&amp;=&amp;=B{r}+C{r}</code>

## Define an Excel Range

If you define a range in the Excel template that includes the header columns, and the template columns, the range will automatically be expanded as results are inserted into the spreadsheet.



## Define Excel Pivot Table / Formulas

The Excel template file can contain Pivot tables or Excel formulas that can be automatically run against the data. Create the Pivot Table against a RANGE name that encompasses the template and header column. When the Knowledge View is run, it will stream (export) the results into the RANGE specified. The number of rows in the RANGE will be expanded automatically to accommodate the number of records returned from the Knowledge View.

To automatically run a Pivot Table, ensure that the option is selected in Excel that indicates the Pivot Table Report should run when loaded (i.e. Refresh on open).

Also, when using a Pivot table, ensure that the Excel file you upload in to the [Content List](#) was saved with the Pivot Table Graph as the sheet that is displayed. When the Excel file is saved, it will open with the same sheet displayed.

### Special Excel Columns

You can include special columns in the Excel output range which will automatically be filled in when the report is delivered to the user. You don't need to include these columns in the Knowledge View Definition.

Variable Syntax for these special columns should be:

`&=$Var_Name`

	A	B
1	<b>Kview Name</b>	<b>Kview Description</b>
2	<code>&amp;=\$BP_KV_NAME</code>	<code>&amp;=\$BP_KV_DESC</code>

The following variable names are available for use:

COLUMN	DESCRIPTION
BP_KV_NAME	The Name of the Knowledge View Definition.
BP_KV_DESC	The Description of the Knowledge View Definition.
BP_ROW_SECOND	The value of the "second line" configured for this Knowledge View of the current row. This can be used like any other column name.
BP_FILTER_1	The value of the first filter criteria (if any) in the Knowledge View Definition
BP_FILTER_2	The value of the second filter criteria (if any) in the Knowledge View Definition, etc.
BP_KV_CURR_USER	The name of the current user
BP_KV_CURR_USER_EMAIL	The email of the current user
BP_KV_CURR_DATETIME	The datetime this excel file was generated

### Excel Formulas

Formulas must be formatted according to a specific syntax when create an excel report template.

The characters “&=&=” must precede any formula. The excel report template equation for “1 + 1” would then be as follows:

`&=&=1+1`

The character sequence “{r}” represents the current row number. To display the current row number in a cell, one would use the following formula:

`&=&={r}`

One can also access rows other than that which a cell belongs to. For example, the access the previous row, one could put the following formula into a cell:

`&=&={-1}`

To return the value of a specific cell, you'd use the format "Column\_Name{Row\_Number}". So, to display the value of column B in the current row, one would use the following formula:

`&=&=B{r}`

The string (numeric) after a formula will convert the value into a number. So, if you were to want to display this column as a number, you'd use this formula:

`&=BP.MyColumn(numeric)`

These components can be combined in any way to create a formula. For example, if I wanted to add one to the product of the previous row number and the value of column b in this row, I could use this formula:

`&=&={-1} * B{r} + 1`

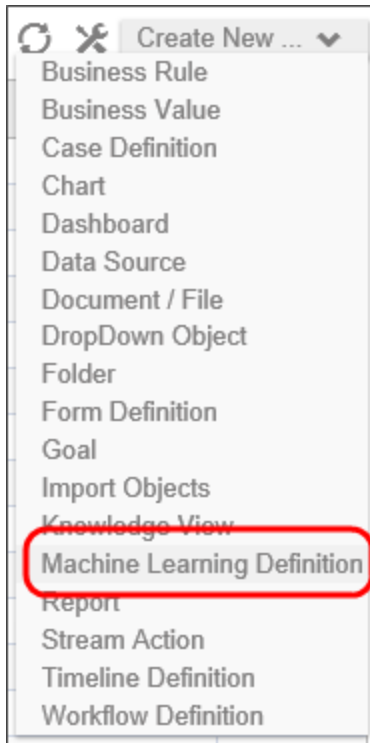
## Machine Learning (ML) Definition

Users of Process Director v5.0 and higher have access to the Machine Learning, or ML, definition object. The ML Definition enables you to use Process Director's Artificial Intelligence capabilities to review a dataset, and make predictions based on the state of that dataset.

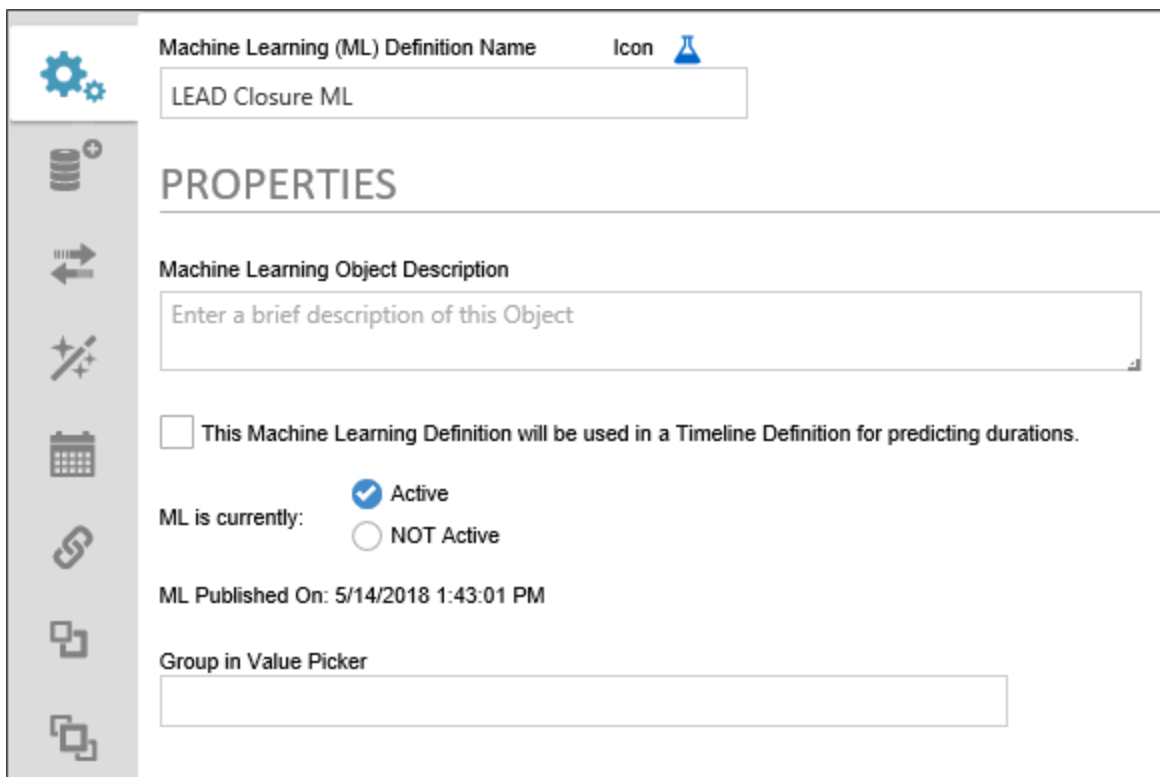
Process Director has long used Machine Learning/Artificial Intelligence (ML/AI) to analyze how Timelines work in the real world, and make predictions about when tasks will run in the current instance, based on the ML/AI analysis. For instance, this AI capability is how Process Director can predict when a task will be late. With the ML Definition, you can use the same capability to make predictions on any desired data, using a number of different statistical and analytic functions. The ML Definition object is globally available in Process Director, just like a Business Value, and can analyze data from both inside of and/or external to Process Director.

Keep in mind that this help topic isn't designed to teach you what statistical models are, or provide a lesson on how ML/AI works. It is, rather, intended to assist you in familiarizing yourself with the Process Director object itself. Just like with a SQL Business Value, where many users won't have the experience that enables them to construct SQL statements, the ML Definition assumes some basic familiarity with statistical functions, e.g., regression analysis, SVM, etc., to use effectively.

To create an ML Definition, simply select [Machine Learning Definition](#) from the [Create New...](#) dropdown menu located in the upper right corner of the [Content List](#).



## Properties Tab #



The Properties tab contains the basic configuration and publication options for the object.

## Icon

The Icon Property enables you to use the [Icon Chooser](#) to pick the Desired Icon for the object.

## Machine Learning (ML) Definition Name

The Name of the object you wish to display in the Content List.

## Machine Learning Object Description

A brief description of the ML Definition's purpose. This is mainly for administrative purposes, and any data entered here will appear on the second line of the [Content List](#) entry for this object.

## This Machine Learning Definition will be used in a Timeline Definition for predicting durations.

This property, when checked, tells Process Director that this ML object will be used to make time-based, predictive analyses for the completion of Timeline Activities.

## Machine Learning is currently Active/NOT Active

Selecting the **Active** radio button will expose the ML Definition to the dropdown menu used in the [Choose System Variable](#) dialog box. Setting the definition to **NOT Active** will deactivate the definition, and it won't be available for use in process Director until it is set to **Active**.

## Group in Value Picker

A text field that sets the menu category in which the ML Definition will appear in the in the [Choose System Variable](#) dialog box.

## Data Set Tab #

The **Data Set** tab enables you to choose the dataset that will be used for the ML Analysis. You can select any of the following data sources, and each selected data source will change the user interface to reflect the type of dataset you choose.

## SQL Data Source

The screenshot shows the configuration interface for a Machine Learning (ML) Definition. At the top, there is a section for the ML Definition Name and Icon, with the name set to "LEAD Closure ML". Below this is the "DATA SET" tab, which is currently selected. The "Data Set Type" is set to "SQL Data Source". Underneath, there is a "Please Choose Data Source" section with a text input field, a "Test Connection" button, and a "Test Query" button. The "SQL Command" section contains a large text area for entering the SQL query. To the right of the text area, there is a "[Select Table to Inspect Columns]" dropdown menu and a "Hide Columns" button.

The SQL Data Source can extract data from any accessible SQL-based data source supported by Process Director.

**Data Set Type:** You can use any existing SQL Datasource object to access the data, then write the appropriate SQL Command to extract only the data you'd like to use for your analysis. Simply select the Data-source object that connects to your database, using the **Object Picker**. You can click the **Test Connection** button to ensure you are connecting to the database specified in the Datasource object.

**SQL Command:** Write the desired SQL Command into the Text Box. Once you have written your SQL query, you can use the **Test Query** button to ensure that the SQL Command returns data.

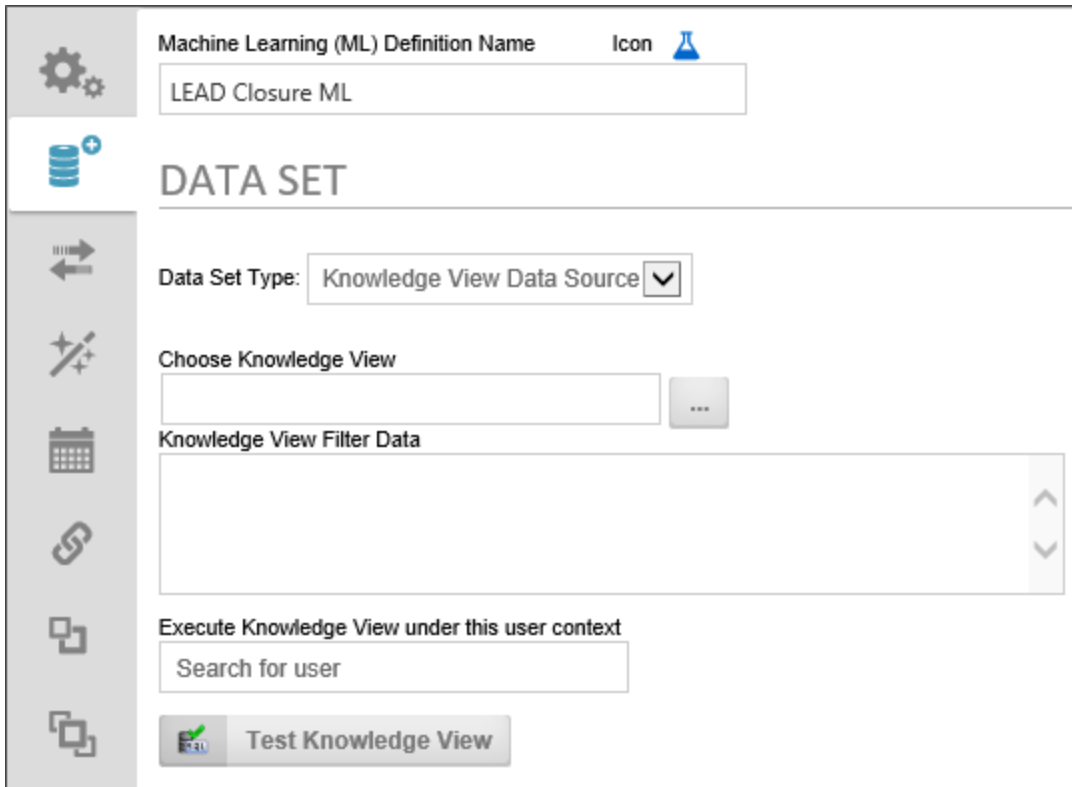
**Select Table to Inspect Columns:** To assist you, the **Select Table to Inspect Columns** dropdown will automatically be filled with all of the table names of the tables that are accessible from the Datasource object you have selected. When you select a table name from the dropdown, a list of the fields in the selected will be displayed.


## Form Data Source

The Form Data Source enables you to use the existing instances of any Form Definition to use for the ML analysis. Using the **Select the Form Definition to be used for this ML data set Object Picker**, select the form definition that contains the instances you wish to use. Once you do so, a list of form fields from that form definition will appear. Using the check boxes adjacent to each field, you can choose the specific form fields you wish to include in your ML analysis. Additionally, you can choose all form fields by clicking the **Select All** button, or no form fields by clicking the **Select None** button.

In most cases, you probably won't want all of the form fields included in your analysis. For instance, many forms have common fields like names or telephone numbers that probably don't contribute much to an ML analysis. Conversely, unchecking all the form fields leaves you with nothing to analyze. You'll need to select only the form fields that have relevance to your analysis.


## Knowledge View Data Source




Machine Learning (ML) Definition Name      Icon 

LEAD Closure ML

**DATA SET**

Data Set Type: Knowledge View Data Source 


Choose Knowledge View



Knowledge View Filter Data

Execute Knowledge View under this user context

Search for user

 Test Knowledge View

Any existing Knowledge View can be used as a data source for your ML Analysis.

**Choose Knowledge View:** Select the Knowledge View you wish to use by choosing it from the [Object Picker](#).

**Knowledge View Filter Data:** If your Knowledge View uses any filter variables, you can enter the appropriate filter statements in this text box, using a new line for each desired filter statement.

**Execute Knowledge View under this user context:** Using the [User Picker](#), select a user that has run and view permission for the Knowledge View. Since ML Definitions, Like Business Values, are global, the user who runs the ML definition may not have permissions to access the underlying Knowledge View. Selecting a user with the appropriate permission to run the Knowledge View will run it in that user's context. You must also provide a [Password](#) for the user, once selected.

## REST Data Source



The screenshot shows the configuration interface for a Machine Learning (ML) Definition. The 'Machine Learning (ML) Definition Name' is 'LEAD Closure ML'. The 'DATA SET' section is active, showing 'Data Set Type' as 'REST Data Source'. The 'REST URL' field is empty. A 'Test REST Call' button is visible at the bottom.

Any accessible REST web service can be used to provide analysis data. Simply enter the URL for the REST web service, along with any required URL parameters, into the **REST URL** text box. You may use system variables as URL Parameters.

### Azure IoT Hub

The screenshot shows the configuration interface for a Machine Learning (ML) Definition. The 'Machine Learning (ML) Definition Name' is 'LEAD Closure ML'. The 'DATA SET' section is active, showing 'Data Set Type' as 'Azure IoT Hub'. The 'Please Choose Data Source' field is empty, with an object picker button (three dots) to its right.

Users of Microsoft Azure's IoT Hub service can access their IoT Hub directly through Process Director. Use the **Please Choose Data Source Object Picker** to select the Datasource object that connects to your hub.

### Transformation Tab #

Once your dataset has been selected from the **Data Set** tab, you may find it necessary to apply some changes to your data, or to ignore part of the data that you think isn't relevant to the decision or prediction that you'd like the ML Definition to make. This process of altering or ignoring some data in the dataset is called **transformation**, and conducting those transformations is the purpose of the **Transformation** tab.

Machine Learning (ML) Definition Name      Icon

LEAD Closure ML

## TRANSFORMATION

Condition      Transformation Type

e.g. (Status='C' Or Emp>100) and Start<'4/6/2018'      [Select Action]

Add Transformation      Show Transformed Data Set

To add a Transformation, click the **Add Transformation** button, which will make a transformation row appear on the page. Click the button again to add additional rows.

The transformation row initially has two properties to set:

### Condition

The **Condition** enables you to identify a specific data criterion to look for to determine whether the data in that row should be transformed. The **Condition** is much like the "WHERE" clause of a SQL statement, and, in fact, uses a SQL-like syntax to determine what data row to look for to apply the transformation. The **Condition** is specified by using two values, separated by some kind of operator in the middle. The Left-side value is the data field you'd like to look for, and the right side of the **Condition** is the value you are trying to match in that field, e.g.:

`FieldName='FieldValue'`

You can use all of the expected operators such as `<`, `>`, `=`, etc. when creating your **Condition**. Text or date values need to be identified with single quotes, while number values need no identifiers. For instance, let's say you want to look for the data record that has the Primary Key ID of 10. Your **Condition** could be written as:

`id = 10`

Conversely, if you were looking for data from a specific date, your **Condition** might be written as:

`StartDate = '5/15/2018'`

You can also use the AND and OR keywords to create a complex **Condition** that uses multiple search criteria, such as

`StartDate = '5/15/2018' AND Category = 'C'`

### Transformation Type

There are two options for conducting the transformation. The first is **Ignore Row**. If you choose this option, the data will be removed from the dataset. The second option, however, is to **Set Column to Value** which enables you to actually change the existing data in some way. If you select this option, some new property selections will also appear.

Machine Learning (ML) Definition Name Icon

LEAD Closure ML

**TRANSFORMATION**

Condition Transformation Type

e.g. (Status='C' Or Emp> 100) and Start<'4/6/2018' Set Column to Value [Select Column] [Select Operation] ↑ ↓ ✖

+ Add Transformation Show Transformed Data Set

In the new column that is labeled, [Select Column], you can choose the column whose value you wish to change.

In the [Select Operation] column, you can choose the type of change you'd like to make. The following options are available:

- Median - Change the value to the Median of all values in this data column.
- Average - Change the value to the Average of all values in this data column.
- Minimum - Change the value to the smallest value of all values in this data column.
- Maximum - Change the value to the largest value of all values in this data column.
- Least Common - Change the value to the least common, i.e., the value with the fewest number of instances of all values in this data column.
- Most Common - Change the value to the most common, i.e., the value with the largest number of instances of all values in this data column.
- Text Value - A value you can specify. When you select this option, a text box appears to enable you to enter the desired value.
- Upper Case - Changes the case of all text to upper case.
- Lower Case - Changes the case of all text to lower case.

Once you have added all of the desired transformations, you can view the resulting data by clicking the **Show Transformed Data Set** button, to display a data window showing you the transformed data.

Machine Learning Object Datasource

Filter: e.g. Status='C' Or Emp>100 Filter

Select feature to analyze

Employees	Public Company	Lead Source	Closed	Custom Demo	Company	Sales Rep	Velocity	Deal Size	Inquiry Date	Country	Number Of Demos	Title	Department	Onsite Demo
30481	1	Partner	0	0	Ac Eleifend Vitae Foundation	Rick Rob	Medium	99076.22	12/23/2013	United States	1	Director	Quality Assurance	1
26940	1	Partner	0	0	Ac PC	Diana Stuart	Medium	77181.59	7/29/2013	United States	1	VP	Quality Assurance	0
3048	1	Outbound Phone	0	0	Est LLP	Barb Stanley	Medium	24937.71	1/3/2013	United States	1	C-Level	Customer Service	0
35285	1	Partner	0	0	Vitae Diam Institute	Diana Stuart	Medium	34987.95	5/26/2012	United States	1	Director	Quality Assurance	0
19348	1	Google	0	0	Tempus LLP	Diana Stuart	Medium	38540.23	3/13/2014	United States	1	Director	Legal Department	1
21615	1	Inbound Phone	0	0	Est Erat Corp.	Diana Stuart	Medium	88517.96	12/8/2012	United States	1	VP	Legal Department	0
18365	1	Google	0	0	Vel Nisi Corp.	Diana Stuart	Medium	49731.68	1/19/2014	United States	1	Director	Customer Relations	1
6880	1	Conference	0	0	Et Natus Et Limited	Diana Stuart	Medium	67957.81	10/31/2014	United States	2	VP	Tech Support	0
4689	1	Other	0	0	Malesuada Consulting	Diana Stuart	Medium	51424.01	4/21/2015	United States	1	VP	Tech Support	0
47063	1	Webinar	1	0	Euismod Mauris Eu Corp.	Diana Stuart	Medium	77766.66	4/8/2016	United States	2	C-Level	Legal Department	0

Page size: 10 600 items in 50 pages Close

## Training Tab #

Once you have selected and transformed your dataset, Process Director needs to train itself on the data to apply the type of analysis or prediction you want to apply. The **Training** tab is where you conduct this training.

Machine Learning (ML) Definition Name    Icon

LEAD Closure ML    Publish

### TRAINING

Prediction Feature

Predicted Column: Closed (number)

Predicted Column Type: <Select Optional Data Type>

Machine Learning Algorithm: MLA Regressor Logistic Regression

Validate Using: Training Error

Train    Test Predict    Suggest Input Features and Algorithm

Input Features, 501 current records

Feature	Dropdown	Chart	Statistics
<input type="checkbox"/> Closed (number)	Automatic		Unique values: 2    Median value: 1 Most common value: 0    Average value: .5 Least common value: 1    Max value: 1 Min value: 0
<input type="checkbox"/> Company (string)	Automatic		Unique values: 490    Most common value: Est LLP Median value: Sem Institute    Least common value: Ac Eleifend Vitae Foundation
<input type="checkbox"/> Country (string)	Automatic		Unique values: 5    Most common value: United States Median value: Canada    Least common value:
<input type="checkbox"/> CustomDemo (number)	Automatic		Unique values: 2    Median value: 1 Most common value: 0    Average value: .5 Least common value: 1    Max value: 1 Min value: 0

The **Training** tab is divided into two sections. The top section is where you configure the **Prediction Feature**. The purpose of ML/AI is to analyze data and make predictions based on that analysis, much like the Process Timeline, based on past instances of a Timeline definition, can predict whether a future Activity is likely to be late.

### Predicted Column

This property sets the data column or form field, depending on the data type you're using, that will store the value that will be set as a result of a prediction.

### Predicted Column Type

This property sets the desired data type of the column or form field to store the value that will be set as a result of a prediction. Your options are:

- Text
- Number
- Yes/No
- Date

### Machine Learning Algorithm

This is the mathematical or statistical method that Process Director will use to make the prediction. You have the following options:

- **No Machine Learning:** No Algorithm will be applied.
- **MLA Classifier Multinomial Logistic Regression:** Multinomial logistic regression using Newton's method as its optimization algorithm.

- **MLA Classifier Linear SVM:** A support vector machine implementing a L2-regularized L2-loss support vector regression (SVR) learning algorithm that operates in the primal form of the optimization problem.
- **MLA Classifier Naive Bayes:** A naive Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions.
- **MLA Classifier C45:** Uses the C4.5 algorithm to train a decision tree. Each branch in the tree attempts to split on the feature that maximizes the amount of information gain.
- **MLA Regressor Least Squares:** A simple linear regression is able to fit a line relating the input variables to the output variables in which the mean-squared-error between the line and the actual output points is minimum.
- **MLA Regressor Logistic Regression:** Logistic regression is used for prediction of the probability of occurrence of an event by fitting data to a logistic curve.
- **MLA Regressor Linear SVM:** A Support Vector Machine trained on a learning algorithm specifically crafted for linear machines only. It utilizes an L2-regularized, L1 or L2-loss coordinate descent learning algorithm for optimizing the dual form of learning.
- **MLA Regressor Fan Chen Lin SVM:** This class implements the same optimization method found in LibSVM. It can be used to solve quadratic programming problems where the quadratic matrix Q may be too large to fit in memory.

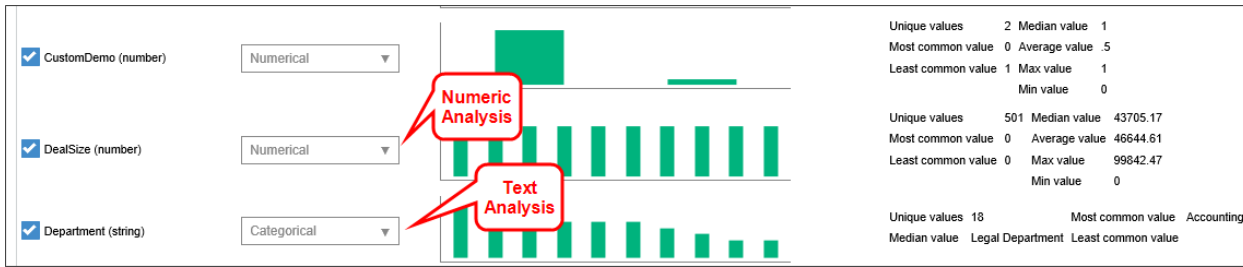
## Validate Using

This property specifies the data validation method to use for training on the dataset. The available methodologies are:

- **Training Error:** Derived by calculating the classification error of a model on the same data the model was trained on, producing a numerical estimate of the difference in predicted and original responses.
- **5-Fold Cross Validation:** The data is divided into 5 subsets, and the ML object repeats a holdout validation on each subset, using 1 subset as the test/validation set, and the other 4 sets collected into a training set.
- **10-Fold Cross Validation:** The data is divided into 10 subsets, and the ML Object repeats a holdout validation on each subset, using 1 subset as the test/validation set, and the other 9 sets collected into a training set.
- **Leave-One-Out Cross Validation:** One point of the training data is used as the validation set, while all of the remaining data points are used as the training set.

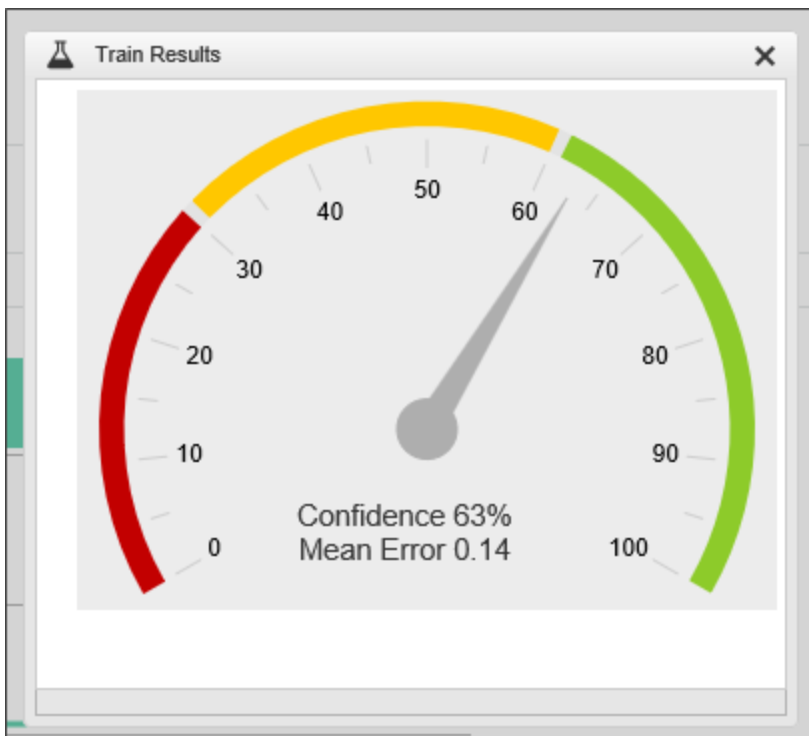
The **Input Features** section enables you to select the fields from your dataset that you'd like to analyze to create the prediction. Different fields will have different levels of effectiveness in the analysis. It may be difficult for you to know which fields will provide the best predictive result. You can do sample training on a field or collection of fields to enable Process Director to help you find the most effective fields to analyze by clicking the **Train** button.

For each available field, a graphical representation of the field's data is displayed. You can select a field to train on by checking the box next to the field, then for each selected field, choose the type of data analysis you wish to perform during the training. For numerical columns, you can perform Categorical, Numerical, or Exponential analyses, while, for text fields, you can conduct Categorical or "Bag of Words" analyses.



## Training the ML Definition

For this example, we have a set of form instances that contain data from a sales process. Along with data about the prospective customer and sales rep, we also have form data that tells us whether the sale closed, how many product demos were done, and other information. Based on the data from our existing sales form instances, we want to make a prediction about whether a sale is likely to close. By selecting a field or fields, then clicking the **Train** button, Process Director will analyze our data and give us some indication of how effective the selected data will be in a prediction about whether a sale will close.



As previously mentioned, not all fields are good candidates for prediction. Take a look at the result below, when we select and train for a prediction based on the title of the customer's point of contact:



This field doesn't give us much confidence in the predicted result, i.e., whether a sale will close. This uncertainty arises from;

- The fact that the title has too few common values. Every Customer rep may have a completely different title.
- The point of contact may not have any decision-making power. They may just be the person through whom communication passes.

So we need a better field or fields. Let's try some different fields. What about the customer department that is making the sales inquiry? Are some departments more historically likely than others to buy our product? Let's see:



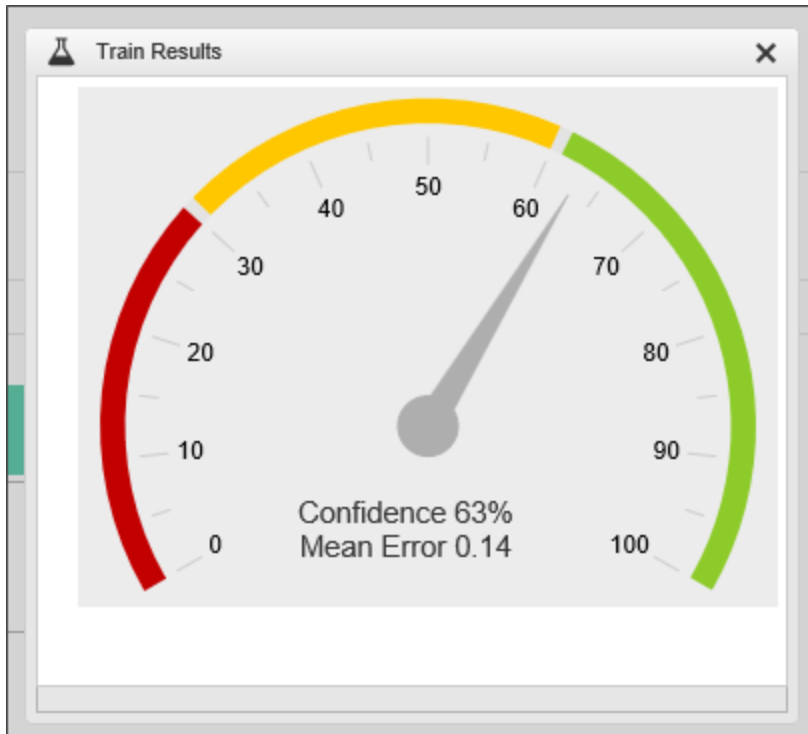
The confidence in the predictive value of this field is much better, but still not great. Let's add some more

<input checked="" type="checkbox"/> Department (string)	Categorical ▼
<input type="checkbox"/> Employees (number)	Automatic ▼
<input type="checkbox"/> InquiryDate (string)	Automatic ▼
<input checked="" type="checkbox"/> LeadSource (string)	Categorical ▼
<input checked="" type="checkbox"/> NumberOfDemos (number)	Numerical ▼

fields to the Department:

Now, that we've added the additional fields, we can train again to see how predictive our data looks now.





It looks like we've found a set of values that have some fairly good predictive powers. We can use these values to test our prediction, by clicking the **Test Predict** button to open a prediction test screen.

The 'Test Prediction Object' window contains five input fields with the following values: LeadSource: Google; SalesRep: Diana; Velocity: High; NumberOfDemos: 4; Department: Accounting. At the bottom right are 'OK' and 'Cancel' buttons. At the bottom left, the text reads 'Result: 1, probability: 99%'.

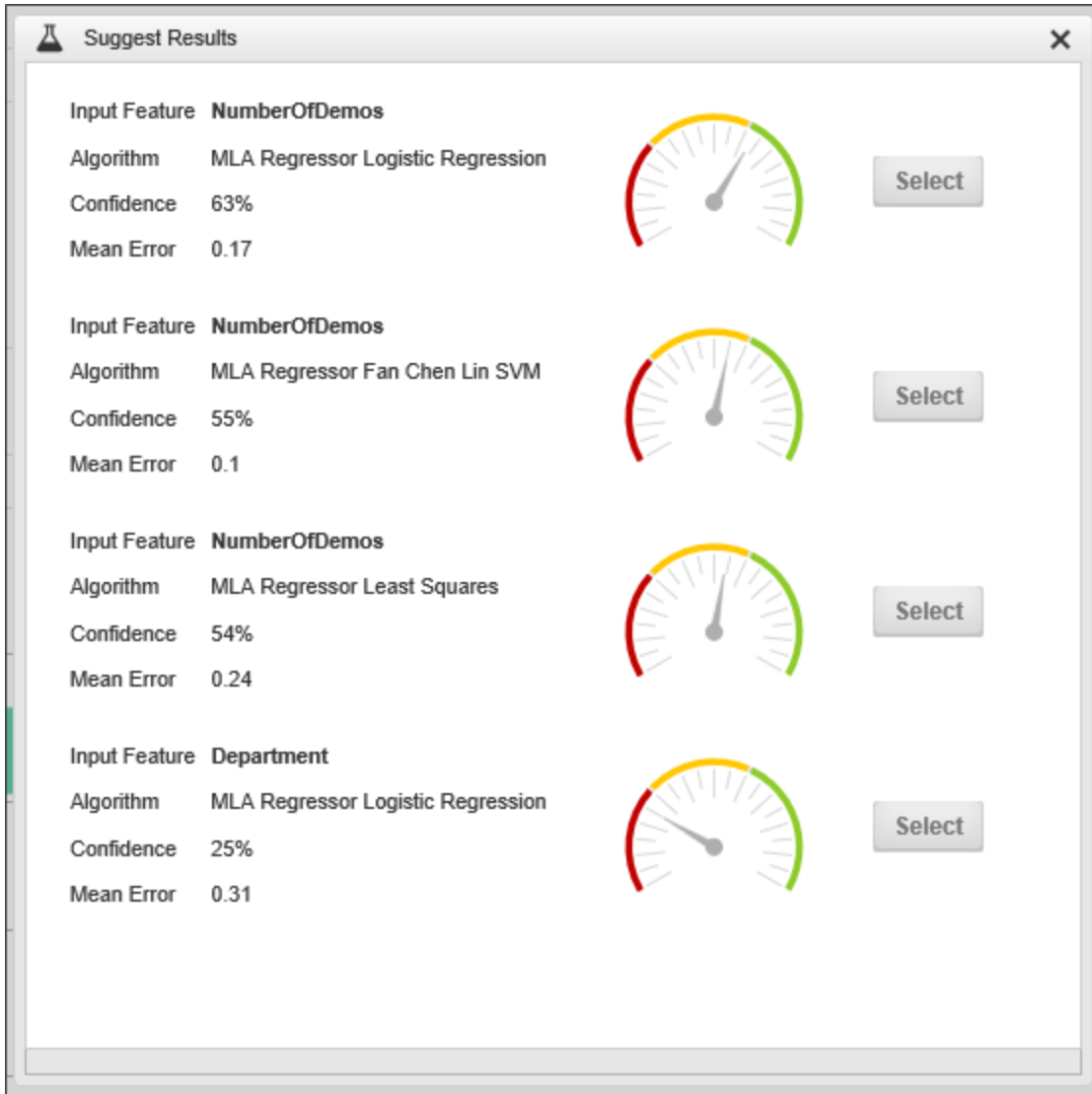
Using these test values, we can see that there appears to be a 99% probability that the sale will close (Result: 1).

We are now ready to Schedule and/or Publish our ML Definition.

### ML Object Suggestions

The ML Definition has an additional feature to help you analyze your data. When you click the button labeled **Suggest Input Features and Algorithm**, The ML object will analyze all of the data—and, be aware,

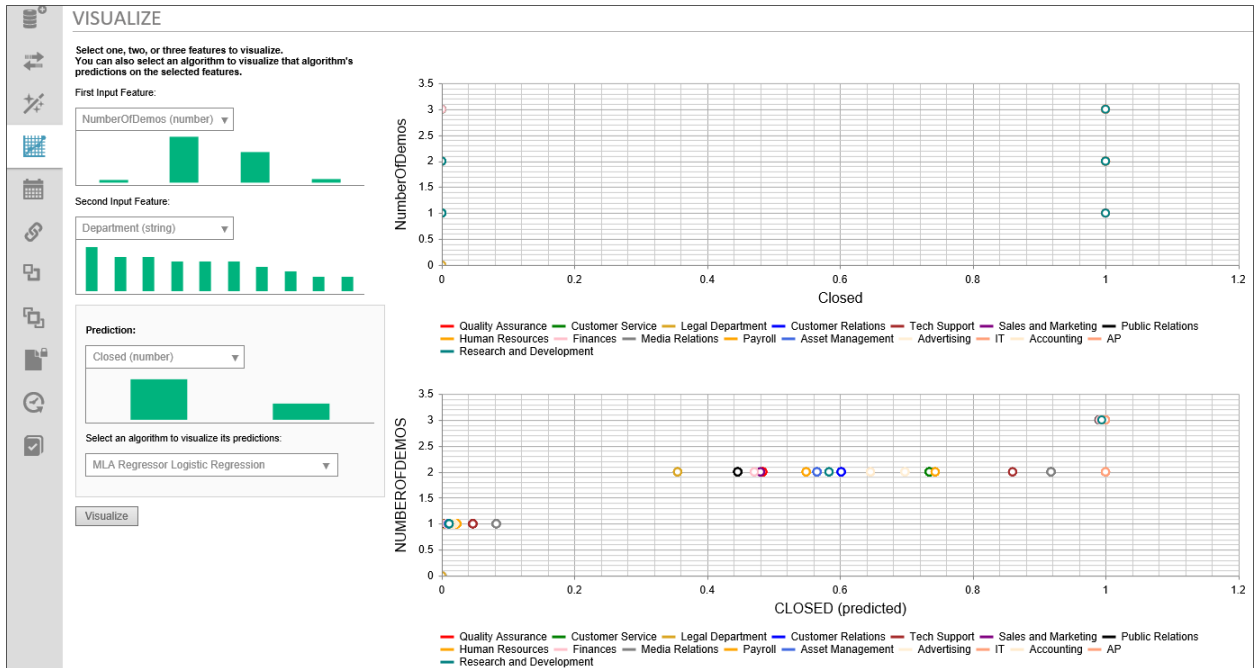
depending on how much data you have, this might take some time—and provide some suggestion about what data in your dataset might be the most predictive.



You can click the **Select** button next to the data column and regression method you'd like to use, and The ML Object will be updated with your selection.

## Visualize Tab #

The visualize table enables you to select from your data columns and your predicted column to visualize the data set in graphical form. Once you have selected your data, click the **Visualize** button to see the data representation.



## Schedule Tab #

You can manually publish your ML definition, using the current data, by selecting **Publish** from the actions menu in the upper right corner of the ML Definition. This will make the ML Definition available, but only the currently existing data will be used for all future analyses/predictions. In order to update the and retrain the ML definition on a continuing basis, so that new data is included in the ML Definition, we need to go to the **Schedule** tab to configure how often we want to retrain and republish the ML Definition.

Machine Learning (ML) Definition Name: LEAD Closure ML

Icon:

**SCHEDULE**

Train & Publish on a Schedule

Repeat Every: 1 Days

Repeat Interval Starts At:

Scheduler Period End:

Days of Lead Time: 0

Limit to Business Hours Only

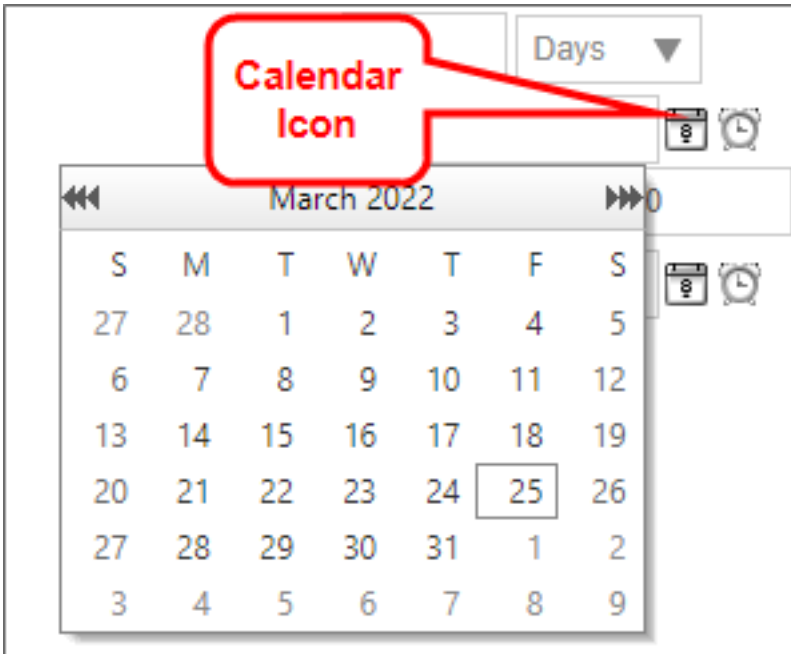
The first item to configure is to turn on scheduled training and publishing. Change the Dropdown value from **No Automatic Training & Publishing** to **Train & Publish on a Schedule**. When you do so, scheduling controls will appear that enable you to specify the training and publishing schedule.

## Repeat Every

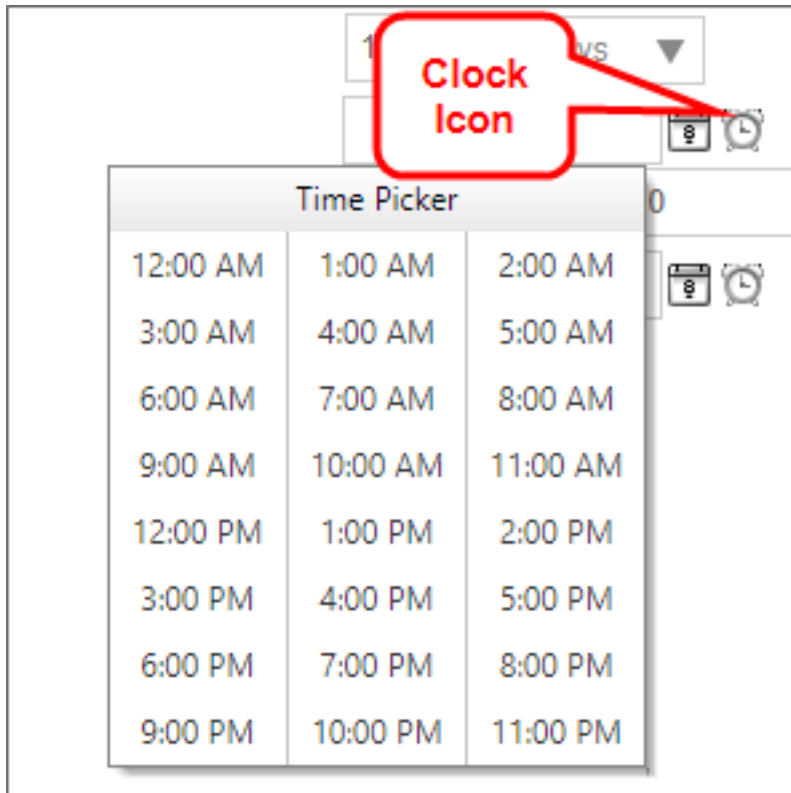
You can simply set the retraining to repeat every **N** days, weeks, months, hours, etc. No further configuration is required. Once you manually publish the first time, the desired repetitions will occur at the specified interval.

## Repeat Interval Starts At

The optional date and time to begin scheduling training and publishing. To select a date, click the Calendar icon located to the left of the text box control to open a calendar you can use to select the date.



Similarly, to select a time, click the Clock icon located to the left of the text box control to open a time selector you can use to select the time.



### Scheduler Period End

The optional date and time to end scheduling training and publishing. The Configuration method is the same as the [Repeat Interval Starts At](#) property.

### Days of Lead Time

Sometimes, you want to begin an action prior to the due date. For example, a month-end report may need to be submitted on the first day of each month, covering the activities of the prior month. If we assume this report takes a couple of days to compile and generate, we might want to have a lead time of 2 days. In this case, the publishing and training will be evaluated two days early, so you have adequate lead time to generate the report.

### Limit to Business Hours Only

Check the box to publish and train only during the Business Hours that have been configured in the [BusinessHourStart](#) and [BusinessHourStop](#) Custom Variables for your installation.

## Meta Data

**Knowledge Management** is defined as the collection of processes or information that governs the creation, dissemination, and utilization of knowledge. Meta Data is a key component of the Knowledge Management features of Process Director.

At the core of Process Director is a management system that contains documents, Forms, images, web pages—essentially, all of your digital content. This management system can be governed by the use of

Meta Data. **Meta Data** is loosely defined as “data about data.” Meta Data, for example, is used by libraries in order to classify data about books. If you wanted to know when the book was received by the library, any summaries of the book by critics, or the last time it was checked out, that’s a good description of Meta Data. Each book, of course, has different content, but the attributes of, say, when a book was checked out, is common to all books, and thus, is “book meta data”.

In Process Director, Meta Data can be applied to unrelated objects, so that they can share both a shared set of data and a system of organization that isn't governed by their location in the Content List, or the organizational section that created or uses them. Using Meta Data, objects can be organized and retrieved by any categorization system you desire. Using Meta Data categories and attributes provides a powerful, consistent way to categorize content across an entire partition.

## Power of Meta Data #

The ability to apply the same Meta Data to completely unrelated objects grants you a power to unify objects in a way that is otherwise unavailable. Normally, in a Knowledge View, your choice is to choose to return either:

- Instances from a single object definition—normally Form instances of a specified Form definition
- Instances in a specified location in the Partition's [Content List](#), or
- All instances in the Partition.

What Meta Data enables you to do is apply the same Meta Data to different objects, anywhere in the partition, then build a Knowledge View to search for and return all instances that share the same Meta Data categories and/or attributes.

In a large organization, for example, there might be an application that stores policy documents that apply to the entire organization, while other sections of the organization might have their own applications for storing policy documents that are specific to their operations. All of these applications might do similar things, but they’re otherwise unrelated, and without Meta Data, it’s difficult or impossible to create a Knowledge View that will return the documents from all of these separate applications. Applying Meta Data to these different applications enables you to build Knowledge Views that can collate and return all of the documents from all of the applications in a single place.

Meta Data, therefore, provides an organizational method that enables you to return data in ways that brings together otherwise unrelated object definitions, regardless of who manages them, or their location in the partition.

## Meta Data Elements #

Process Director provides two main Meta Data elements that can be applied to objects.

**Categories** are higher-level classifications that can be applied to any object. To refer to our HR example above, every HR form might have the Category "HR" or "Human Resources" applied, so that all Human Resources forms can be found easily by searching that Category. Meta Data categories can be organized hierarchically, meaning that every Meta Data Category can have one or more subordinate categories, which enable you to create a branching tree of Meta Data categories. When assigning a lower-level Category to an object, all the higher-level, or parent, categories in the tree are also assigned to the object.

An **Attribute** is a data element that can be applied to a Category to store additional Meta Data about objects in that Category. For example, we might have a Meta Data Category for policy documents named "Policies". Each policy document will be assigned to the **Policies** Category. But there's more information we might need about these policy documents. For instance, we might need to know the effective date or expiration date of the policy. So, we might want to create additional attributes for **Policies** called "Effective Date" and "Expiry Date". Obviously, these dates will be different for each policy document, but each Attribute will be assigned to every policy document for us to store these dates. Assigning an Attribute to an object automatically assigns the Category to which the Attribute belongs.

So, the Meta Data Category is **immutable**, which is to say that the Category value is the same for every object assigned to it. An Attribute, on the other hand, is **mutable**, which means that, while every object has the same Attribute, the **value** of that Attribute may be different for every object.

## Some Caveats

Meta Data can be very useful, but Meta Data categories and attributes must be actively managed, and not implemented in an ad hoc manner. Excessive use of Meta Data—or a poorly designed Meta Data schema—can add an excessive administrative burden and may require extensive effort to revise. Use of Meta Data should be planned and implemented with an eye firmly set on the structure of the category hierarchy, and to ensure the ease of maintaining and scaling it over time.

## How Meta Data is Applied #

There are three main methods you can use to apply Meta Data to categorize Process Director objects like Form or Process Timeline definitions. Implementers can assign data to each object individually, or to folders in the [Content List](#).

### Object Assignment

Categories can be assigned by any user who has access to the object definition of any Form or Process Timeline. The **Meta Data** tab of the object definition is where Meta Data is configured. As categories are assigned to an object, the associated category attributes are displayed with an input field allowing variable data to be entered.

The screenshot shows the Meta Data configuration interface for a form named "Leave Request". The interface includes a sidebar with navigation icons (gears, pencil, list, calendar, checkmark). The main content area is titled "META DATA" and contains the following elements:

- Form Name:** Leave Request
- Icon:** A small green tree icon.
- Buttons:** "Assign Meta Data", "Add Meta Data to All Child Items", and "Replace Meta Data for All Child Items".
- Categories:** [ Training ] [ Training.Forms ]
- Attributes:** Form Type: Leave

Meta Data applied to object definitions is always inherited by all child instances when they are created, as well as on any attachments or other objects that are created as part of the instance.

Form instances can, additionally, link Meta Data attributes to a Form field. In the **Form Controls** tab of a Form definition, the **Properties** dialog box for each Form field contains a **Link to Attribute** property that, when configured, will save the value of the Form field as a Meta Data **Attribute**, assigning both the **Attribute** and its parent Category to the Form instance.

## Folder Assignment

Categories and attributes can be set in the **Meta Data** tab of a folder's properties. This configuration will automatically categorize any new object that is created inside that folder. When categories are inherited from the parent folder settings, this inheritance won't replace or overwrite the existing categories configured for an object or subfolder. Instead, the folder Meta Data will be **added** to the existing category assignments that might be configured in the child objects. Additionally, if a child object already has the same category attributes as the parent folder, those attributes won't be overwritten by the parent folder's attributes.

## Instance Assignment

Any object instance, such as a Form Instance, can have Meta Data Applied in an *ad hoc* manner. The Online Form Designer contains both **Category Picker** and **Attribute Picker** controls that enable you to choose a different Category or Attribute for every Form instance. When using these controls, you must run the **Set Meta Data** Custom Task to apply any Meta Data changes you make.

## Meta Data Replication #

When using object and folder assignment, editing or adding Meta Data to the object/folder definition does not affect the Meta Data of existing objects. To apply Meta Data changes to existing child objects, there are two replication buttons on the **Meta Data** tab.

- **Add Meta Data to All Child Items** – This button will **add** the folders categories and attributes to all objects below this folder. It won't overwrite any existing categories or attributes already assigned to those objects.
- **Replace Meta Data for All Child Items** – This button will **remove** the categories and attributes for all objects beneath this folder and **replace** them with the object's current settings.

Both of these replication settings will only apply to children of the object for which the Meta Data has been configured.

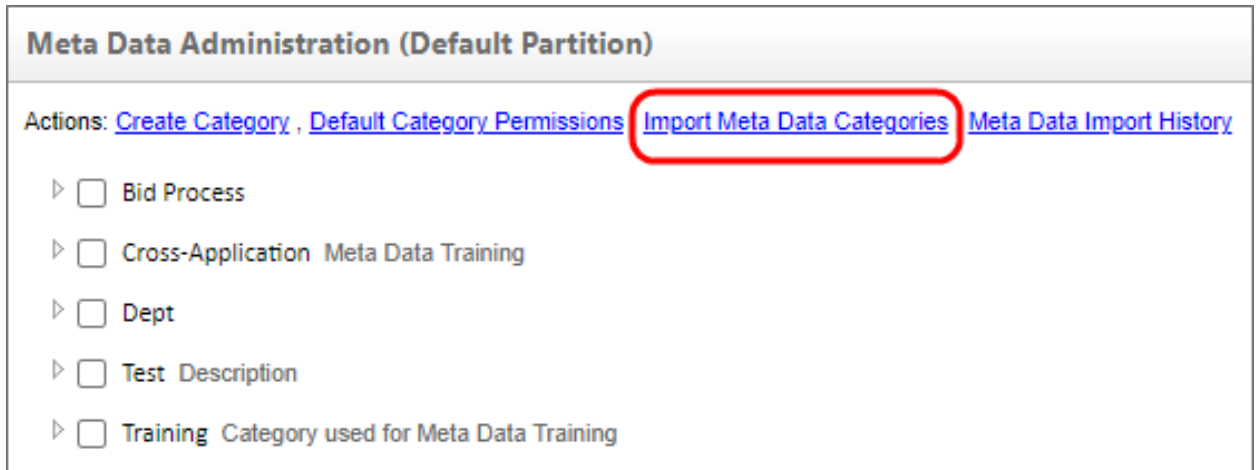
## Importing / Exporting Categories & Attributes #

For users of Process Director v4.05 and higher, Meta Data schemas can be exported and imported from the Meta Data Admin screen. This enables you to export the category and attribute definitions and import them onto another system as an XML file.

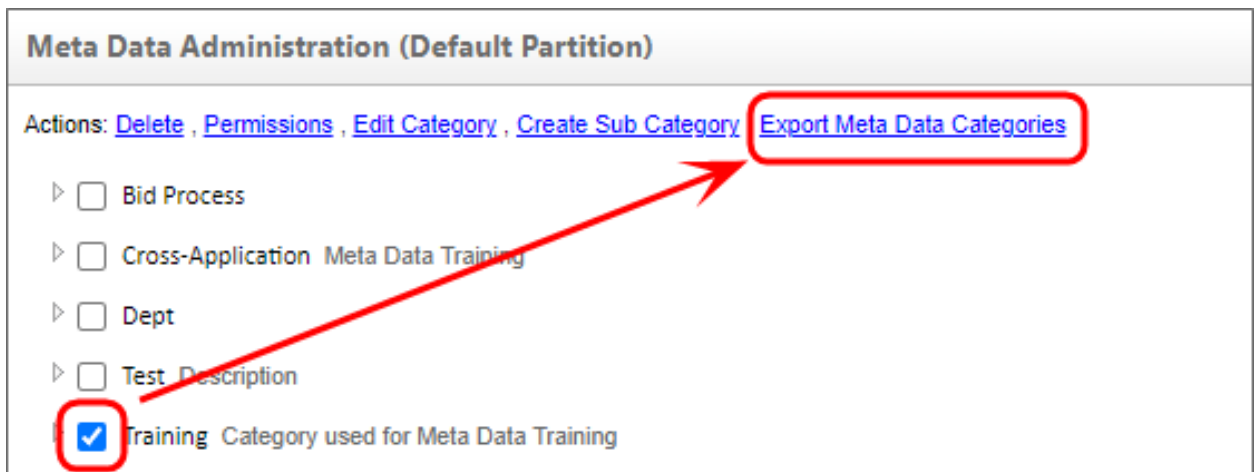
Category permissions are also enabled for v4.05 and higher, allowing control over who can modify the schema or use them for assignments. A partition admin has full permissions, but other users may not. The addition of permissions means that a user may not see certain categories or sub-categories based on their permission level, both in the meta data schema builder and in the assignment of categories to objects.

Import and export functions can now be accessed from the action links at the top of the **Meta Data Administration** screen. The **Import Meta Data Categories** action link is available as soon as you open it.





The **Export Meta Data Categories** action link will appear when you select a Meta Data Category by clicking the check box adjacent to one or more categories.



A couple of things to remember when exporting/importing:

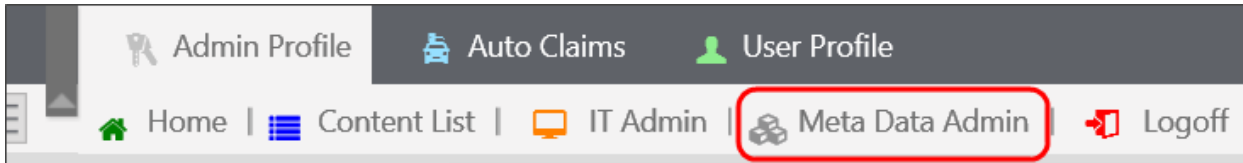
- The Import won't remove any attributes from an existing Meta Data schema, but it will display a warning message indicating that the existing attributes were not in the XML file.
- The Export takes the entire category tree and all selected categories in other tree branches.
- The import can only be performed at the root level of the Meta Data Admin, and the path can't change, so if you export something and import it to another system, the parent path will be exactly the same, starting at the root.

## Creating Meta Data

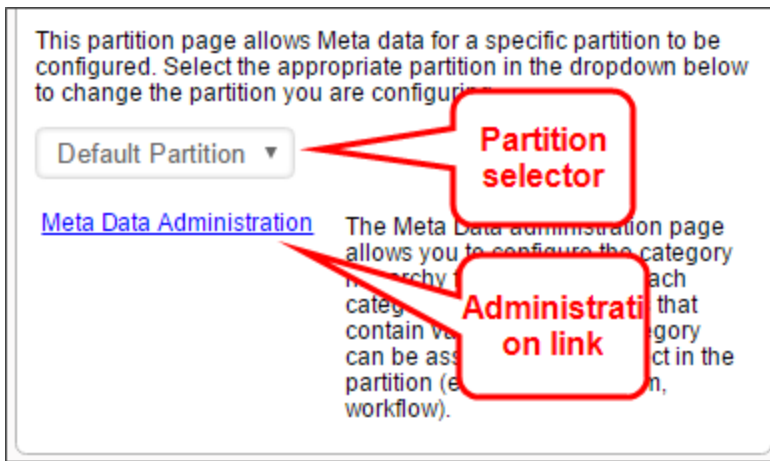
We can't show you a working sample of this feature in the BP Logix [\[Samples\]](#) folder, since Meta Data categories can't be transferred between partitions. So, let's do an in-depth walk through of how and why to create Meta Data categories and attributes inside of Process Director.

First of all, you must have access to the [Meta Data Admin](#) area to create the categories of Meta Data that will apply to the objects inside Process Director. **Access to Meta Data Admin is only available to users with System Administration privileges.**

Select your admin profile and look for the [Meta Data Admin](#) button. If you don't see that button at the top of the screen, you may have to change your Admin profile in the [IT Admin > Configuration](#) section to add the button to your Workspace.



Meta Data is stored by partition, so you'll have to select the partition that holds the objects you want to describe using metadata, then click the [Meta Data Administration](#) link.

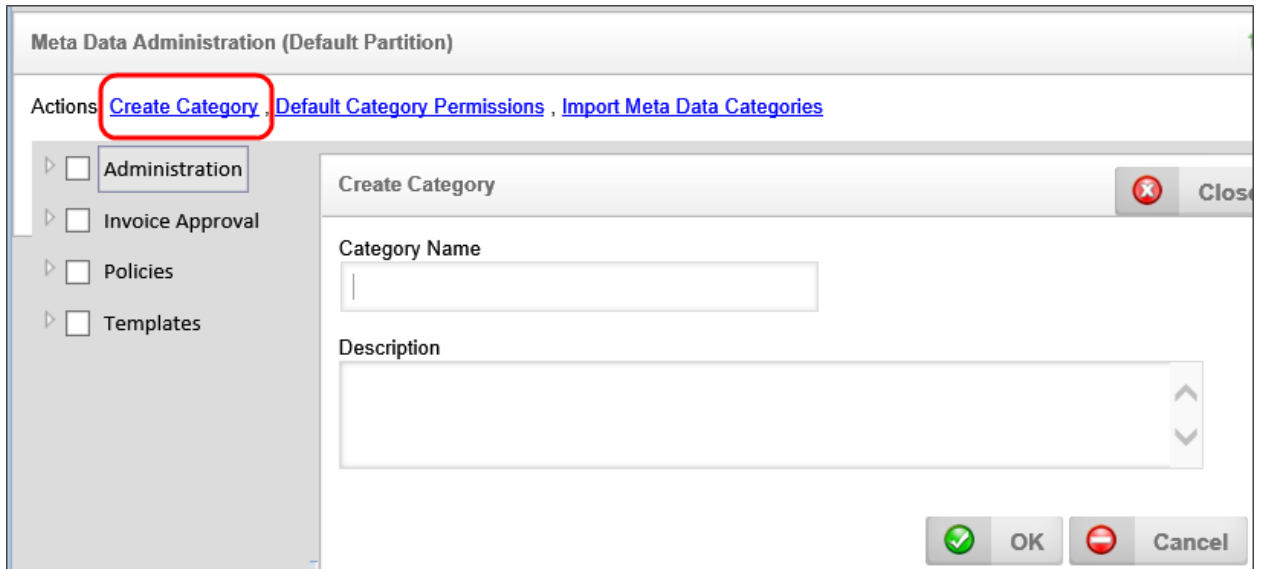


## Meta Data Categories #

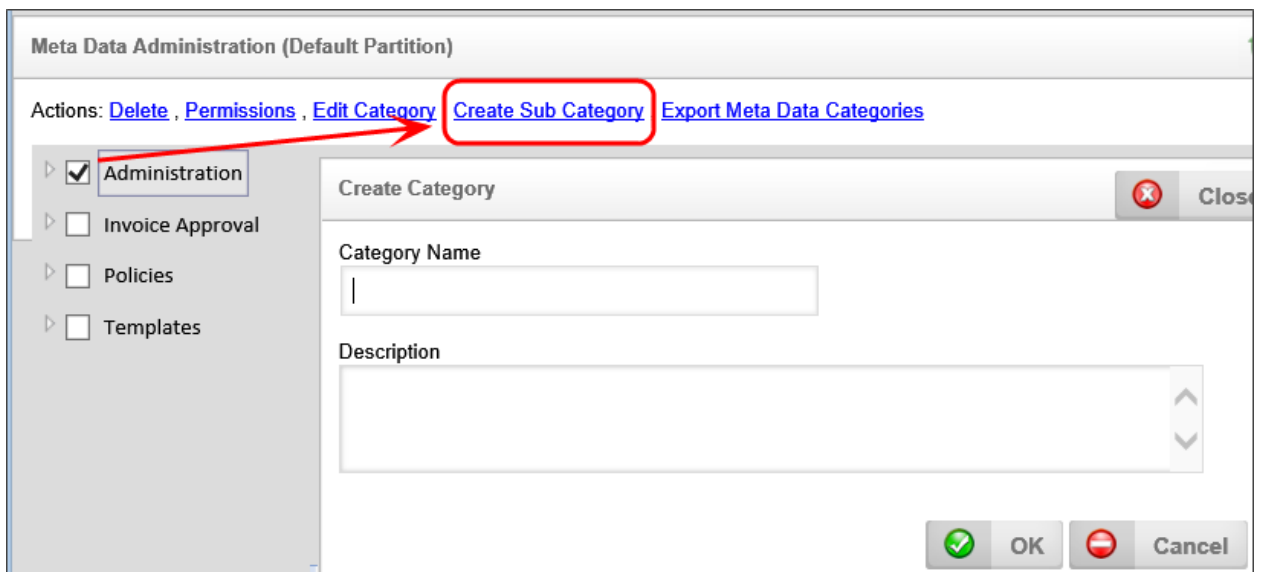
The first and most important step in classifying your content with Meta Data is to define your category schema (i.e. taxonomy). This schema defines a hierarchical tree structure of categories. Each category can have any number of attributes which allow custom information to be associated with the category when it is assigned to an object in the Process Director database. This tree is the foundation for categorizing (or “tagging”) objects and defines how information is organized, processed and distributed. The category tree, or any branch within the tree, should represent how you want data organized in your enterprise.

To create Meta Data categories in the [Meta Data Admin](#) area, click the link labeled [Create Category](#) to open the [Create Category](#) page.

Insert a name and a brief description of what this category contains. Then click the [OK](#) button.



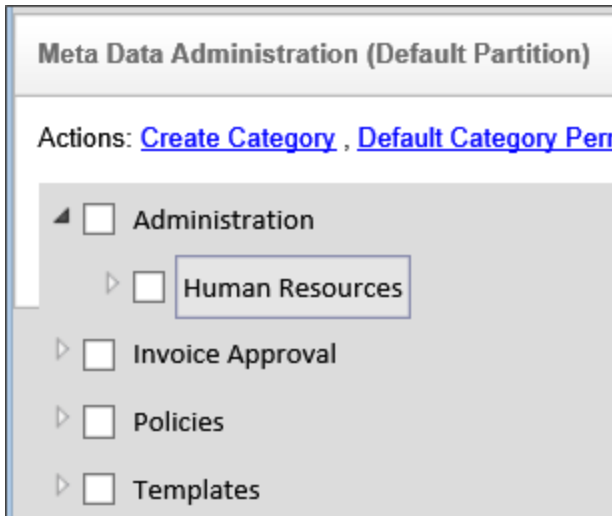
A category may have subcategories as well. To create a subcategory, click on the parent category to select it, then click the [Create Sub Category](#) link.



In this example, there is an [Administration](#) category and we will create a new subcategory called "Human Resources".

When you have finished creating your category, click the **OK** button to save it.

You should now see the new [Human Resources](#) subcategory under the [Administration](#) parent category.

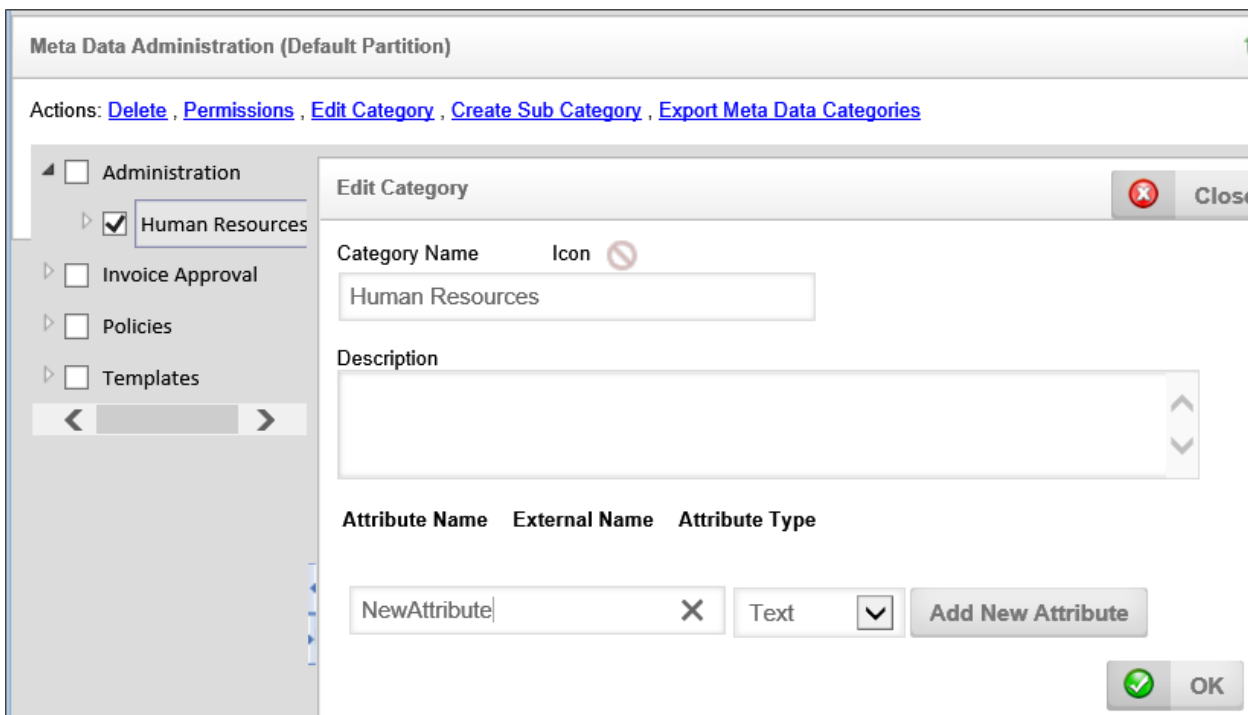


When you are finished creating your Meta Data categories, you can leave the [Meta Data Admin](#) area.



## Meta Data Attributes #

In addition to creating categories, you can also create Attributes for a category. Attributes are different than categories in that, instead of being a predefined value that is assigned to an object or instance, they are data fields into which you can store information from a form or process.

To create an attribute, click on the Category for which you'd like to create the attribute. Type the name of your new Meta Data attribute into the **Attribute Name** text box, select the type of data you'll be storing from the **Attribute Type** dropdown, then click the **Add New Attribute** button.

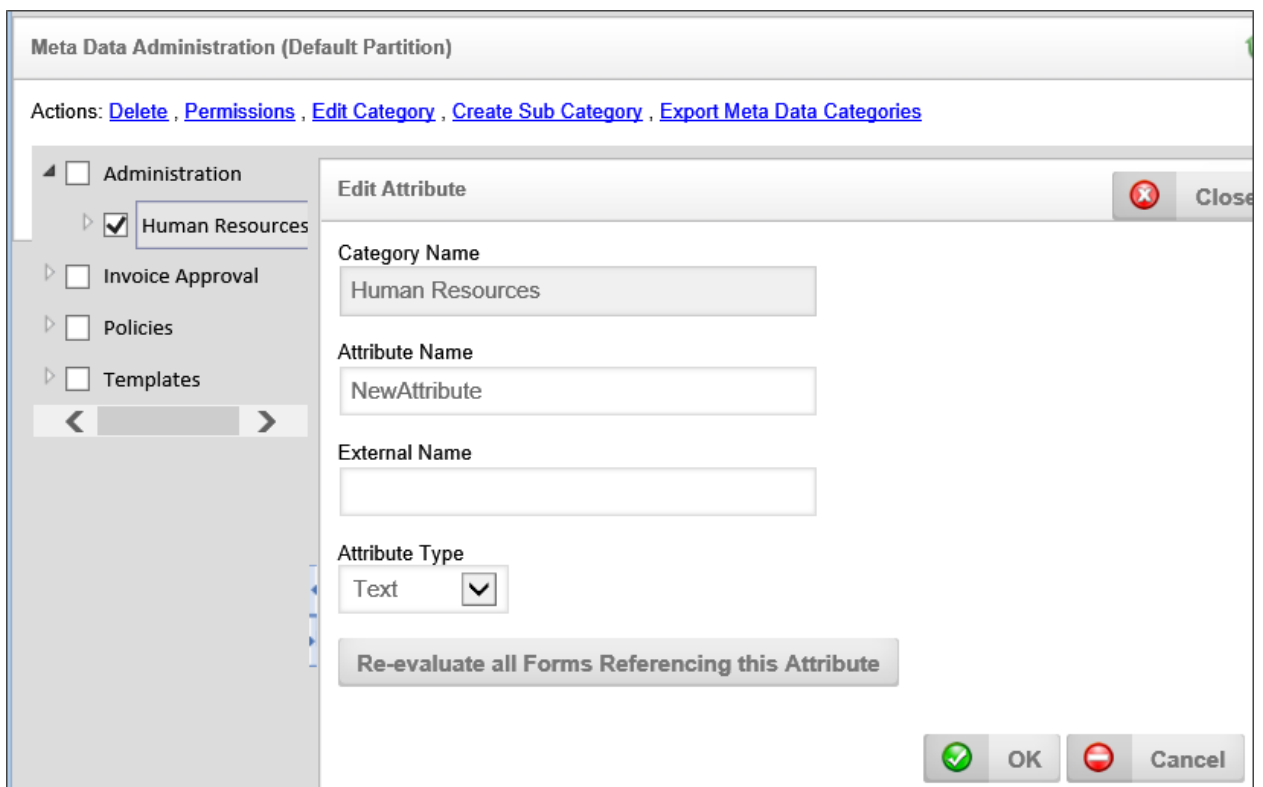


Selecting the **Edit this Item** icon will enable you to change the Attribute's properties, while selecting the **Remove this item** icon will delete the Attribute.

Attribute Name	External Name	Attribute Type	
newattribute		Text	 

The **External Name** can be used while importing XML information from other systems into Process Director.

The **Re-evaluate all forms referencing this Attribute** button should be used when you change attribute types. If you were previously storing numbers in a text format and then switched to numeric, any Business Rules or other form data about that attribute may be evaluated differently.



Once the desired attributes have been created, you can exit the [Meta Data Admin](#) area.

## Documentation Examples #

The examples below use software simulation to walk through the process of creating a metadata schema with categories and attributes.

### Use the Simulator

[Navigate to Meta Data Administration for a Partition](#)

[Create Meta Data Categories and Subcategories](#)

[Create Meta Data Attributes](#)

## Using Meta Data

Once Meta Data categories and attributes have been created, as shown in the [Creating Meta Data topic](#), this meta data can be used to both apply it to object instances, and to use Knowledge Views to filter objects using meta data to return all objects that match some desired Meta Data.

### Assigning Meta Data #

Meta data can be applied to objects by either setting the Meta Data in an object definition, or by using the [Set Meta Data](#) Custom Task. These two methods apply Meta Data to object instances in fundamentally different ways.

When Meta Data is set in an object definition's **Meta Data** tab, all new instances of the object will inherit the configured Meta Data automatically. So, when you wish, for example, all Form instances to apply a specific Meta Data category, setting the Categories in the **Meta Data** tab of the object definition is the easiest way to do that. Additionally, all attachments to the object, such as document attachments, will inherit the meta data categories as well.

There may, however, be cases where you only wish to apply Meta Data to specific instances, or where each instance must have different Meta Data applied. In that case, using the [Set Meta Data](#) Custom Task is the easiest method. This method enables you to determine what Meta Data is applied to an instance, or when the Meta Data should be applied, on an instance-by-instance basis.

The most common use case for using Meta Data is to set meta data categories in an object definition—usually a Form definition—then set Meta Data attributes from Form fields on each Form definition. In that case, each Form instance will have the same categories applied to every instance, but each instance will have unique attributes, set from the Form fields in each instance. We'll describe this use case below.

### Assigning Categories to Objects

Metadata categories once created, can be assigned to Process Director objects via the object definition's **Meta Data** tab. For example, if you open a Form definition, you can assign categories to it by selecting the **Meta Data** tab in the Form definition. Remember, Meta Data Categories are created separately in each partition, so you can only assign Meta Data from the same partition as the Process Director object you are configuring. Most Process Director installations maintain a single default partition, so this isn't usually an issue. But, if you have separate partitions in your Process Director installation, the Meta Data in the partition can only be used for that partition's objects.

In the **Meta Data** tab, you can assign meta data categories to the Form definition by clicking the **Assign Meta Data** button. The object definition, and all new instances that are subsequently created, will be assigned to the category you select. You can also add or replace the meta data categories assigned to existing instances by clicking the **Add** or **Replace** buttons for the child items.

Using this method, you can assign Meta Data to any object definition.

## META DATA

Categories: [ Administration ]

Attributes

Function

Selecting a lower-level category (i.e., a subcategory) will implicitly include all higher categories in the hierarchy tree. For example, let's use the following example image of a category hierarchy:

Actions: [Create Category](#) , [Default Category Permissions](#) , [Import Meta Data Categories](#) , [Meta Data Import History](#)

- ▶  Bid Process
- ▶  Dept
- ▶  Test Description
- ▲  Webinar Webinar category
  - ▲  Documents
    - ▶  HR
    - ▶  IT
  - ▶  Forms

If you apply the [HR](#) category to a Form definition, doing so will automatically include every higher category in the tree, so that the [Webinar](#), [Documents](#) and [HR](#) categories are all applied to the object. All of the categories will be displayed in the [Meta Data](#) tab of the object, once you select the [HR](#) category, as shown below.

Data List Name

## META DATA

Categories: [ Webinar.Documents ] [ Webinar ] [ Webinar.Documents.HR ]

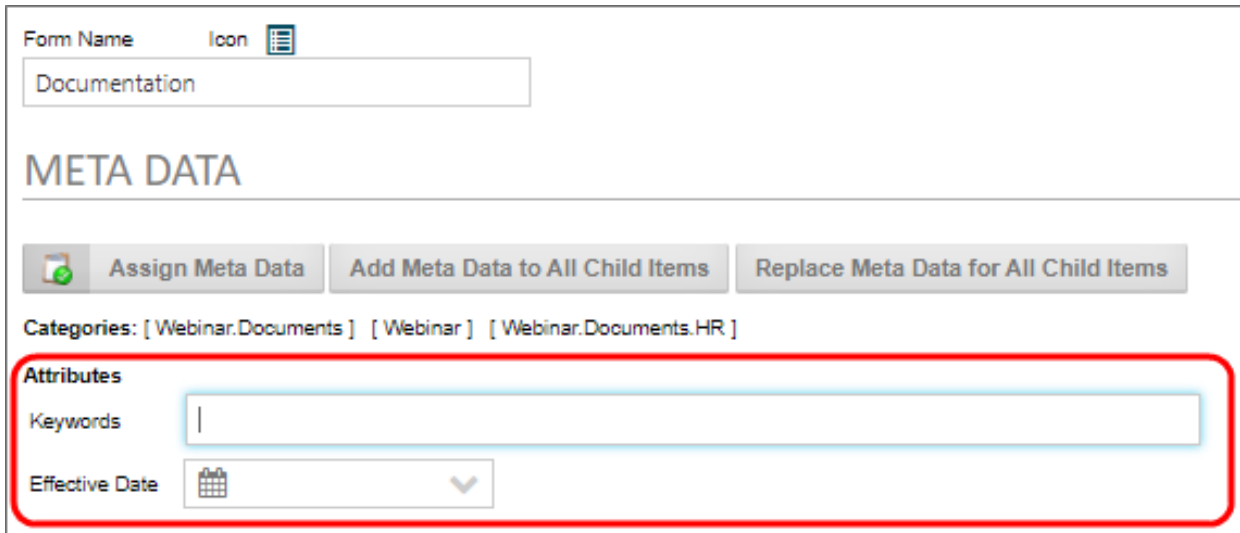
Attributes


The system works this way because Meta Data categories are hierarchical, and thus lower levels of the hierarchy implicitly include all of the higher-level categories of which they are a part. This hierarchical

structure also makes category navigation in Knowledge Views work properly. So, in a Knowledge View, selecting the [Documents](#) category would show all objects that are assigned to both the [HR](#) and [IT](#) categories. We'll discuss Knowledge Views below.

## Applying Meta Data Attributes to Objects


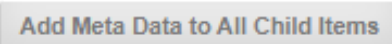
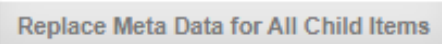
When you apply a Meta Data category to an object, any attributes associated with the category will automatically be added to the [Meta Data](#) tab of the object.



Form Name    Icon 

Documentation



### META DATA

 Assign Meta Data     Add Meta Data to All Child Items     Replace Meta Data for All Child Items

Categories: [ Webinar.Documents ] [ Webinar ] [ Webinar.Documents.HR ]

**Attributes**

Keywords

Effective Date  

Attributes store data that you wish to apply to the Form. There are two methods for assigning an attribute to an object.

### *Meta Data Tab*

On the Meta Data tab of the Object definition, you can supply the values you'd like for the attribute. Any attributes placed into the Meta Data tab will be automatically filled out in **every** new instance of the object when it is created. So, for instance, if you provide an [Effective Date](#) value on the Meta Data tab, every form instance that's created will use that value as the effective date for the form, irrespective of when the Form instance is created.

### *Form Field*

While giving every instance of a Form the same attributes might be useful in some cases, you commonly need to have each object instance associated with unique attributes. For instance, in the example screen shot above, the two available attributes are [Keywords](#) and [Effective Date](#). For these attributes, what you probably want to do is base them on some information found in each form instance.

On the [Form Controls](#) tab of the Form Definition, you can open the [Properties](#) dialog box for a Form field to access the [Link To Attribute](#) property.



**Form Field Properties: EffectiveDate**

Event Field   
  Synchronize Field Within Process   
  Encrypt

ToolTip:

Friendly Name:

Default Value:

**Link to Attribute:**

Case Property:

▼

▼

▼

▼

This property is an **Object Picker** that enables you to navigate to the Attribute you desire to link to the field, in order to store the field's value as the Attribute value for the Form instance. In this example, the form field's name is **EffectiveDate**, so we obviously want to associate the field with the **Effective Date** attribute. Clicking the property's button opens the **Choose Attribute** dialog box.

**Choose Attribute** [No Selection](#) [Update](#) [Cancel](#)

Select the Attribute from the list below.

- [-] Bid Process
  - [-] Dept
    - [-] Test
      - [-] Webinar
        - Documents [Keywords](#)
        - HR [Effective Date](#)
        - IT
- Forms

From here, we can simply click on the link to the [Effective Date](#) attribute. Once we do, the dialog box will close and the attribute will be linked to the [EffectiveDate](#) form field.

**Form Field Properties: EffectiveDate**

Event Field     Synchronize Field Within Process     Encrypt

ToolTip:

Friendly Name:

Default Value:

Link to Attribute:

Case Property:

▼

▼

▼

▼

Now, when a user fills out this field and submits the form, the attribute will be assigned only to this form instance, so each form instance can have a different [Effective Date](#) attribute. We can fill out a new form instance and submit it, then go into the [Content List](#) and open the properties for the form instance. If we navigate to the Meta Data tab, we can see that the [Effective Date](#) attribute has been properly applied to the form instance.

We could, of course, do the same thing for the [Keywords](#) attribute, so that every form instance also has unique keywords as well.

## The Set Meta Data Custom Task #

The [Set Meta Data](#) Custom Task is a Form and Process Timeline Custom Task that enables you to set Meta Data on an individual object instance. As mentioned previously, this Custom Task is used to set Meta Data on an instance-by-instance basis.

When used on a Form, the [Set Meta Data](#) Custom Task can only run on a form instance that has already been submitted. It can't run on new Form instances, since none of the form data is saved until the form has been submitted. The submission of the form is where the form instance is actually created and a record of it kept, including its Object ID, and the Custom Task requires the Object ID to properly associate an object with a Meta Data.

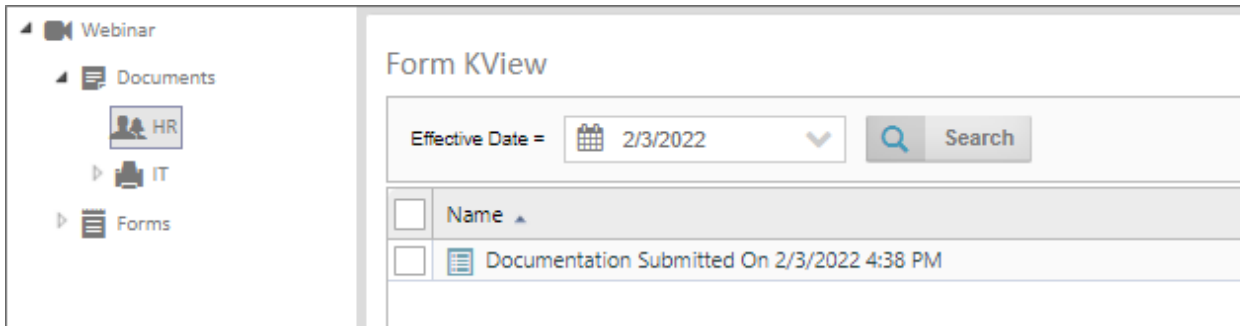
When used in a Process Timeline, the [Set Meta Data](#) Custom Task is run as part of a Custom Task Activity Type.

You can manually apply Meta Data in the configuration of the Custom Task, or apply Meta Data from Form fields. Two form controls, the [Category Picker](#) and [Attribute Picker](#) controls, will display a list of existing Meta Data categories and attributes on the form, making the selection of Meta Data on the Form much easier. When using the [Category/Attribute Picker](#) form controls, you must run the [Set Meta Data](#) Custom Task to apply the values you select in the controls to the instance's Meta Data.

In either case, the Custom Task can set Meta Data categories or attributes on any object, including document attachments. For more information on the [Set Meta Data](#) Custom Task, refer to [the task's documentation](#) in the Custom Tasks Reference Guide.

## Meta Data and Knowledge Views #

Knowledge Views can be configured to enable navigation via Meta Data categories. This property is called [Category Navigation](#).



When **Category Navigation** is enabled in a Knowledge View, the Knowledge View will display a list of all available categories on the left side of the screen. Clicking on a category will filter the right side of the screen to show only forms, documents, or other objects that match the category you select on the left.

In addition, you can configure the Knowledge View's **Filter** tab to include the Attribute as a filter, as shown in the example above. As you can see, the document we submitted in the Attributes section, above, is the only document in the **HR** category that has an effective date of 2/3/2022.

So, when using a Knowledge View that enables category navigation, the Category filtering is automatically done via the Knowledge View's navigation pane, while and desired attribute filtering must be done via a filter configured in the Knowledge View definition's **Filter** tab.

## Process Timelines



For Process Director v6.0.300, the look and feel of the Process Timeline has been refreshed with a more modern design. These changes are largely visual, and the fundamental operation of the object is unchanged.

Process Timeline is the automation of a business process in which documents, information or tasks are passed from one participant(s) to another for action. A Process Timeline is made up of many functions and activities such as a review process, **Task Lists**, notifications, alerts/triggers, reminders, context sensitive tasks, an approval process, status/tracking, due dates and reporting.

### What is a Process Timeline Definition?

Process Timeline Definitions in Process Director are a series of logical activities, each with a specific task and assigned participants. The Process Timeline defines the path or route that a Form or document must take. Each Activity in the Process Timeline path defines the task type, the participants, and the rules that govern how the Process Timeline will advance or transition to the next Activity.

The Process Timeline may look familiar to you, because it is presented in Gantt chart format. The Gantt chart is a type of chart that places each Activity in a process on a single line that is horizontally marked in some increment of time, usually days. Each Activity is marked with a bar that whose left end begins at the task's start date, and ends at the task's end date. This type of chart is widely used in project management.

**Training TL - Training**

Options

Set Form Data

Actions: [Create Activity](#), [Expand All](#), [Collapse All](#), [Import MS Project](#), [Export as MS Project](#)

<input type="checkbox"/>	Activity	Type	Duration	Actions	Timeline (Days)
					1 2 3 4
<input type="checkbox"/>	1 Set Req No		0 Day(s)		
<input type="checkbox"/>	2 Approval Section		2 Day(s)		
<input type="checkbox"/>	2.1 Supervisor Approval		1 Day(s)		
<input type="checkbox"/>	2.2 Manager Approval		1 Day(s)		
<input type="checkbox"/>	3 HR Action		1 Day(s)		
<input type="checkbox"/>	4 Requester Notification		0 Day(s)		

Process Director uses this generic Gantt chart format for the Process Timeline, and adds additional functionality for use in modeling and managing business processes. BP Logix developed the Process Timeline to address the need to measure and predict process execution times.

Time isn't a process driver for a flowchart-based workflow process. Indeed, looking at a traditional process diagram gives you no sense of how long completing the process, or any process task, will take. The Process Timeline, on the other hand, shows you immediately and graphically how much time your process, and each Process Activity, should take. Because the Process Timeline incorporates time explicitly as a process driver, and records the process times for each Timeline instance, Process Director can, using historical data, predict how long each Activity and process will take, and compare that prediction to your configured times. For instance, with the Process Timeline, Process Director can use historical data to predict whether an Activity will be early or fail to meet its configured due date, often well in advance of the Activity starting. This predictive ability enables Process Director, using the Process Timeline model, to provide you with the earliest possible notification of slippages in the process' scheduled run time.

Another important difference between traditional process workflows and Process Timelines is that Timelines are **constraint-driven**. In a traditional process, tasks don't start until they are prompted to begin when the process follows a specific branch. In a Process Timeline, however, all Timeline activities want to run all the time. The designer's job with a Process Timeline is to apply *constraints* to each Activity to

prevent it from running immediately. By default, this makes Process Timelines implicitly parallel. If you add two activities to a new Process Timeline, but don't configure any restraints, then both activities will run immediately, and in parallel, as soon as the Process Timeline instance starts. Conversely, in a flowchart-based process model, you must explicitly specify when two steps must run in parallel.

Business users design Process Timelines by answering three questions that define the Activity's constraints, as they add each step to the process:

- **What activities must complete before this Activity begins?** We call this the "dependency question". For each Timeline Activity, you can specify what other Activity, or activities, must complete, before this Activity can begin. Process Director constantly evaluates this question, so that activities can start as soon as their dependencies are satisfied. Dependencies are configured on the **Start When** tab of a Timeline Activity.
- **What conditions must apply before this Activity can begin?** We call this the "eligibility question". You can create a Start When condition or conditions that must be true before the Activity can begin, even if the dependencies are complete. This question is also constantly evaluated, in conjunction with the dependency question. The eligibility conditions, or "start when" conditions, are also configured on the **Start When** tab of a Timeline Activity.
- **Is this Activity needed?** We call this the "necessity question". If both the dependency and eligibility questions are satisfied, Process Director evaluates this Needed When condition, or conditions, once, when the Activity begins. If the Activity isn't needed, Process Director will skip the Activity, and automatically move to the next Activity. The necessity conditions are configured on the **Needed When** tab of a Timeline Activity.

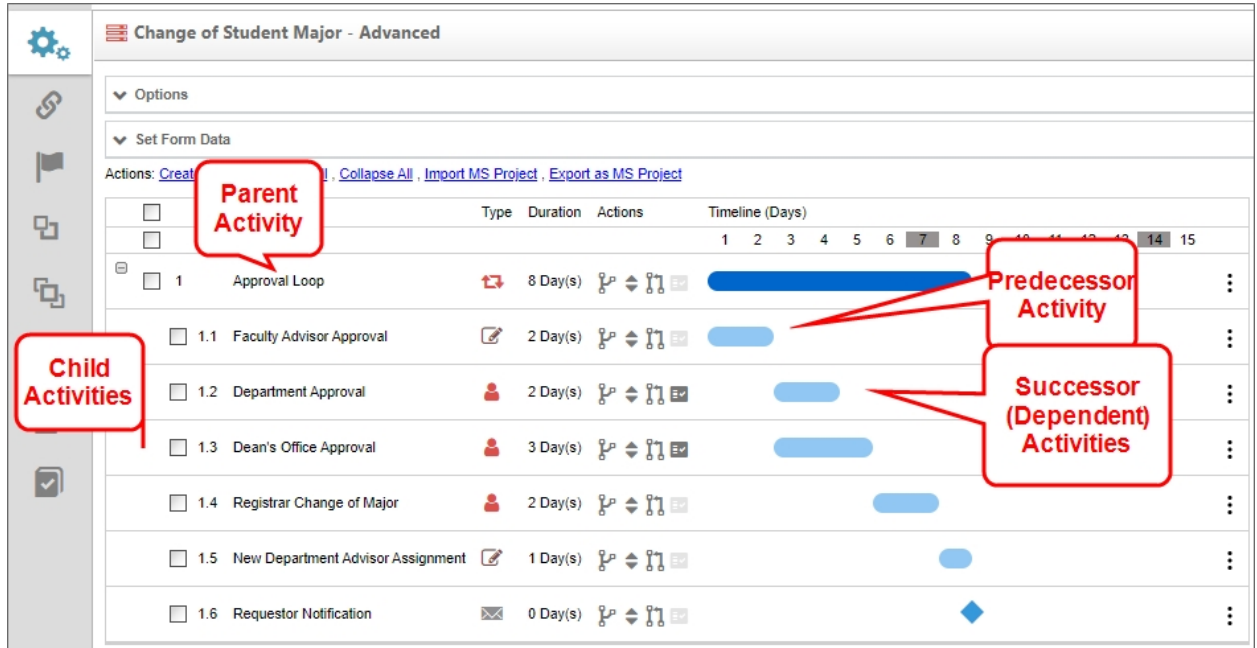
The Process Timeline's constraint-driven model enables organizations to build robust, complex executable process models within Process Director much more easily than the flowchart-based model can allow. The result is a solution with many valuable features:

- Modeling is greatly simplified. Project owners list each Activity, estimate its duration, and then drag-and-drop it onto the Activity or activities that must complete before it can begin.
- As many of the activities as possible will run at the same time, without the need to explicitly configure parallel behavior.
- The status of the process can be determined at a glance.
- At any point—even the moment the process is launched—the system can determine which future activities, if any, may not be complete by their due date.
- The system records actual versus predicted execution times each time the process is run, and adjusts its time estimates accordingly.

Most activities have a dependence on one or more other activities. Dependence refers to the requirement that an Activity can't begin until a predecessor Activity, or activities, ends. This type of relationship (often referred to by project managers as finish-to-start) represents the most fundamental starting condition for any Activity.

Below is a sample Process Timeline, consisting of several activities. The Timeline column of the Process Timeline shows the number of days planned for each Activity. [Faculty Advisor Approval](#) is scheduled to take two workdays, [Department Approval](#) is expected to take three workdays, and so forth. All of the

activities will be tracked for start times and due dates, and successor, or dependent, activities will begin as soon as their predecessor activities end, via the automated process that's governed by Process Director. The [Registrar Change of Major](#) Activity is dependent on the [Dean's Office Approval](#). The [New Department Advisor Assignment](#) Activity, in turn, is dependent on [Registrar Change of Major](#).
















An Activity may also be a parent or child Activity. A parent Activity is a container for other, subordinate activities, each of which is referred to as a child Activity. Child activities have an implied dependence on their parent, so, absent any additional conditions, the child activities begin when the parent Activity begins. Once all of the child activities are complete, the parent is complete.

### Activity Types

Each Activity in the Process Timeline has a **Type** column that displays an icon to identify the activity type. Each activity type displays a unique icon. In the example above, the [Department Approval](#) Activity's Type icon indicates that the Activity is a User Activity Type that is assigned to a Group of users, While the [Registrar Change of Major](#) Activity's icon indicates the activity is a user activity assigned to a single user. We'll discuss each **Activity Type** in detail in the documentation topic for each type, but the available icons you might see in this column are shown below.


ICON	Type
	Parent Activity without Looping Conditions
	Parent Activity with Looping Conditions
	User Activity - Single User

ICON	Type
	User Activity - Group of Users
	User Activity - Chosen from Form Field
	User Activity - Chosen via System Variable
	User Activity - Chosen via Business Rule
	User Activity - Timeline Initiator
	Notify Activity
	Process Activity
	Script Activity
	Form Actions Activity
	Branch Activity
	End Process Activity
	Wait Activity
	Case Activity




An additional **Activity Type**, the Custom Task Activity, does not have a specific display icon. Instead, this Activity type will display an icon in the **Type** column that is specific to the Custom Task that's configured to run in the activity.

### Activity Actions

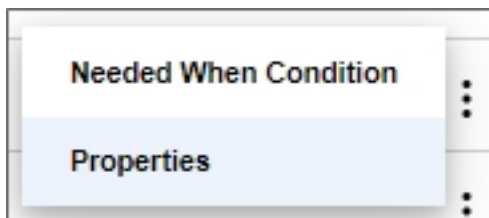
Similarly, each Activity in the Process Timeline has an **Actions** column with four icons, each of which enable you to modify one of the Activity's properties.

ICON	ACTION
	Drag to move under another parent: Drag the Activity to a parent Activity. The Activity you drag will become a child of that parent.



ICON	ACTION
	Drag to move order: Change where the Activity appears in the Process Timeline. Note: changing the location of the Activity doesn't change the behavior of the Process Timeline, which is strictly governed by the rules of dependence and other Activity starting conditions.
	Drag to add a Depends On: Drag and drop an Activity onto another to make the dragged Activity dependent upon the target.
	Conditions: Click this icon to open the Activity's properties on the Needed When tab. If there are no conditions, this icon will display in a lighter gray color.

The properties for the Activity can be accessed via the **Properties** menu on the hamburger menu at the right side of each activity. The **Needed When Condition(s)** can also be accessed from this menu.



An Activity can be added to the Process Timeline by clicking the **Create Activity** action link. When a new Activity is added to the Timeline, it will be automatically selected and highlighted. Once an Activity has been added, the properties for each Activity are presented in a tabbed interface that can be accessed by double-clicking on the Activity name, or clicking on the Properties icon for the Activity. Each tab in the interface groups similar properties together.

TAB	DESCRIPTION
Activity	This tab enables the user to select the Activity Type, such as a User, Custom Task, Form Action, etc. It also contains other general Activity settings.
Activity type	This tab will show specific properties that pertain to the Activity Type you select. The tab's label will change to reflect the selected Activity Type.
Start When	This tab enables you to inspect and modify the dependencies and conditions that determine when the Activity will start.
Completed When	This tab enables you to inspect and modify the conditions that determine when the Activity will complete. Rarely used, as most activities complete organically (such as when a user submits a form, or an automated task completes).
Needed When	This tab enables you to inspect and modify the conditions that determine if an Activity will be run once its start conditions have been met. These conditions will be evaluated once when the Activity becomes eligible to start.
Due Date	This tab enables you to inspect and modify the due date for the Activity, and the actions to take when the Activity isn't completed by the due date.
Notifications	This tab enables you to configure who is notified about the Activity and when noti-

TAB	DESCRIPTION
	fications should be sent.
Advanced Options	This tab enables you to configure advanced properties, such as how the Activity is assigned to user, what steps to take when a user completes the Activity, etc.

## Processes and Subprocesses #

A business process may occasionally need to invoke a separate process as a subprocess, or child process, of the main process. This subprocess may need to run synchronously or asynchronously with the main, or parent, process.

A **synchronous** subprocess pauses the parent process while it is running. Once the subprocess completes, the parent process continues from the point where the synchronous subprocess began. The parent process can't continue until the subprocess is complete.

An **asynchronous** subprocess, on the other hand, does *not* pause the main process while subprocess runs. Instead, both processes will run in parallel, and both will complete independently of each other.

To invoke a subprocess to run synchronously, you can use the [Process Activity Type](#). To invoke a subprocess to run asynchronously, you can use the [Custom Task Activity Type](#), and use the [Run Process Custom Task](#) to start the desired subprocess.

As an example, a Process Timeline that runs a Process activity, or which calls the Run Process custom Task, becomes the parent Timeline, while the Timeline that is invoked by one of these two methods becomes the child timeline.

## Background Operations #

A Timeline Activity of the Custom Task, Script or Process types can be configured to run in the background by checking the **Run this Activity in the background (if unsure, leave this box unchecked)** check box in the **Advanced Options** tab. Setting the operation to run in the background can be used to prevent a long-running, machine-centric Activity in the Process Timeline from hanging a user session while the processing occurs. The Activity will be run in a different context from the current user's, even if the user invoked the transition of the Activity. Additionally, if you have a [Rendering Server](#) enabled, the background processing will be conducted by the Rendering Server when the option is checked.

The screenshot shows a dialog box titled "Activity 1" with a "Name" field containing "Activity 1". Below the name field are several tabs: "Activity", "Custom Task", "Start When", "Completed When", "Needed When", "Due Date", "Notifications", and "Advanced Options". The "Advanced Options" tab is selected. Inside this tab, there is a checked checkbox labeled "Activity is required to complete parent". Below that, there is an unchecked checkbox labeled "Run this activity in the background (if unsure, leave this box unchecked)", which is highlighted with a red circle. At the bottom of the dialog are "Close" and "Cancel" buttons.

Please note that, when this option is selected, if the user's task is followed (after one or more intervening "background" tasks) by another task assigned to the same user, the current behavior in which the window remains open and is refreshed with the Form for the subsequent task will no longer be seen. Instead, the Form will close, and the user will have to click the appropriate link in their [Task List](#) or email notification to open the form to complete the new task.

On some systems, when starting a subprocess using the asynchronous option, the system can mark the calling task as complete before the called subprocess completes. This may be especially true if the subprocess contains complex rendering operations. To avoid this, a wait time, in seconds, can be set using the [nAsyncSubProcessWaitSecs](#) variable in the Custom Vars file. The default setting for this variable is 5 seconds.

## Other Process Timeline Topics

To see other Process Timeline topics, you can use the Table of Contents displayed on the upper right corner of the page, or use one of the links below.

[Process Timeline Definitions](#): General Properties for configuring a Process Timeline.

[Process Timeline Activities](#): Properties for configuring Timeline activities.

[Managing Process Timeline Instances](#): Information about how to perform administrative actions on Timeline instances.

## Process Timeline Definitions

Process Timeline Definitions can be created and modified by implementers. Use the [Content List](#) tab in the Process Director navigation bar to view the Content List screen. You must have Modify permission in the folder you are viewing to be able to create a new Process Timeline Definition in that folder.

Process Timeline Definitions can be stored anywhere in the [Content List](#), under any folder structure, though BP logix recommends you use the organizational method suggested in the [Content List Folders section](#) of the Importing/Exporting Content topic.


## Process Timeline Options #

To configure the Process Timeline Definition settings, click on the [Options](#) section when viewing the [Properties](#) tab of the Process Timeline Definition. This will open the section to enable you to set the general Process Timeline options.

The screenshot displays the configuration page for a 'PR/PO Process'. The interface includes a sidebar with navigation icons and a main content area with the following fields and options:

- Timeline Name:** PR/PO Process
- Description:** (Empty text area)
- Instantiated Name:** {TIMELINE\_DEF\_NAME}{TIMELINE\_ATTACHMENTS,ObjectType=FORM,pre=" (" ,p
- eForm for Timeline (will attach eForm instance if needed):** (Empty field with a menu icon)
- Default Email Template in this Timeline:** Notification Email Template (with a menu icon)
- Timeline Script:** (Empty field with a menu icon)
- Copy objects in Timeline Package to Parent
- Only from this group:** (Empty field)
- Copy objects from Parent Process to this Timeline
- Only from this group:** (Empty field)
- Remove Timeline instance after process completes
- Do not show this object in 'Items I Can Run' Knowledge Views
- Priority:** 1 (dropdown menu)
- After a user submits a form which runs this process:** Automatically show user's next task if it is in this process (dropdown menu)
- Show columns:**  Index  Activity  Type  Actions  Timeline
- Display timeline in:** Days (dropdown menu)
- Default activity duration:** Business Days (dropdown menu)
- Reset Predicted Values for Instances** (button)
- PRID:** 21056777-86c2-440e-bab1-5a6c33bc58a6

## Process Timeline Icon

A custom icon can also be configured for this Process Timeline Definition. This icon will be displayed for all running Process Timelines for this definition (e.g. [Task List](#), [Content List](#), Knowledge View, etc.). To change the icon used for this Process Timeline Definition, click on the  icon.

## Process Timeline Name and Description

The Process Timeline Definition name and description are important because this is what a user will see when selecting a Process Timeline to run. The description should describe why and when this Process Timeline Definition should be run.

## Instantiated Name

This field contains an optional name that should be used when a Process Timeline is started (i.e. instantiated). The default name of a running Process Timeline is the same name as the Process Timeline Definition. This name can contain system variables using the {SYSVAR} tag. This property allows the running Process Timeline names to contain more meaningful information (e.g. "PR Review – started by {CURR\_USER}). The running Process Timeline name is re-evaluated after each Process Timeline Activity completes allowing variable data (e.g. Forms field values) to be used to construct the name. Refer to the section named System Variables in this document for more information.

## Form for Process Timeline

You can optionally link a Process Timeline to a specific Form definition in the [Content List](#). When this is configured you can choose a form field from a dropdown list of all fields in that form instead of choosing a form then a form field.

### **Default Email Template in this Process Timeline**

This is an optional default email template to use when sending email notifications to users in an Activity. If an email template isn't specified in an Activity, this default email template will be used. If no default email template is assigned, the system default email template will be used that was shipped with Process Director. For more information on email templates refer to the [Email Templates](#) section of this guide. Email Templates are stored in the [Content List](#) as a Form that is specifically configured as an Email Template Form.

### **Timeline Script**

Select any custom script that you'd like to associate with the Timeline.

### **Copy Objects in Process Timeline Package to Parent**

If checked, this option will copy objects in this Process Timeline package to the package's parent. You can optionally limit the objects copied by group.

### **Copy Objects from Parent Process to this Process Timeline**

If checked, this option will copy objects from this Process Timeline's parent process to this Process Timeline. You can optionally limit the objects copied by group.

### **Do not show this object in 'Items I Can Run' Knowledge Views**

If checked, this option will prevent the Timeline from being displayed in the [Items I Can Run](#) Global Knowledge View.

### **Priority**

The Process Timeline priority is used to show participants the importance. The higher priority items are displayed first in a user's [Task List](#). The Process Timeline's Priority column can be resorted in the user's [Task List](#).

### **Remove Timeline Instance after process completes**

This will delete the Timeline instance from the [Content List](#) when the Timeline instance is complete. This is useful for Timelines that perform automated tasks for which you don't need to keep a record of once the tasks are finished.

### **After a user submits a form which runs this process**

In some cases, a user may be assigned to two or more activities in a row. The default behavior for Process Director is to immediately display the Form to the user to perform the second task once the first task is completed. This may be confusing, as the user generally expects the Form to close when a task is completed. This property enables you to change the default behavior so that, if a user is assigned to two activities in a row, the Form won't immediately reload to complete the second task. To perform the second task, the user will have to manually re-open the form from the [Task List](#).

For Process Director v5.26 and higher, this setting also enables the next task to be shown to unauthenticated users, as well.

### Show Columns

This property enables you to display the Index, Activity, Participants, and/or the Process Timeline columns of the Process Timeline Definition.

### Display Process Timeline in

This property enables you to display the Process Timeline in minutes, hours, days or weeks. You may also specify the default Activity duration.

### Reset Predicted Values for Instances

Process Director uses the historical task performance times for the Timeline to make predictions of future performance, such as when tasks are predicted to be completed. Over time, changes to the Timeline definition, the process, or other changes, may invalidate the historical data, and skew Process Director's predictions. Clicking the **Reset Predicted Values for Instances** button will erase the historical performance data, and begin recompiling it from the date it was reset.

### PRID

The unique GUID, or identifier, of the Timeline definition is displayed here.

### Set Form Data #

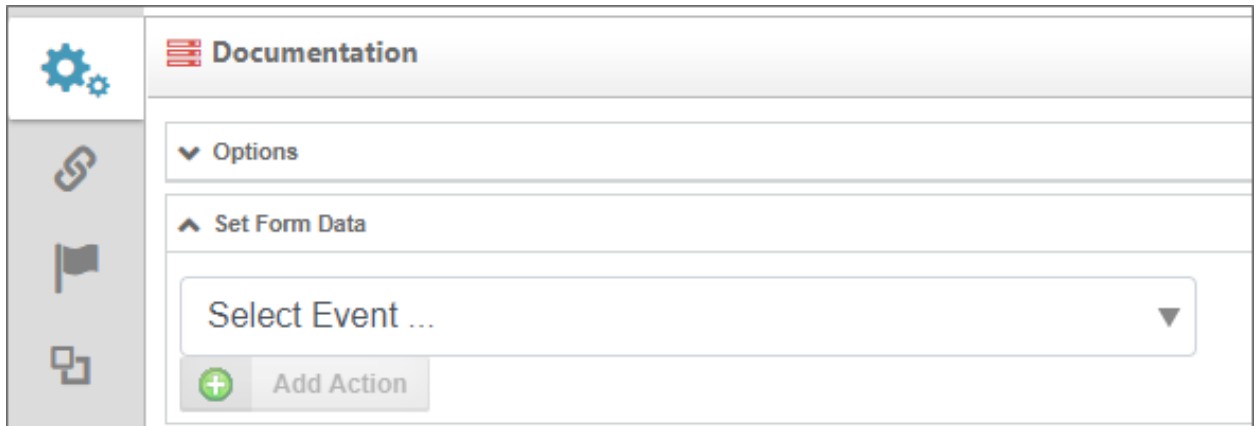
For Users of Process Director v3.75 and higher, Form data field values may be set directly from the Timeline definition. The field value can be set when a Timeline Activity task starts or ends. This is a very useful option when you want to set form data without having to open the Form. This function is especially useful for setting a form or process status, or setting action dates, without having to open the Form. In most cases, the **Set Form Data** tab on the Timeline Activity will replace the requirement to use a Custom Task to set form data in the Timeline.

You can configure the Set Form Data function in either the **Set Form Data** section of the Timeline Definition, or on the **Set Form Data** tab of an individual Timeline Activity. Irrespective of which method you use, the Set Form Data settings you configure will be displayed automatically in both places. Configurations can be edited in either location as well.

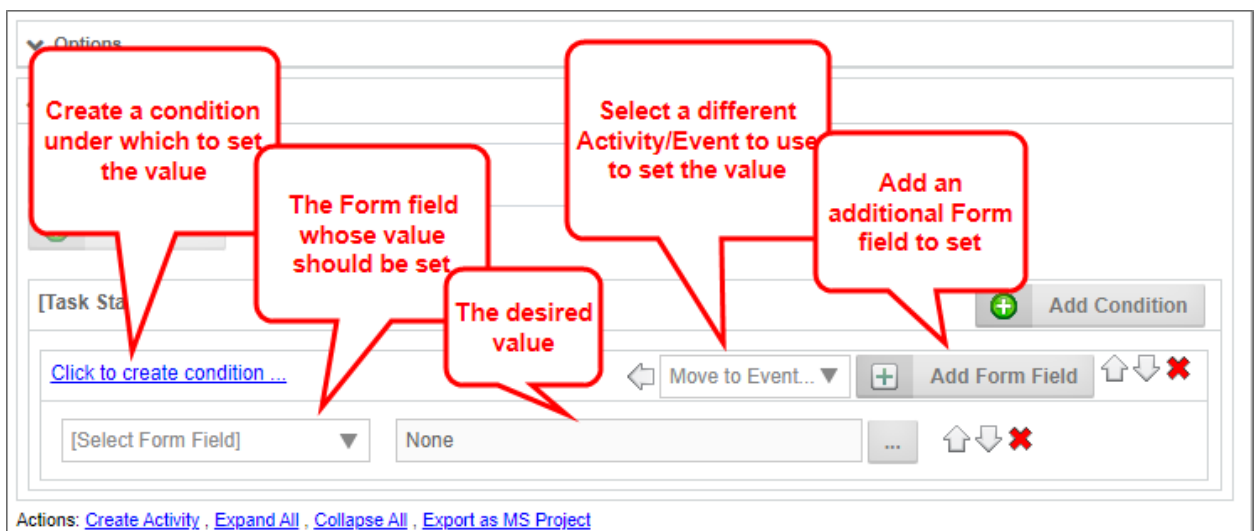
To set a Form field value, first ensure that you have specified a form to associate with the Process Timeline, via the **Form for Timeline** property in the **Options** section of the Timeline Definition. Next, in the **Set Form Data** section of the Timeline definition or in the **Set Form Data** tab of an individual Timeline Activity, select the Activity and Activity event for which you'd like to set the value. The two Activity events that appear as usable events are **Task Start** and **Task End**. An additional option for the event, **No Event**, can be used as a placeholder. The Set Form Data action will NOT occur if the event is set to **No Event**.



For Process Director v4.03 and higher, these selections are available as separate dropdown controls. Prior versions present them as a single dropdown control.



For each Timeline Activity, you have the choice to select the Activity, or to specify that the value must be set when the Activity begins or ends. Once you select the task or specific event that should set the Form's data, the selected item will appear in the **Set Form Data** section.

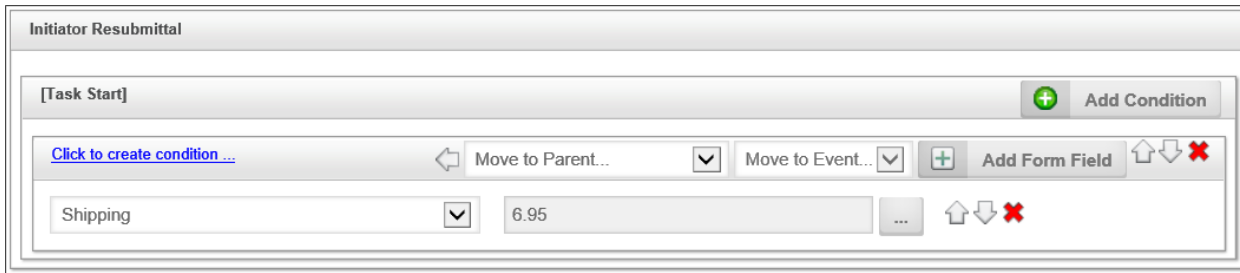


As the example above shows, you now have a variety of settings you can apply to set the Form field's value.

- **Click to Create Condition...:** Invokes the [Condition Builder](#) to create a condition set under which to set the value.
- **Move to Event:** Select a different event, in order to move the Set Form Data action to a different Activity/event.
- **Add Form Field:** Select a different event, which will change the event under which the Set Form Data will run.
- **Add Condition:** Add another condition to this event, to set a field's value.

You can change the order of conditions, or fields by clicking the up or down arrow icons, or you can delete an item by clicking the red "X" icon.

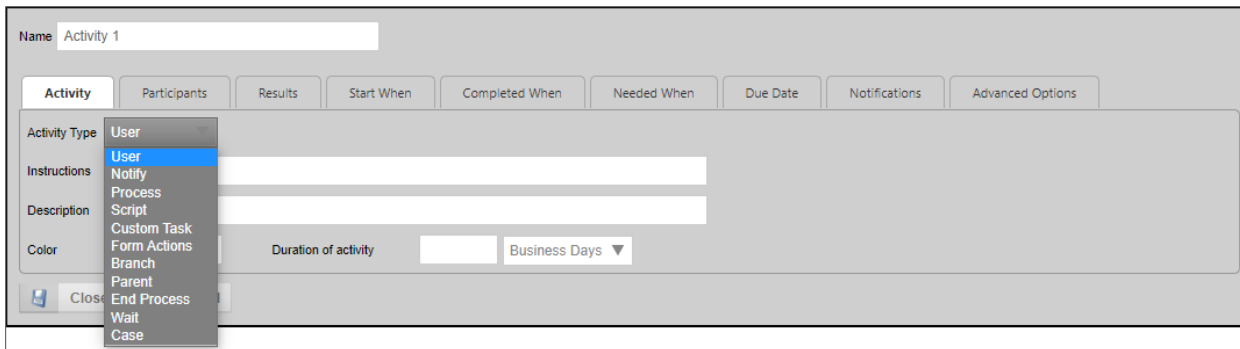
In the example below, the generic Initial Approval Activity was selected, selecting the Task Start as the event. When the Activity runs, the Shipping field is set to a cost of \$6.95.



## Process Timeline Activities


A Process Timeline is made up of a series of activities. Each Activity in a Process Timeline identifies the task to be performed and the users, if any, that should perform the task. Activities can be easily added, moved and deleted from a Process Timeline Definition.

Each Timeline Activity defines a set of functions or actions that can be performed by that Activity. The function that the Activity will perform is determined by the **Activity Type** property that appears on the **Activity** tab for the Timeline Activity.



Please refer to the [Process Timeline Activity Types](#) topic for specific information about specific Activity Type, and how to configure them.

## Common Activity Settings #

To configure an Activity in the Process Timeline, double click on the Activity or click on the Properties  icon to display the **Properties** dialog box for the Activity.

### Name Property

At the top of the **Properties** dialog box there's a **Name** property, which identifies the Activity in the Process Director user interface. The value set in this property will be displayed in the Process Director UI for all references to the Activity. For instance, it is used to display what Activity in the Process Timeline is currently running. The **Name** is also displayed on the **Routing Slip** to show when the Activity ran.

The **Properties** dialog box is presented in a tabbed interface, with each tab grouping similar properties together. The number of tabs displayed to configure an Activity will depend on the Activity Type. Most Activity Types have specific configuration tabs that only apply to that Activity Type. For instance, User activities have a **Participants** tab, which enables you to specify the users who must be assigned a task to complete the Activity. A Notify Activity, on the other hand, only sends email notifications and doesn't assign any tasks, so doesn't have a **Participants** tab.



There are, however, several configuration tabs that are common to all Timeline activities.

## Activity Tab #

The **Activity** tab enables you to specify the Activity's general settings, the most important of which is the **Activity Type**.

## Activity Type

This dropdown contains type of Activity to be used for the current Activity.

The default **Activity Type** is the User Activity. With the User Activity selected, the second tab on the options dialog will be **Participants**. Selecting a different Activity Type will change the second tab to a different option setting, and some Activity Types will remove some of the tabs completely, as they contain inapplicable settings for that type of Activity.

## Description

This field contains an optional description of this Activity. This can be used by the Process Timeline builder to document what this Activity is used for and why it is here. The end-users won't see this field.

## Instructions

This contains instructions for the users telling them how to perform the task assigned to them. This information will be contained in the email sent to the users, it will be displayed on their **Task List**, and is displayed in the Process Timeline Package.

## Color

This field contains an optional color of this Activity, which, when set, will change the color of the bar that displays in the Gantt chart for this Activity. The use of color can be helpful to make different sections of the Gantt chart, or individual activities, more visually distinct.

## Duration of Activity

This field contains an option to set the duration of the Activity. This property consists of a text box to enter a number of time increments, and a dropdown to specify the increments to measure. So, if you wanted an Activity to last for two weeks, you'd enter "2" in the text box, then select "Weeks" from the drop-down control.

## Results Tab #

The **Results** tab appears for User and Wait activities, where the results are configured explicitly. Other activities, however, have results, even if they are not configured directly by the designer. For instance, the Parent activity's results are, by default, the result of the last child activity, though this can be configured to use a comma-separated list of all child activity results, if desired.

## *Subprocesses and End Process Activities*

The Process activity type requires that a Process Timeline definition be configured. The configured process will be started as a subprocess (child process) of the main (parent) process. You may copy objects from the parent process to the child process and vice versa.

When a subprocess contains an End Process activity, it will return the name of the End Process step as a result for the parent Timeline Activity.


The screenshot shows a configuration dialog for an activity named "Activity 1". The dialog has a "Name" field containing "Activity 1". Below the name field are three tabs: "Activity", "Start When", and "Needed When". The "Activity" tab is selected. Under the "Activity" tab, there are three fields: "Activity Type" with a dropdown menu set to "End Process", "Description" with an empty text box, and "Color" with a dropdown menu set to "Default". At the bottom of the dialog are four buttons: a save icon, "Close", a red stop icon, and "Cancel".

The End Process Activity in a Process Timeline stops the running of the Process Timeline immediately. As such, the End Process Activity can only run once in a Process Timeline. So, if you have parallel activities running in a Process Timeline, all of the parallel steps must complete before the End Process Activity is reached. Process Timelines with multiple End Process activities will need to have conditions set on each End Process Activity to ensure that only one of the End Process Activities runs. When the subprocess completes, the name of the End Process Activity that was run will be returned to the parent process as the result of the Process Activity that started the subprocess.

## Start When Tab #

The settings on this tab determine when the Activity will run by specifying its dependencies, i.e. what Activity or activities must complete before the Activity starts. The Activity's start time can depend on the state of other activities, or some evaluable condition. Activity dependencies can be added via the **Add Dependence** button, and the dependence specified via a dropdown. Conditions that make the Activity eligible to start can be configured via the **Condition Builder** by clicking on the **Click to create condition...** link.

Clicking the **Recurring Activity** check box makes an Activity a recurring Activity, and enables you to configure the time interval at which the Activity repeats. The Activity won't recur if the box is unchecked. This option is relevant primarily for activities in which you have set a Start When condition. When an Activity is set as a recurring Activity, the Activity, *once completed*, will automatically restart again at the time interval specified, until the Start When condition is no longer met.

 If you make an Activity a recurring Activity without a Start When condition, the Activity will continue to restart indefinitely, until the system's internal loop checks cancel the recurrence as an indefinite loop.

To set an Activity dependence, click on the **Add Dependence** button. A dropdown list will display of all the activities that are available in the Process Timeline. From the dropdown list, you can select the Activity which must end before this Activity will be allowed to start. You can add more than one dependence, and the Activity won't start until **all** of the selected activities have completed.

If an Activity has a start condition that isn't met and the **Activity Required to Complete Parent** checkbox is checked on the **Advanced Options** tab, the parent Activity will remain running indefinitely. The parent Activity can only be completed when this Activity completes.

## Completed When Tab #

This tab enables you to indicate when the Activity is considered complete. The dropdown determines whether the Activity is completed when all users have completed the Activity or when the first user has completed the Activity. Additionally, you can set conditions to complete the Activity independently, by clicking the [Click to create condition...](#) link, and using the [Condition Builder](#) to set the desired conditions. These conditions are known as Completed When conditions.

**!** By default, a User Activity completes when all users have completed the Activity manually. If you wish to set a Completed When condition, you must set the Completed When dropdown to "When Any Result Condition Is Met". Additionally, you must set a default result on the Results tab for the Activity to complete the Activity, and/or to prevent the Activity from going into an error state.

The screenshot shows the 'Results' tab of a configuration window. At the top, there are four tabs: 'Activity', 'Participants', 'Results' (selected), and 'Start When'. Below the tabs, the text reads 'List the possible results of this activity.' There is an 'Add Result' button with a plus icon. Below that, a result named 'Done' is shown with a description field. Underneath, there are two sub-tabs: 'Conditions' and 'Options'. The 'Conditions' sub-tab is active, showing two input fields: '% of participants that choose this result' and '# of participants that choose this result', both set to '0'. Below these are two more fields: 'Confirmation prompt for this result' and 'Activity result condition met when', both with a 'Click to create condition ...' link. The 'Show this result button when' field also has a 'Click to create condition ...' link. At the bottom, there is a 'Default Result' checkbox which is checked and highlighted with a red circle.

Another option on the dropdown, "All Users Complete or Result Number Met", enables you to immediately end a task automatically—even if the task requires all users to complete—by specifying certain results to override that and force the Activity to be completed. For example, you may have 3 results on an Activity and normally want all users to complete their task, however you may want one result to immediately end the Activity when a user selects it using the "Number of Users" set to 1.

## Needed When Tab #

In most cases, we expect every Activity in a Process Timeline to run every time it is eligible to do so, and in those cases, we would ignore this tab completely. There are, however, some cases when you do *not* want an Activity to run unless a specific condition is met. This tab enables you to define conditions that specify when the Activity is needed. Once an Activity's dependence requirements have been met and the Activity is eligible to run, the system will evaluate any conditions on this tab. If the conditions aren't met, the Activity will be skipped, and the next Activity in the Process Timeline will be started. If the conditions are met, or there are no conditions configured on this tab, the Activity will run normally. The conditions on this tab are known as Needed When conditions.

Conditions for enabling the Activity can be configured using the [Condition Builder](#) by clicking the [Click to create condition...](#) link.

## Due Date Tab #

Each Activity in the Process Timeline can have unique time limit/due date options. The due date can be calculated in three ways.

1. You can use the dropdown control for the **Due Date is** property. This control will use the duration you specified on the **Activity** tab to calculate the due date from either the actual or the configured start of the Activity.
2. The **Set due date from form field** field enables the determination of the due date from a form field, irrespective of the duration set on the Activity tab.
3. The **Set due date from system variable** field enables the determination of the due date from a system variable, also irrespective of the duration set on the Activity tab.

## Due Date Is

This property specifies how to calculate a due date for this Activity. The Activity due date is optional. If an Activity doesn't complete by the due date specified it can automatically transition to the next Activity in the Process Timeline. The due date can also be used to prioritize items in a user's [Task List](#).

Option	Result
No Calculated Due Date	A Due Date isn't calculated.
Calculated from Actual Activity Start	The due date is calculated from the date the Activity was actually started.
Calculated from Configured Activity Start	The due date is calculated from the date the Activity was configured to start in the Timeline Definition, rather than when it actually started.
Calculated from minimum of Actual or Configured Activity Start	The due date is calculated from either the date the Activity was configured to start in the Timeline Definition, or when it actually started, whichever is first.

### Set Due Date from Form Field

Enables you to specify a due date using a [Date Picker](#) control on the Form used for the process.

### Set Due Date from system variable

The due date will be set from the specified system variable.

For Process Director v6.0 and higher, all subtasks, if any, can be provided with individual due dates. In previous versions, all subtasks shared the same due date, i.e., the Due Date of the Timeline Activity. For Process Director v6.0, this is still the default behavior. However, using this property in conjunction with a Business Rule that specifies different due dates for different subtasks enables you to use a Business Rule System Variable to return different due dates for each running subtask.

For example, you might have a Business Rule named [SubtaskDueDates](#) that returns the first value matching condition, with the values set to dates that are a specified number of days after the form was submitted:

Value	Condition
{FORM_SUBMIT_DATE, days=5}	({SUB_TASK_NAME}=Task1)
{FORM_SUBMIT_DATE, days=8}	({SUB_TASK_NAME}=Task2)
{FORM_SUBMIT_DATE, days=10}	None

The [Set Due Date from system variable](#) property could then be set to `{RULE:SubtaskDueDates}` to return the appropriate due date for each subtask (or for no assigned subtask).

In this scenario, if the running subtask is "Task1" the due date for the subtask will be set to five days after the Form was submitted, while the due date for "Task2" will be set to eight days after Form submission. If a subtask is not assigned to the user, the due date will be 10 days after Form submission. This enables the same Timeline Activity to have different due dates for different users.

### If Time Limit Expires

If the time limit (due date) passes before this Activity is complete these are the actions that can be taken. This dropdown will display different items in the [User Activity type](#) and the [Wait Activity type](#).

**i** Process Director invokes all time related events like due dates *only* when Activity is present on the system (e.g. a login occurs, or the Activity Check page runs). Please refer to the [Activity checking topic](#) in the System Administrator's Guide.

## Notifications Tab #

The screenshot shows a configuration window for an activity named 'Activity 1'. The 'Notifications' tab is active, displaying a checked checkbox for 'Notify participants when activity starts'. Below this, there is a 'Using email template' field with a dropdown arrow. An 'Add Notification' button with a green plus icon is present. At the bottom of the window are 'Close' and 'Cancel' buttons.

Each Activity in the Process Timeline can have unique notification options. You can send single notifications when a specified event occurs, or you can send recurring notifications on a specified schedule. A default schedule is built into Process Director, but you can build a custom schedule in the Custom Variables file via the [Project Reminder Times](#) Custom Variable. Process Director does not track the number of reminder or recurring messages that have been sent.

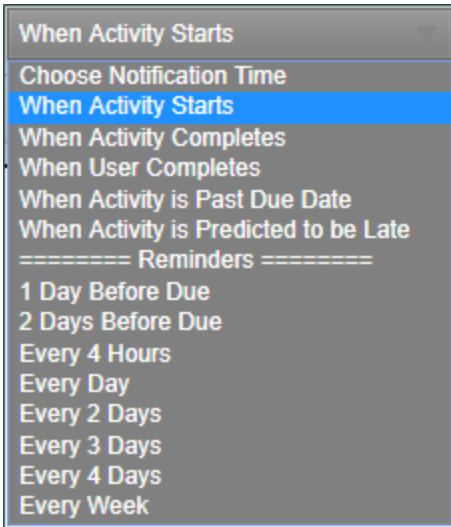
### Notify Participants When Activity Starts

This setting is enabled by default, and the Process Timeline will send a notification to all users that have been assigned from the **Participants** tab.

### Add Notification (Button)

This button enables you to add and select a user or users who should be sent a notification by this Timeline Activity, other than the participants configured for the Activity (Participants are notified automatically via the **Notify Participants...** property above). Once you have made your selection you'll have to select from the **Choose Notification Time** property.

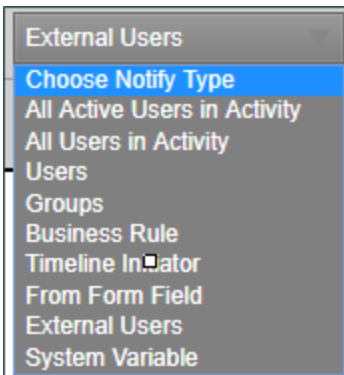
## Choose Notification Time



This property enables you to specify when the notification should be sent to the additional user. Your options are:

- **When Activity Starts**
- **When Activity Completes**
- **When User completes:** You can choose specific users or groups whom you wish to complete the task prior to notification.
- **When Activity is past due date**
- **When Activity is predicted to be late:** Since Timelines can predict when activities will run late, this notification will be sent any time Process Director makes such a prediction.
- **Reminders:** You may also choose various reminder times, to send notifications a varying number of days before or after a task is due. Sending these reminders are useful for creating escalation notifications, or notifying managers that tasks are slipping past due dates.

## Notify Type



This property enables you to specify the notification type that should be sent. Please note that notifications for participants in the Activity are intended for notifications you'd like to send **in addition** to the



task assignment notification that is automatically sent via the [Notify Participants...](#) property. Your options are:

**All Active Users in Step:** Send a notification to all users who have yet to complete a user step.

**All Users in Step:** Notify all the users involved in this User step.

**Users:** Specific users to notify.

**Groups:** Notify all users in a specific group.

**Business Rule Value:** Notify the user result of a Business Rule.

**Timeline Initiator:** Notify the initiator of this Timeline instance.

**From Form Field:** Notify a User specified by a field on the container Form.

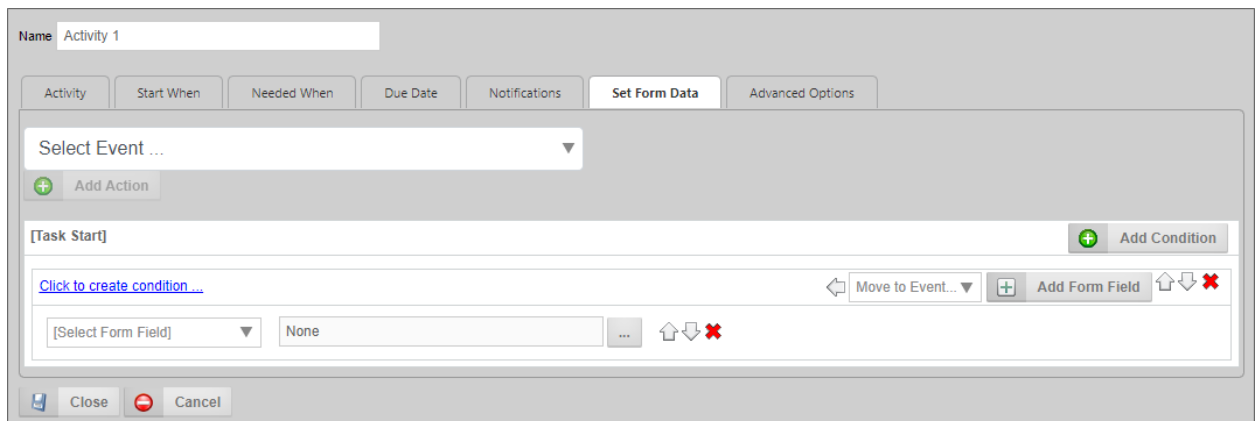
**External Users:** Send a notification to an email address you can specify in the provided text box.

### Using Email Template

This is an optional email template to use when sending email notifications to users in this Activity. If this isn't specified, the default email template for the Process Timeline will be used. If a default email template isn't specified, the notification will use the email template that is shipped with Process Director. For more information on email templates refer to the section named [Email Templates](#) in this guide.

**i** Process Director invokes all time related events like sheduled notifications *only* when Activity is present on the system (e.g. a login occurs, or the Activity Check page runs). Please refer to the [Activity checking topic](#) in the System Administrator's Guide.

### Set Form Data Tab #



This tab only appears on Timeline activities when you have used the [Set Form Data section](#) of the Timeline definition to create a Set Form Data action. Once you have done so, this tab will appear on all Timeline Activity types, enabling you to configure Set Form Data actions on any Activity. If you have not configured at least one Set Form Data action on the Timeline definition, this tab won't appear for any Timeline Activity.

This tab is used to set data in the Process Timeline's default Form when the Activity begins or ends.

## Advanced Options Tab #

Depending on the Activity Type, several different options will be available on this tab. In most cases, Timeline activities have a single property on this tab, **Activity is required to complete parent**. This property is enabled by default, and requires this Activity to complete before its parent Activity, if any can complete. Occasionally, and Activity that isn't part of the critical path of the process might have this property unchecked, so that a lengthy Activity doesn't hold up the rest of the Process Timeline.

Please see the topics for each individual Activity Type for more detailed information about this tab's settings.

## Activity Types

There are eleven different Activity Types, each of which performs different functions, and which has its own topic that can be accessed from the Table of Contents on the upper right portion of the page.

### User

This Activity Type assigns a task to a user or users, which must be completed to end the task. In most cases, an User Activity provides the user with a set of result buttons the user must select to both complete the Activity and set its result value. Please refer to the [User Activity topic](#) for more information about this Activity Type's unique settings.

### Notify

This Activity Type sends email notifications to users who aren't participants in the process. While the User Activity Type notifies Activity participants of their task assignments by default, you may wish to send additional notifications to non-participants, such as managers, or the person who initiated the process. Please refer to the [Notify Activity topic](#) for more information about this Activity Type's unique settings.

### Process

This Activity Type invokes a different process that will run as a separate, synchronous subprocess. When the subprocess completes, the Process Activity that called it will complete, and the parent timeline will continue. While the subprocess runs, the parent timeline will pause on the Process Activity until it completes. Please see the topic on [subprocesses](#) for more detailed information about how they work in Process Director. Please refer to the [Process Activity topic](#) for more information about this Activity Type's unique settings.

### Script

This Activity Type enables you to invoke a custom script. This option is only applicable to customers who are licensed to use the SDK component. Please refer to the [Script Activity topic](#) for more information about this Activity Type's unique settings.

### Custom Task

This Activity Type invokes a Custom Task to run when the Activity starts. Please refer to the [Custom Task Activity topic](#) for more information about this Activity Type's unique settings.

### Form Actions

This Activity Type enables you to manipulate the Form used for the process, such as specifying a different form to set as the default Form for the process. Please refer to the [Form Actions Activity topic](#) for more information about this Activity Type's unique settings.

## Branch

This Activity Type enables you to change the operation of the Process Timeline to invoke a different Activity than the one which would normally run, usually when a specific condition applies. Please refer to the [Branch Activity topic](#) for more information about this Activity Type's unique settings.

## Parent

This Activity Type performs two functions. First, it creates a container for grouping related activities, enabling more detailed organization of the Process Timeline. Second, it enables you to create repeating loops for all or part of a Process Timeline. Please refer to the [Parent Activity topic](#) for more information about this Activity Type's unique settings.

## End Process

This Activity Type enables you to conditionally end a process, even if the process hasn't completed. Please refer to the [End Process Activity topic](#) for more information about this Activity Type's unique settings.

## Wait

This Activity Type enables you to pause a Process Timeline. You can pause for a specified length of time, or until a specified condition has been met. Please refer to the [Wait Activity topic](#) for more information about this Activity Type's unique settings.

## Case

This Activity Type enables you to manipulate the Case instance that is associated with the Process Timeline. Please refer to the [Case Activity topic](#) for more information about this Activity Type's unique settings.

## User Activity

The user Activity is used to create an actionable item assigned to a user. If a user Activity has been assigned to a user, a task will appear in the user's [Task List](#). The user must complete the task before it can be removed from the [Task List](#). A user who is assigned a task will be automatically given permission to any object in the Process Timeline package so they may complete the task. Examples of this task can be an approval task or update task.

When a Process Timeline Activity is assigned to a user with an invalid UID or user ID, Process Director will immediately stop the process, and place the Process Timeline Activity into an error state. This is also true for anonymous user assignments if the email address isn't a valid format (Process Director can't validate the email address itself, only the format).



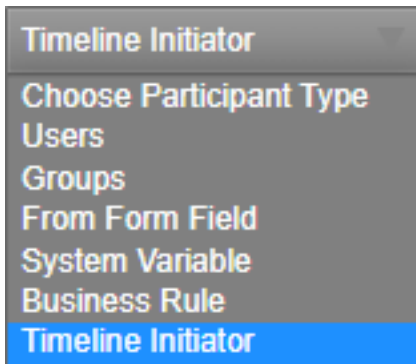
If your process includes tasks for unauthenticated users, please refer to the documentation on [Anonymous Users](#) for special concerns that may apply to some settings.

In addition to the common properties tabs that appear for all Timeline activities, the configuration settings below are unique to this Activity Type.

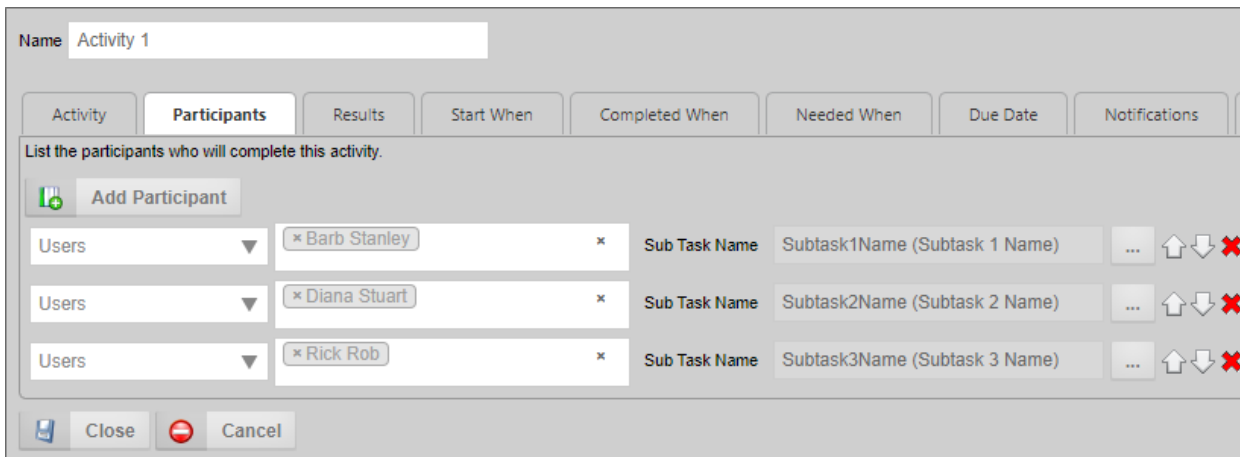
## Participants Tab #



Determines which users are participants of the user task. Users can be added via the **Add Participant** button, and the participants can be specified via the **Participant Type** dropdown.




When multiple users are assigned to a task, a Subtask can be created for each user. You must have a form field that specifies the name of each desired subtask.



In this example, each user is assigned a Subtask Name from a different form field, i.e., **Subtask1Name**, **Subtask2Name**, and **Subtask3Name**. When the users are assigned their task, Both the Task Name (which is the Activity's **Name**) and **Sub Task Name** will be assigned to them, so that you can track both the task and subtask for each user as they are completed.

For Process Director v 6.0.100 and higher, each subtask can have a unique due date, assigned via a System Variable. Unlike prior versions, where a single due date was assigned to both the Timeline Activity and all of its included subtasks, you can now configure a Business Rule to return a unique Due Date for each subtask, based on any evaluable condition, such as the **Sub Task Name**.

Business Rule    Icon 


SubtaskDueDatesBR

---

## VALUES

---

▲ Parameters













 Add Parameter


---

Returns

First Value Matching Condition ▼

This returns the value for the first matching condition only. If no conditions are met, an empty value is returned.

Values	Condition
{FORM_SUBMIT_DATE, Days=3} 	(({SUB_TASK_NAME}) = Subtask1)   
{FORM_SUBMIT_DATE, Days=5} 	(({SUB_TASK_NAME}) = Subtask2)   
{FORM_SUBMIT_DATE, Days=8} 	(({SUB_TASK_NAME}) = Subtask3)   

 Add Value

In this example, the Business Rule, named [SubtaskDueDatesBR](#), will return one of three different due dates, based on the name of the Subtask, e.g. "Subtask1", "Subtask2", or "Subtask3". On the **Due Date** tab of the Timeline Activity, this Business Rule is referenced in the **Set due date from system Variable** property to provide the appropriate due dates for each subtask.

The screenshot shows a configuration window for 'Activity 1'. At the top, the name 'Activity 1' is entered. Below the name are several tabs: 'Activity', 'Participants', 'Results', 'Start When', 'Completed When', and 'Needed'. The 'Due date is' dropdown is set to 'Calculated from the Configured Activity Start'. Below this, there are fields for 'Set due date from form field' (empty), 'Set due date from system variable' (containing '{RULE:SubtaskDueDateBR}'), and 'If time limit expires' (set to 'Take no action'). There is also a 'Machine Learner to set duration' field. At the bottom, there are 'Close' and 'Cancel' buttons.

Since each subtask is assigned to as different user, Process Director will pass the **Sub Task Name** for that user to the Business Rule when the subtask is assigned, which will return the unique due date for each sub-task participant.

### **Results Tab #**

The **Results** Tab appears when either the User or Wait Activity Types are selected. The Process Activity also has a result, which is generated automatically, and is explained in the "Results of a Process Activity" section below.

The screenshot shows the 'Results' tab of the 'Activity 1' configuration window. The 'Add Result' button is visible. The 'Activity Result' section is expanded, showing fields for 'Result Name' (containing 'Activity Result'), 'Description', and 'Conditions'. The 'Conditions' section has two input fields for '% of participants that choose this result' and '# of participants that choose this result', both set to '0'. There are also links for 'Click to create condition ...' and a 'Default Result' checkbox. A 'Remove Result' button is also present. At the bottom, there are 'Close' and 'Cancel' buttons.

A user Activity in the Process Timeline can have results assigned to it. Multiple results can be added to an Activity by clicking on the Add Result Button. Each result will have its own properties, which are presented in a tabbed interface.

The results will be used as the Activity result for the task, which will appear in the [Routing Slip](#) and in the Timeline Administration screen.

### Result Name

The name of the result, which will appear on all Forms, and as the completed result in the Timeline instance, if the result is selected by a user.

### Description

A brief text description of the result.

## Conditions Tab #

This is one of the two tabs displayed for each result's configuration settings. The [Conditions](#) Tab includes the following controls.

**Activity Result**

Result Name

Description

**Conditions** | Options

% of participants that choose this result

# of participants that choose this result

Confirmation prompt for this result

Activity result condition met when [Click to create condition ...](#)

Show this result button when [Click to create condition ...](#)

Default Result

### % of participants that choose this result

If the specified percentage of users select this result, the task will be considered complete, with this result recorded as the Activity result. For example, if you wish the task to be concluded when 75% of users select this result, then the value you'd place in the text box would be "75".

### # of participants that choose this result


If the specified number of users select this result, the task will be considered complete, with this result recorded as the Activity result. For example, if you wish the task to be concluded when 3 users select this result, then the value you'd place in the text box would be "3".

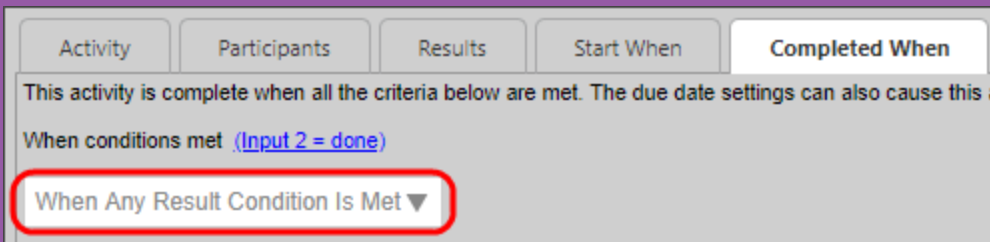
### Confirmation prompt for this result

If a user selects this result, the specified confirmation prompt will be displayed to the user, requiring the user confirm this selection before it is recorded.

### Activity result condition met when

This enables you to select a condition, or conditions, which, when met, will complete this task with the result recorded as the Activity result.

 When setting an Activity result condition, you must set the Completed When dropdown to "When Any Result Condition Is Met" to complete the Activity on the Completed When tab of the Activity definition.



### Show this result button when

This enables you to select a condition, or conditions, which, when met, will display this result option to the user. If the condition(s) aren't met, the result button won't be displayed to users.

### Default Result

This is a check box that, when checked, designates the result as the default result for the Timeline Activity. When setting a Completed When condition, you **must** specify a default result, or the Activity will either not end as expected, or will go into an error state.

### Options Tab

This is one of the two tabs displayed for each results configuration settings. The Options Tab includes the following controls.



▲ Activity Result

Result Name

Description

Conditions

**Options**

Button Order on Form

Button Icon on Form

Button Background Color  ▼

Button Text Color  ▼

Optional list of strings for email complete

Signature Comments Required     Allow this result to be completed from the users Task List     Skip Validation on Form when Selected

### Button order on Form

This is a numeric value that determines the order in which the Activity result button will appear on a Form. You may order the Activity results in any order you desire. If you don't order the buttons manually, they'll be displayed in the order in which the result was created. In older versions of Process Director, Process Timeline Activity results were specified using a comma-separated list. The display of the buttons on the form for each result followed the order in which they were listed. Now, when migrating from an older release to this one, the display order of the results will be automatically preserved. To modify the order, adjust the "Button Order on Form" property of each result in the Activity Results tab.

### Button icon on Form

Clicking on the icon allows you to select a custom icon from a list. The selected icon will appear on the Activity result's button on the Form.

### Button Background Color

The setting enables you to select a background color for the result button that will appear on the form.

### Button Text Color

The setting enables you to select a text color for the result button that will appear on the form.

### Optional list of strings for email complete

This text box will accept a comma-separated list of string values. For tasks that can be completed by email, a user can complete the task by replying to the task assignment email with one of the strings in the list you enter into this control. For example, the result displayed above is the "Reject" result. You might put the following list of responses into this control: "Reject, No, Disapprove, Rejected, Not Approved". If a user replies to the task assignment email with "No"—or any of the other possible responses in the list—as the body of the email message, the Reject result will be recorded as the Activity result.

### Signature Comments Required

If this option is selected, then, if the user selects this result, then the Results Comments control in the Form can't be empty.

## Allow this result to be completed from the user's Task List

If this option is selected, then users can complete this task from their [Task List](#) without opening the Form for the process.

 This feature requires the use of a custom Task List that includes a column to display the results.

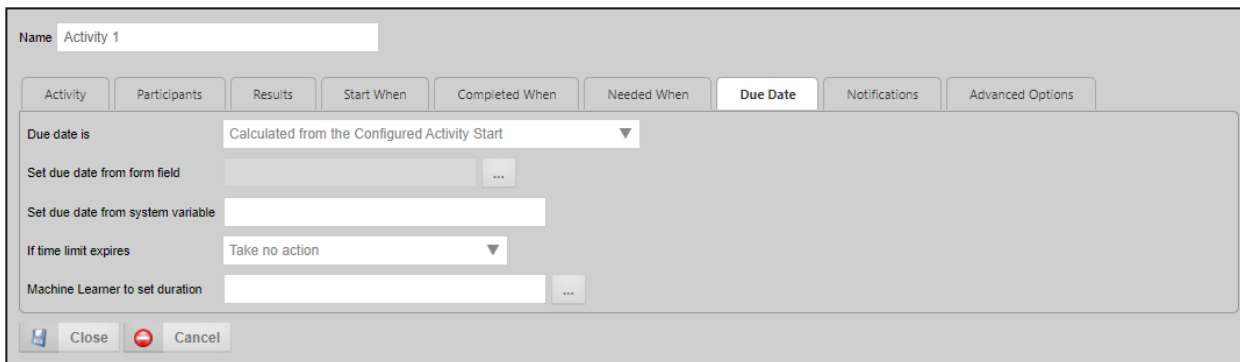
When the user completes a task from the [Task List](#), Process Director will display a small popup that indicates the task completion is in progress. Additionally, if the result selected is configured to require completion comments, Process Director will prompt the user for the comments before completing that task.

The size of the prompt and confirmation dialogs won't usually need to be changed, however, if, for some reason the size does need to be changed, there are custom variables available to change the width and height of the dialogs: `nTaskCompleteDialogWidth`, `nTaskCompleteDialogHeight`, `nTaskCompletePromptDialogWidth`, and `nTaskCompletePromptDialogHeight`. Documentation for these four custom variables can be found in the [Miscellaneous Custom Variables](#) topic of the Developer's Guide.

## Skip validation on form when Selected

Check this box to specify that a given Activity result will cause validation to be skipped. The feature is especially useful for steps/activities that include an option for the user to abandon or reject a form, which the user should be able to accomplish without, for example, filling in all required fields.

## Due Date Tab #



The standard options for the Due Date tab are described in the [Due Date Tab section](#) of the Timeline Activities topic. The User Activity Type also has an additional option specific to the User Activity Type.

## If time limit expires

A dropdown that specifies the actions to take when the Due Date expires. The following options are available.

Option	Result
Take no action	Lets the Activity continue running.

Option	Result
Cancel This Activity	Cancel the Activity and automatically advance to the next Activity in the Process Timeline.
Add These Users from Business Rule	Add users identified in the Business Rule.
Replace existing users with those in Rule	Add the user from the Business Rule and cancel any running users.
Cancel the entire Timeline	Cancel the Process Timeline.

### Advanced Options Tab #

This tab provides many additional settings for controlling task assignment, completion, Activity restarts, and more, as described below.

#### Is Activity required to complete parent

Checking this option won't allow a parent Activity to complete until this Activity is complete. This is the default option. This usually applies to an Activity that is organized under a Parent Activity Type. If this Activity doesn't complete, then the Parent Activity can't complete.

#### Set Activity result to a list of all user results

By default, the Result of a user Activity that is completed by multiple users will be the single result selected by most users. For instance, if two users pick Reject and one picks Accept, the Activity result is set to "Reject". In the case of a tie, e.g., if one user selects Accept and one Reject, the Result is set to "Accept, Reject". Selecting this option on a user Activity causes the Result to be set to the comma-separated union of all Results selected by users. So, in the first example above, the Result would be set to "Reject, Reject,

Accept" or some similar result. The order of the results in the results list isn't guaranteed.

### Re-authenticate Users

Forces users to re-authenticate prior to completing a task. For example, when this option is selected, built-in users will be presented with a login screen, requiring them to log in prior to completing the task, even if they are currently logged in to Process Director. This is an extra security precaution.

### Prompt for Signature Comments

Checking this option will cause a prompt to appear to users to enter signature comments when completing the task.

### Allow Task Sharing

Checking this option will enable task sharing through the [Shared Delegation](#) feature. If this option isn't checked, this task won't be available for Shared Delegation.

### Allow task to be completed from email

Checking this option will enable users to complete the task by replying to an email without opening the Form associated with the Activity or Process Timeline. Process Director can monitor an email inbox, import and parse the emails, and search for specific text in the email to determine the result. If a possible result for an Activity is "Approved", a user can reply with the word "Approved" in the body text of the reply, and Process Director will complete the Activity with the result of "Approve".

### Enable email invitations

Checking this option will enable users to invite others to perform the user task by forwarding the task assignment email to them. The task assignee can forward the email to a desired alternate, for example, and when that alternate clicks on a completion link in the email, the task will be completed with the appropriate result, and with an annotation that the task was completed by the users invited by the original task assignee.

### Require at least one user to be assigned and complete this task

If this option is checked, at least one user will be required to complete the task, or the task can't complete. Depending on how the task assignment is configured, it's possible that no users will be assigned to the task in certain scenarios. For instance, if a task restarts after all users have completed their assignment, and the assignment is set to "Start users that did not complete previously" there will be no available user to which the task can be re-assigned. Checking this option will assign the task to one of the previously completed users.



If an administrator cancels a user in an Activity where this option is set, the Activity will go into an error state. The administrator will have to cancel the Activity to cause it to move on.


### Assign task to first user to accept

Checking this option will, when the task is assigned to multiple users, allow the first user to accept the task to be assigned to complete the task. When the task begins, all of the configured users will receive a

**Task List** item that requests them to accept the task. When opening the Form, an **Accept Task** button will be displayed at the bottom of the Form. When any of the configured users clicks the **Accept Task** button, Process Director will remove task acceptance request from the **Task List** of all the other configured users, and the task will be assigned solely to the user who accepted the task.

### Allow anonymous users to be assigned tasks by email address

Checking this option will enable anonymous users to complete a task based on the user's email address. This is a required option when using the [Email Anonymous Task List](#) system variable. In the vast majority of use cases, however, the email address will be taken from a Form field identified on the Participants tab.

 **Access to Process Director for unauthenticated or anonymous users is enabled for Cloud and Subscription licenses. On-premise licenses using the legacy Tired licensing model require an additional, optionally licensed component.**

### Switch the user in Case Folder View when opening this task

If this task is part of a Case Management application, checking the property will automatically display the Case Folder view when the user opens the task.

### Assign users to this task

This option enables you to select how wish to assign a task when multiple users are associated with the same task. You have the following options:

OPTION	DESCRIPTION
All in parallel	All users will be assigned the task at the same time, and will complete their actions at the same time as the other users.
All in series	Each user will be assigned the task individually, and each user must complete the assigned action before the task is assigned to the next user. The process will continue until all users have completed their actions.
One (Round Robin)	Only one user will be assigned the task each time an instance of the Timeline is run. Each time an instance is run, a different user will be assigned the task until all users have been assigned an instance of the process. This will divide the assignments equally among all assigned users.
One (fewest tasks in this process)	Only one user will be assigned the task each time an instance of the Timeline is run. Process Director will assign the task to the user who has the fewest number of active tasks pending in this process. This method assigns the task to the user who has the smallest workload in this process.
One (fewest tasks overall)	Only one user will be assigned the task each time an instance of the Timeline is run. Process Director will assign the task to the user who has the fewest number of active tasks in their <a href="#">Task List</a> . This method assigns the task to the user who has the smallest overall workload.

### After a user completes this task

This option determines how a user's tasks are displayed when a user is assigned two sequential tasks in a process. The default in Process Director is to automatically show the user's next task if it is in the current process.

For example, if a user completes a task in a Form, and is also assigned the next task in the sequence, once the user clicks the OK or other task completion button on the Form, the Form won't close, but will simply redisplay itself to allow the user to complete the next task in the sequence. For various reasons, you may wish to override this behavior, in which case the Form will close once the first task has been completed, and the user will have to re-open the Form to complete the next task in the sequence.

You have the following options for configuring the behavior of Process Director:

OPTION	DESCRIPTION
Automatically show user's next task if it is in this process	This is the default behavior, and will automatically redisplay the Form once the user completes the first task in the sequence.
Automatically show user's next task if it is for this step in a different process	This selection will automatically show the user's next task if the user is assigned a task in another process. For instance, if the current task starts a sub-process, and the user is assigned a task in the subprocess, the subprocess task will automatically be displayed.
Do not show the user's next task	Once a user completes the task, the Form will close, and the user must reopen the Form from the <a href="#">Task List</a> to begin performing the next task.
Automatically show user's next task if it is for any step for any related sub- process.	This selection will automatically show the user's next task if the user is assigned a task in either the current process or related sub-process.

## Restart Users

This option determines how participants are assigned to the Activity when it must be restarted, which is particularly relevant when an implementer changes the participants assigned to an Activity in the Process Timeline definition. An existing instance may have been created when different participants were configured to participate in the task. This option enables you to determine how to handle user assignments when the configured users have changed.

It is also important to note that the term "restart" is used for both:

- Cases in which a running task is restarted while the task is still running, and
- Cases in which a task has already been completed, but must be re-run, as in an iterative process, or as a result of a change to some condition.

In either case, the [Restart Users](#) options you select will be applied.

The following options are available:

OPTION	DESCRIPTION
Start only configured task participants	This option will assign the task to the currently configured users, irrespective of which users may have been previously assigned to the Activity.
Start configured and users all previously run in this task.	By choosing this option, you ensure that both the currently configured users, as well as any users who were previously assigned the task, are assigned the task.
Start only users that previously completed this task	This option will assign the task only to users who completed the task when it was previously run for this instance. Those users will be re-assigned the task irrespective of whoever is currently configured for task assignment in the Process Timeline definition.
Start from last completed user	If multiple users are assigned to the Activity instance, this option will re-assign the Activity to the last user who completed it when it was initially run.
Start users that did not complete previously	If multiple users are assigned to the Activity instance, this option will assign the task to users who did not previously complete it. For instance, if the Activity uses a round-robin assignment, the Activity will be assigned to one of the users who were not assigned it on the initial run.

### Use Form in this Activity

This will allow you to specify a Form to use in this Activity. This controls the form displayed when a user opens their [Task List](#).

### Owner that can share tasks with any of the running users

This property enables you to specify a system variable that can return users who will be given shared task permission for this specific Activity. This enables dynamic task sharing per each Timeline Activity, as opposed to the normal task sharing, which is set universally.

### Other Activity Types

To view the documentation for other Activity Types, you can navigate to them using the Table of Contents displayed in the upper right corner of the page, or by using one of the links below.

**Notify:** This Activity Type sends email notifications to users who aren't participants in the process.

**Process:** This Activity Type invokes a different process that will run as a separate, synchronous sub-process.

**Script:** This Activity Type enables you to invoke a custom script.

**Custom Task:** This Activity Type invokes a Custom Task to run when the Activity starts.

**Form Actions:** This Activity Type enables you to manipulate the Form used for the process.

**Branch:** This Activity Type enables you to change the operation of the Process Timeline to invoke a specified Activity.

**Parent:** This Activity Type serves as a container for other activities and to create a looping segment in a Process Timeline.

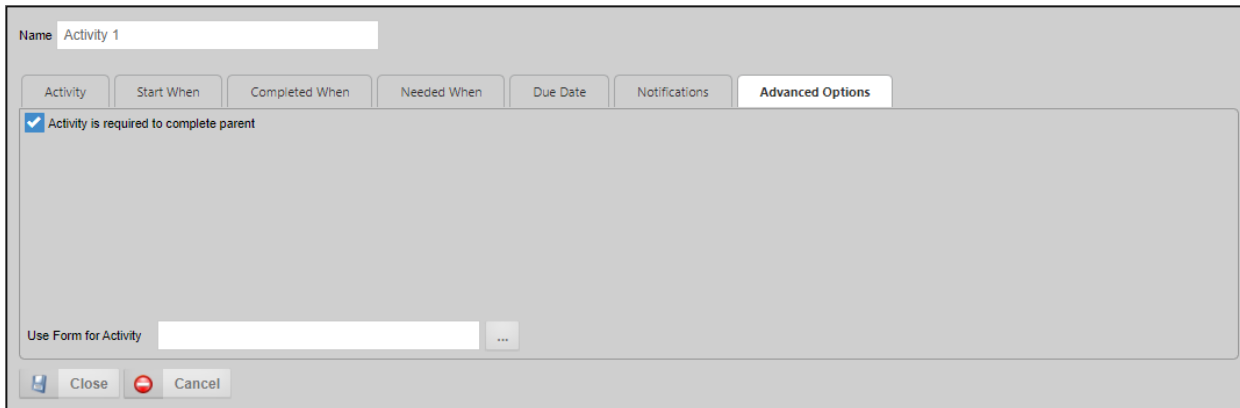
**End Process:** This Activity Type enables you to conditionally end a process.

**Wait:** This Activity Type enables you to pause a Process Timeline.

**Case:** This Activity Type enables you to manipulate the Case instance that is associated with the Process Timeline.

## Notify Activity

The notify task allows you to notify users or groups during a process. Most of the Notify Task options are configured via the common configuration options for an Activity, except for the options in the **Advanced Options** tab.



The screenshot shows a configuration window for an activity. At the top, there is a 'Name' field containing 'Activity 1'. Below this are several tabs: 'Activity', 'Start When', 'Completed When', 'Needed When', 'Due Date', 'Notifications', and 'Advanced Options'. The 'Advanced Options' tab is currently selected. Under this tab, there is a checkbox labeled 'Activity is required to complete parent' which is checked. At the bottom of the window, there is a 'Use Form for Activity' field with a dropdown arrow, and two buttons: 'Close' and 'Cancel'.

There are two advanced options for Notification activities.

### Is Activity required to Complete Parent

This option defaults to true, so that the Parent Activity, if any, can't be completed until this Activity is complete. This is, by far, the most common relationship between parent/child activities.

### Use Form for Activity

This option enables you to specify a form instance for a particular notify step/Activity. This feature is useful when configuring email templates to include information from a form instance other than the default form for that process.

### Other Activity Types

To view the documentation for other Activity Types, you can navigate to them using the Table of Contents displayed in the upper right corner of the page, or by using one of the links below.

**User:** This Activity Type assigns a task to a user or users, which must be completed to end the task.

**Process:** This Activity Type invokes a different process that will run as a separate, synchronous sub-process.

**Script:** This Activity Type enables you to invoke a custom script.

**Custom Task:** This Activity Type invokes a Custom Task to run when the Activity starts.

**Form Actions:** This Activity Type enables you to manipulate the Form used for the process.



**Branch:** This Activity Type enables you to change the operation of the Process Timeline to invoke a specified Activity.

**Parent:** This Activity Type serves as a container for other activities and to create a looping segment in a Process Timeline.

**End Process:** This Activity Type enables you to conditionally end a process.

**Wait:** This Activity Type enables you to pause a Process Timeline.

**Case:** This Activity Type enables you to manipulate the Case instance that is associated with the Process Timeline.

## Process Activity

The Process Activity enables you to specify a Process Timeline to run as a subprocess.

When a Process Activity runs, the process invoked will run synchronously with the parent process. The Process Activity will continue to run until the subprocess completes. Once the subprocess ends, the Process Activity will complete, and the parent Timeline will start the next eligible Activity. For more information on subprocesses, please see the [Subprocesses section](#) of the Process Timelines topic.

In addition to the common properties tabs that appear for all Timeline activities, the configuration settings below are unique to this Activity Type.

### Process Tab

Determines the process that the Activity starts. It also enables objects in the child process to be copied to the parent and vice versa. The objects copied can be limited to a specified group.

### Start Process When Activity Begins

You may use the [Object Picker](#) to select a process that will automatically start when the Activity starts. The selected process will run as a child process of the current process, i.e., the process containing this Activity.

### Copy Objects in Timeline Package to Parent

Checking this option will copy all objects in the child process to the current process.

### Only from this group

You may enter the Group Name. Only objects from the specified group will be copied from the child process.

### Copy Objects in Timeline Package to Child Process

Checking this option will copy all objects in the current process to the child process.

### Only from this group

You may enter the Group Name. Only objects from the specified group will be copied from the current process.

### Other Activity Types

To view the documentation for other Activity Types, you can navigate to them using the Table of Contents displayed in the upper right corner of the page, or by using one of the links below.

**User:** This Activity Type assigns a task to a user or users, which must be completed to end the task.

**Notify:** This Activity Type sends email notifications to users who aren't participants in the process.

**Script:** This Activity Type enables you to invoke a custom script.

**Custom Task:** This Activity Type invokes a Custom Task to run when the Activity starts.

**Form Actions:** This Activity Type enables you to manipulate the Form used for the process.

**Branch:** This Activity Type enables you to change the operation of the Process Timeline to invoke a specified Activity.

**Parent:** This Activity Type serves as a container for other activities and to create a looping segment in a Process Timeline.

**End Process:** This Activity Type enables you to conditionally end a process.

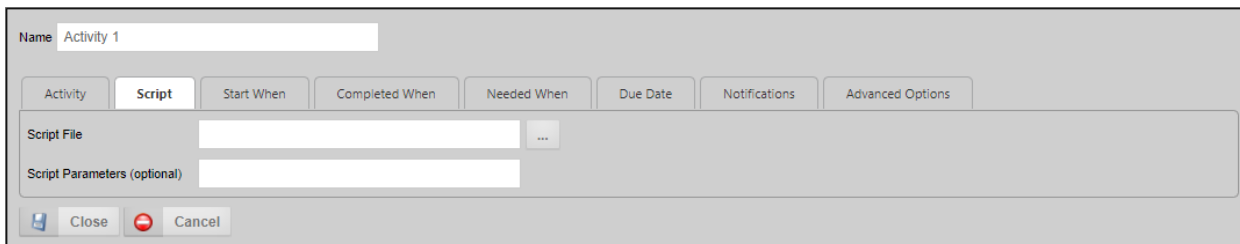
**Wait:** This Activity Type enables you to pause a Process Timeline.

**Case:** This Activity Type enables you to manipulate the Case instance that is associated with the Process Timeline.

### Script Activity

The Script task supports the ability to call custom script functions. This Activity Type is only available for installations that are licensed for the SDK component. In addition to the common properties tabs that appear for all Timeline activities, the configuration settings below are unique to this Activity Type.

#### Script Tab



The screenshot shows a configuration dialog for a Script Activity. At the top, there is a text field labeled 'Name' containing 'Activity 1'. Below this is a row of seven tabs: 'Activity', 'Script' (which is selected), 'Start When', 'Completed When', 'Needed When', 'Due Date', 'Notifications', and 'Advanced Options'. The 'Script' tab is active, showing a 'Script File' field with a text input and a file picker icon (three dots), and a 'Script Parameters (optional)' field with a text input. At the bottom of the dialog are three buttons: 'Close', 'Cancel', and a red 'X' button.

The **Script** tab determines the script to be run. The script file is determined via contest list object picker. The script parameters are passed to the script as a string.

#### Script File

You may use the **Object Picker** to select a script that will automatically start when the Activity starts.

#### Script Parameters (optional)

You may enter a parameter string to pass to the selected script.

### *Advanced Options Tab*

The **Advanced Options** tab for this Activity Type has two unique properties.

#### **Run this Activity in the background**

For scripts that may take some time to process, you may wish to have the process run in the background. Please see the [Background Operations section](#) of the Process Timelines topic for more information about what this option entails.

#### **Use Form For Activity**

In Timeline, if parent Activity has "Use Form for Activity" specified with particular form, then all child activities will default to the same form. This property enables a Script Activity in a Timeline to specify a different Form to use a current form for the Activity. If this property isn't configured, the Process Director will check the parent Timeline Activity for a configured Form and, if set, it will use that Form. Otherwise, Process Director will revert to passing the script to the current form instance that is associated with the running process.

#### ***Other Activity Types***

To view the documentation for other Activity Types, you can navigate to them using the Table of Contents displayed in the upper right corner of the page, or by using one of the links below.

**User:** This Activity Type assigns a task to a user or users, which must be completed to end the task.

**Notify:** This Activity Type sends email notifications to users who aren't participants in the process.

**Process:** This Activity Type invokes a different process that will run as a separate, synchronous sub-process.

**Custom Task:** This Activity Type invokes a Custom Task to run when the Activity starts.

**Form Actions:** This Activity Type enables you to manipulate the Form used for the process.

**Branch:** This Activity Type enables you to change the operation of the Process Timeline to invoke a specified Activity.

**Parent:** This Activity Type serves as a container for other activities and to create a looping segment in a Process Timeline.

**End Process:** This Activity Type enables you to conditionally end a process.

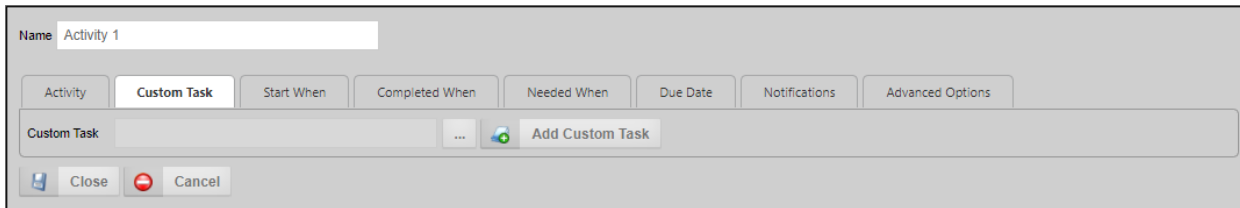
**Wait:** This Activity Type enables you to pause a Process Timeline.

**Case:** This Activity Type enables you to manipulate the Case instance that is associated with the Process Timeline.

## Custom Task Activity

This Activity Type enables you to invoke a Custom Task to run as a Process Timeline Activity. In addition to the common properties tabs that appear for all Timeline activities, the configuration settings below are unique to this Activity Type.

### Custom Task Tab



Specifies the Custom Task(s) to be run for this Activity. To add additional Custom Tasks, click the **Add Custom Task** button.

### Custom Task

You may use the **Object Picker** to select a Custom Task that will automatically run when the Activity starts.

For details on how to configure Custom Tasks, see the [Custom Tasks documentation](#).

### Other Activity Types

To view the documentation for other Activity Types, you can navigate to them using the Table of Contents displayed in the upper right corner of the page, or by using one of the links below.

**User:** This Activity Type assigns a task to a user or users, which must be completed to end the task.

**Notify:** This Activity Type sends email notifications to users who aren't participants in the process.

**Process:** This Activity Type invokes a different process that will run as a separate, synchronous sub-process.

**Script:** This Activity Type enables you to invoke a custom script.

**Form Actions:** This Activity Type enables you to manipulate the Form used for the process.

**Branch:** This Activity Type enables you to change the operation of the Process Timeline to invoke a specified Activity.

**Parent:** This Activity Type serves as a container for other activities and to create a looping segment in a Process Timeline.

**End Process:** This Activity Type enables you to conditionally end a process.

**Wait:** This Activity Type enables you to pause a Process Timeline.

**Case:** This Activity Type enables you to manipulate the Case instance that is associated with the Process Timeline.

## Form Actions Activity

This Activity Type enables you to configure which Form will be the default form within the process, and enables new forms to be attached. In addition to the common properties tabs that appear for all Timeline activities, the configuration settings below are unique to this Activity Type.

### Form Options Tab

### Form Definition


You may use the [Object Picker](#) to select a form definition to be attached to the process.

### Set this as the default Form Instance for the Timeline

Checking this option will make the selected Form the default form for this Timeline Instance.

### Set the form instance version to 1, this will prevent any default data from being applied to form fields

This option will create the Form instance with a version of 1. Normally the version is set to a special value to indicate that the form has never been saved by a user which prevents it from showing up on Knowledge Views, and also applies the default data to the Form fields. Checking this option will cause the form instance to NOT use default data.



You shouldn't use this option unless you know that your application specifically requires this behavior.

### Add new Form Instance

Specifies the conditions under which a new instance of the form definition is created and then attached to the process. The following options are available:

- **Only if the Form doesn't already exist in the package:** A new instance of the Form will be created only if there is no existing instance.
- **Never add a new Form instance:** No new Form instance will be added.
- **Every time the task runs:** Add a new Form instance for every iteration of this task.

### Group Name

Assigns a group name to the added form instance.

### *Other Activity Types*

To view the documentation for other Activity Types, you can navigate to them using the Table of Contents displayed in the upper right corner of the page, or by using one of the links below.

**User:** This Activity Type assigns a task to a user or users, which must be completed to end the task.

**Notify:** This Activity Type sends email notifications to users who aren't participants in the process.

**Process:** This Activity Type invokes a different process that will run as a separate, synchronous sub-process.

**Script:** This Activity Type enables you to invoke a custom script.

**Custom Task:** This Activity Type invokes a Custom Task to run when the Activity starts.

**Branch:** This Activity Type enables you to change the operation of the Process Timeline to invoke a specified Activity.

**Parent:** This Activity Type serves as a container for other activities and to create a looping segment in a Process Timeline.

**End Process:** This Activity Type enables you to conditionally end a process.

**Wait:** This Activity Type enables you to pause a Process Timeline.

**Case:** This Activity Type enables you to manipulate the Case instance that is associated with the Process Timeline.

### **End Process Activity**

This Activity Type enables you to conditionally end a process, even if the process hasn't completed. A Process Timeline ends automatically when all of the activities complete, so this Activity Type isn't normally needed to end a timeline. But there are times when you might wish to specify a set of conditions that, when met, will end the Process Timeline before it would normally complete.

A Process Timeline may only have one End Process Activity.

There are no special configuration options for this Activity. Instead, you'd configure the **Start When** and **Needed When** tabs to configure this Activity's dependencies, and Needed When conditions, respectively.

### *Other Activity Types*

To view the documentation for other Activity Types, you can navigate to them using the Table of Contents displayed in the upper right corner of the page, or by using one of the links below.

**User:** This Activity Type assigns a task to a user or users, which must be completed to end the task.

**Notify:** This Activity Type sends email notifications to users who aren't participants in the process.

**Process:** This Activity Type invokes a different process that will run as a separate, synchronous sub-process.

**Script:** This Activity Type enables you to invoke a custom script.

**Custom Task:** This Activity Type invokes a Custom Task to run when the Activity starts.

**Form Actions:** This Activity Type enables you to manipulate the Form used for the process.

**Branch:** This Activity Type enables you to change the operation of the Process Timeline to invoke a specified Activity.

**Parent:** This Activity Type serves as a container for other activities and to create a looping segment in a Process Timeline.

**Wait:** This Activity Type enables you to pause a Process Timeline.

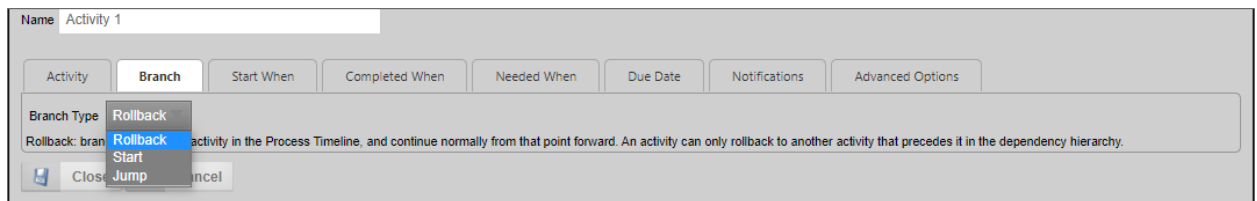
**Case:** This Activity Type enables you to manipulate the Case instance that is associated with the Process Timeline.

## Branch Activity

Enables an Activity to jump or branch to another Activity in the running Process Timeline. For Process Director v3.49 and above, the use of this Activity type should be replaced, in most cases, by the use of the Looping functionality available in the Parent Activity Type.

In addition to the common properties tabs that appear for all Timeline activities, the configuration settings below are unique to this Activity Type.

### Branch Tab



The Branch tab determines where the Activity goes, as specified by the “Branch Type” dropdown. The Branch can either roll back to the Activity on which it depends or start or jump to a new Activity. Jumping to a new Activity skips all the activities in between; they won't be started this process instance. Starting a new Activity allows the steps in between to start as they normally would.

### Branch Type

This dropdown control contains the following configuration options:

- **Rollback:** This will branch back to an earlier Activity and proceed normally from there. This branch can only be configured to roll back to an Activity preceding this Activity on the dependency hierarchy. The Activity must be dependent on the Activity to which this Activity rolls back, or it must be dependent on an Activity dependent on that Activity, and so on. For users of Process Director 5.26 and higher, Timelines can be rolled back to Parent Activity Types.
- **Start:** This will start the defined Activity, complete this branch Activity, and then continue processing normally from this point. Starting another Activity doesn't end the branch that is currently running. The specified Activity will start, and run concurrently with the activities that follow the Branch Activity.
- **Jump:** This will transfer control from this branch Activity to the Activity we want to jump to, any Activity “dependent” on this branch Activity will NOT get run.

### Branch to Activity

This dropdown contains a list of available Activities to which you may branch in the Timeline. The activities that appear in this list will depend on the location of this Activity in the timeline, the **Branch Type** selected, and the activities that are in the critical path of the Process Timeline.

## Other Activity Types

To view the documentation for other Activity Types, you can navigate to them using the Table of Contents displayed in the upper right corner of the page, or by using one of the links below.

**User:** This Activity Type assigns a task to a user or users, which must be completed to end the task.

**Notify:** This Activity Type sends email notifications to users who aren't participants in the process.

**Process:** This Activity Type invokes a different process that will run as a separate, synchronous sub-process.

**Script:** This Activity Type enables you to invoke a custom script.

**Custom Task:** This Activity Type invokes a Custom Task to run when the Activity starts.

**Form Actions:** This Activity Type enables you to manipulate the Form used for the process.

**Parent:** This Activity Type serves as a container for other activities and to create a looping segment in a Process Timeline.

**End Process:** This Activity Type enables you to conditionally end a process.

**Wait:** This Activity Type enables you to pause a Process Timeline.

**Case:** This Activity Type enables you to manipulate the Case instance that is associated with the Process Timeline.

## Parent Activity

Parent activities are parent containers for other activities, which are referred to as "Children" or "child activities".



You shouldn't use any other type of Activity as a Parent Activity.

In Process Director versions 3.45 and higher, additional functionality is available to Parent Activity Types: Results Concatenation and Looping.

In addition to the common properties tabs that appear for all Timeline activities, the configuration settings below are unique to this Activity Type.

### **Results and Result Concatenation**

By default, the result of a parent Activity is set to the result of the child task whose completion resulted in the completion of the parent (this behavior is implemented in Process Director v3.45 and up. In previous versions, the parent never has a result.). Parent activities have the option of setting the result of the Parent Activity to a comma-separated list of all the child results. This list will contain only the results from the immediate child activities of the Parent Activity; it won't use results from any Parent activities that have been created under the top-level Parent. If the parent is configured to loop, only the results from the final iteration will be included. If the parent contains other parent activities, only the results from the direct children of that parent will be included; the results from the children of "subparents" won't be included.



## Looping Tab

Parent activities can also be used to create loops within your Process Timeline. When configured this way, the children are referred to as a “looping segment”, i.e., a segment of your Process Timeline that iterates, or loops. A looping segment will continuously loop through the child activities until a user-defined condition, or set of conditions, is satisfied.

	Index	Activity	Type	Actions	Timeline (Days)
					1 2 3 4 5 6 7 8 9 10 11 12
	1	Iterating Parent			
	1.1	Initiator review			
	1.2	Approval1			
	1.3	Approval2			
	1.4	Approval3			
	2	User Activity			

*Note: A red callout box labeled "Loop" points to the repeating pattern of child activities (1.1-1.4) under the parent activity (1).*

Selecting the Parent Activity Type will display a Looping tab in the designer, from which the iteration conditions can be specified.

Name:

Activity | **Looping** | Start When | Completed When | Needed When | Due Date | Notifications | Advanced Options

This tab enables you to specify that the children of this parent activity should be repeated. When so configured, the child activities are referred to as a "looping segment".

The following condition will be evaluated once prior to each time the loop is repeated. The segment will stop repeating if the condition is met.

Repeat loop until condition is met.

The following conditions will be evaluated after each task within this segment is completed, and, if met, will cause the execution of the segment to be interrupted.

Jump back to the top of the loop when condition is met: [Click to create condition ...](#) Reason if users are cancelled due to jump:

Cancel the remainder of the loop when condition is met: [Click to create condition ...](#) Reason if other users are cancelled:

A parent Activity that implements iteration will also display a unique icon in the Process Timeline definition.

	2	PR Assignment Notification			
	3	Approvals			
	3.1	Initiator Resubmittal			

*Note: A red circle highlights the Looping Parent icon for activity 2.*

The Looping tab contains three methods for controlling iteration through the child activities, each of which controls some aspect of how the looping segment will work by determining the conditions under which the following actions will occur:

- Stop repeating the loop
- Jump back to the top of the loop
- Cancel the remaining activities within the segment and stop repeating the loop.

Here’s how these three methods work.

### Repeat Loop Until condition is met

Each time Process Director prepares to execute the looping segment, this condition will be evaluated to determine whether or not to do so. If the condition isn't met, the looping segment will start again. If the condition is met, however, the loop won't be repeated, and the parent Activity will be marked as complete. Any successive activities in the Process Timeline will now be eligible to run.

The remaining methods of controlling the loop rely on conditions that are evaluated each time any child Activity completes. When the relevant condition is met, execution of the loop will be interrupted.

### Jump back to the top of the loop when condition is met

When the specified condition is met, the current loop will be immediately stopped, and the loop will restart with the first child Activity.

### Cancel the remainder of the loop when condition is met

When the condition specified by the user is met, the current loop will be immediately stopped, and the parent Activity will be marked as complete. Any successive activities in the Timeline will now be eligible to run.

It is quite possible that you'll need to configure all three methods. As an example, let's consider a simple approval process for a request. Suppose that each level of approval can end in one of three results: Approve, Reject, and Terminate. In this case, we might want to configure the segment as follows:

- Stop repeating the loop when all approvers approve the request
- Exit the segment any time an approver chooses the terminate option
- Jump back to the top of the segment when any approver chooses the reject option, to give earlier reviewers a chance to make changes.

The first method for controlling the loop relies on a condition that is evaluated when your Process Timeline first runs the parent Activity, and then again each time the looping segment completes (that is, each time all of the activities in the loop have completed).

### Reason If Other Users Are Canceled/Canceled due to jump

This property gives you the ability to provide an administrative comment that will show up in the [Routing Slip](#) for the Timeline if the loop is canceled or restarted and cancels users in parallel activities.

### Advanced Options Tab

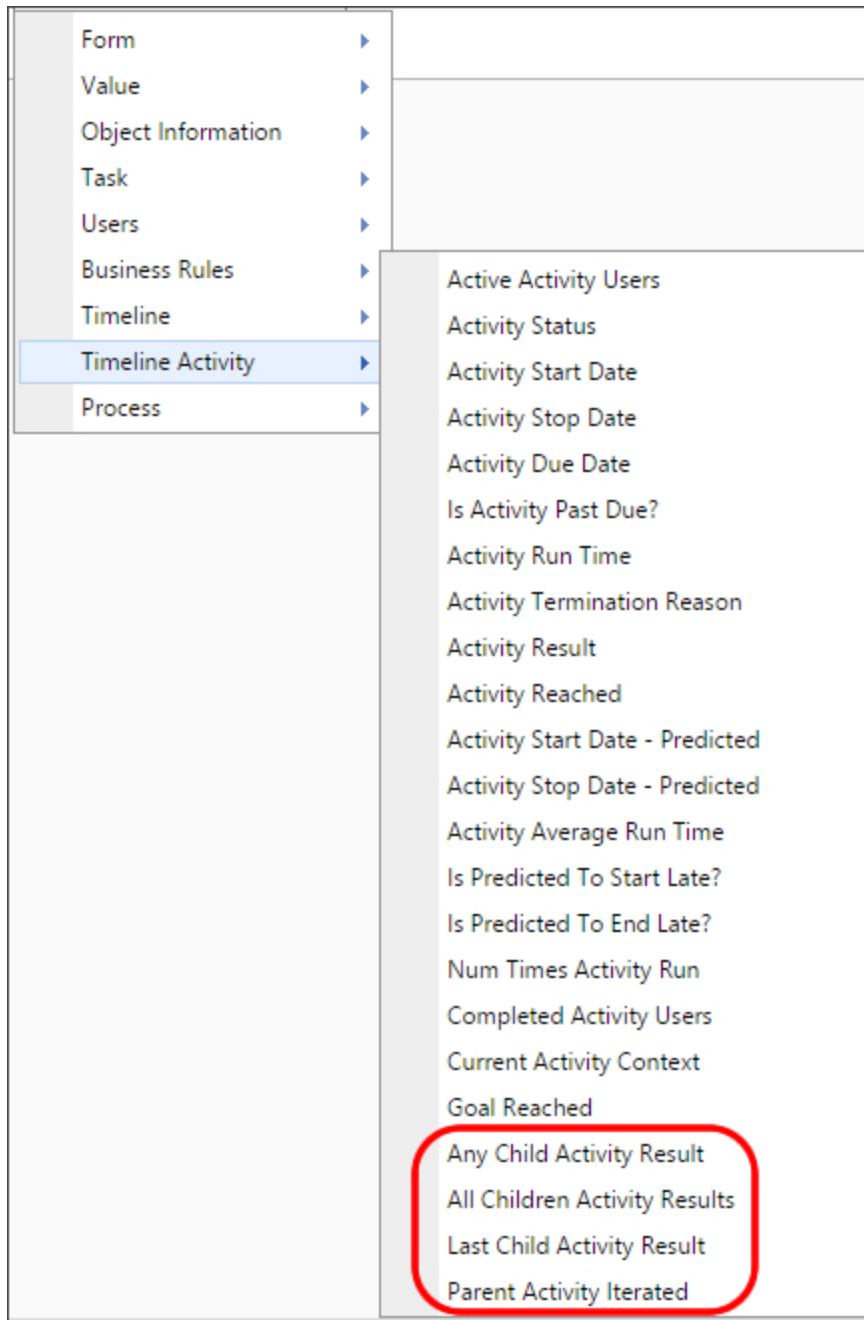
The screenshot shows a configuration window for an activity named "Activity 1". At the top, there is a "Name" field containing "Activity 1". Below this is a row of tabs: "Activity", "Looping", "Start When", "Completed When", "Needed When", "Due Date", "Notifications", and "Advanced Options". The "Advanced Options" tab is selected and active. Inside this tab, there are two checkboxes: the first is checked and labeled "Activity is required to complete parent", and the second is unchecked and labeled "Set Activity result to a list of all children or user results". At the bottom of the tab, there is a "Use Form for Activity" field with a dropdown menu and a "..." button. At the very bottom of the window, there are "Close" and "Cancel" buttons.

## Set Activity result to a list of all children or user results

Selecting this option concatenates the results of all child activities to a comma-separated list of results, which become the result for the Parent Activity.

### *Related System Variables*

The looping functionality is normally predicated on the results of the Child Activities. Specific System Variables have been implemented to make accessing the results of the Child Activities easier.



The Iteration SysVars are available from the Timeline Activity branch of the [Condition Builder's](#) System Variables menu, and consist of:

- [Any Child Activity Result](#)
- [All Children Activity Results](#)
- [Last Child Activity Result](#)
- [Parent Activity Iterated](#)
- [Parent Activity Restarted](#)

Follow the links above to see a detailed explanation of each System Variable.

### Other Activity Types

To view the documentation for other Activity Types, you can navigate to them using the Table of Contents displayed in the upper right corner of the page, or by using one of the links below.

**User:** This Activity Type assigns a task to a user or users, which must be completed to end the task.

**Notify:** This Activity Type sends email notifications to users who aren't participants in the process.

**Process:** This Activity Type invokes a different process that will run as a separate, synchronous sub-process.

**Script:** This Activity Type enables you to invoke a custom script.

**Custom Task:** This Activity Type invokes a Custom Task to run when the Activity starts.

**Form Actions:** This Activity Type enables you to manipulate the Form used for the process.

**Branch:** This Activity Type enables you to change the operation of the Process Timeline to invoke a specified Activity.

**End Process:** This Activity Type enables you to conditionally end a process.

**Wait:** This Activity Type enables you to pause a Process Timeline.

**Case:** This Activity Type enables you to manipulate the Case instance that is associated with the Process Timeline.

### Wait Activity

Wait tasks cause the Process Timeline to pause for a specified amount of time, or until a specific condition applies. In addition to the common properties tabs that appear for all Timeline activities, the configuration settings below are unique to this Activity Type.

#### Activity Tab

The screenshot shows a configuration dialog for an activity named "Activity 1". The "Name" field contains "Activity 1". Below the name are several tabs: "Activity", "Results", "Start When", "Completed When", "Needed When", "Due Date", "Notifications", and "Advanced Options". The "Activity" tab is selected. Under the "Activity" tab, the "Activity Type" is set to "Wait". There is a "Description" field. Below the description, there are "Color" (set to "Default") and "Business Days" (set to a dropdown menu). There are also fields for "Duration of activity" and "Business Days". At the bottom of the dialog are "Close" and "Cancel" buttons.

#### Duration of Activity

Enables you to specify the length of time to wait, and/or to set the calculated due date. **Any time-related Wait Activity must have a Duration set.** If no Duration is set, the Wait Activity will automatically complete, unless a condition on the **Completed When** tab is set.

## Completed When Tab

The **Completed When** tab enables you to specify condition-based criteria for determining the length of the Wait Activity. The following options are available:

Conditional Setting	Result
When conditions met	End the Activity when a specified condition set has been met.
When Sub-Processes Complete	End the Activity when all running sub-processes complete.
When Any Result Condition is Met	A wait Activity can have Results, and each Result can have specific conditions to determine which Result condition will end the task, and should be set as the result of the Activity.
Wait for event string to be posted	If this option is configured, the Activity will wait until that event is posted, usually from an external source or process.

## Due Date Tab

### Due Date is

The **Due Date Is** property enables you to set the manner in which the due date for the task will be calculated. Four options are available.

Option	Result
Calculated from the Configured Activity Start	The due date for the Activity will be calculated

Option	Result
	<p>based on when the Activity should be due, based on the configuration of the Timeline Definition's Activity durations.</p> <p><b>EXAMPLE:</b></p> <p>The Timeline definition contains two activities, each of which takes three days to complete, then this setting will calculate the due date for the first Activity as three days after the Timeline starts running, and the second Activity's due date will be calculated as six days after the Timeline starts. If the first Activity takes longer than three days to complete, the due date for the second Activity <i>won't be changed</i>. Instead, the duration of the second Activity will be shortened to meet the calculated due date on day six. If the first Activity takes less than three days, the duration of the second Activity will be lengthened to remain due on day six.</p> <p><i>Activity 2 will always be due on day six.</i></p>
No calculated due date	<p>The system won't calculate a due date for the Activity. Without a due date, escalations and notifications that are based on due date conditions won't run.</p>
Calculated from the actual Activity start	<p>The due date for the Activity will be calculated based on when the Activity actually started, rather than when it is configured to start. An Activity that is configured for three days duration will calculate the due date as three days after the Activity actually started.</p> <p><b>EXAMPLE (Continued):</b></p> <p>Irrespective of the length of the first Activity, the second Activity's duration won't be changed. The due date will be recalculated to enable the full three days of duration for the Activity.</p> <p><i>Activity 2 will always be due three days after it starts.</i></p>
Calculated from minimum of Actual or Configured Activity start	<p>The Activity's due date will be calculated by choosing the soonest date between when the Activity was configured to end, or the due date calculated from</p>


Option	Result
	<p>when the Activity began.</p> <p><b>EXAMPLE (Continued):</b></p> <p>With this setting, if the first Activity takes less than three days to complete, the second Activity will become due three days after the Activity begins. On the other hand, if the first Activity takes longer than three days, the system will shorten the Activity duration to ensure it becomes due on day six, as configured in the Timeline definition.</p> <p><i>Activity 2 will be due either three days after it begins, or on day six, whichever is soonest.</i></p>

### If time limit expires

This dropdown shows items specific to this Activity Type, to determine how the Activity will respond to the expiration of the **Duration of Activity** property that is set on the **Activity** tab.

Option	Result
Take no action	Let the Activity continue running.
Cancel This Activity	Cancel the Activity and automatically advance to the next Activity in the Process Timeline.
Cancel the entire Timeline	Cancel the Process Timeline.
Wait until at least Due Date	Keep the Activity in a wait status until at least the specified due date. This must be set when using the Wait Activity to pause for a specified period of time, as determined by the <b>Duration of Activity</b> you set in the <b>Activity</b> tab.

It is possible to set both a time- and condition-based Wait-Activity. The settings you choose on this tab will, however, change the way the Activity responds. For example, if you set both a completion condition as well as a **Duration**, then set the **If Time Limit Expires** property to "Wait until at least Due Date", then the Activity will NOT end until the **Duration** expires, even if the condition evaluates as true.

 Setting a **Duration** AND setting the **If time limit expires** property to "Wait until at least Due Date", will ALWAYS keep the Timeline Activity in a wait status until the **Duration** time is met.

### Other Activity Types

To view the documentation for other Activity Types, you can navigate to them using the Table of Contents displayed in the upper right corner of the page, or by using one of the links below.

**User:** This Activity Type assigns a task to a user or users, which must be completed to end the task.

**Notify:** This Activity Type sends email notifications to users who aren't participants in the process.

**Process:** This Activity Type invokes a different process that will run as a separate, synchronous sub-process.

**Script:** This Activity Type enables you to invoke a custom script.

**Custom Task:** This Activity Type invokes a Custom Task to run when the Activity starts.

**Form Actions:** This Activity Type enables you to manipulate the Form used for the process.

**Branch:** This Activity Type enables you to change the operation of the Process Timeline to invoke a specified Activity.

**Parent:** This Activity Type serves as a container for other activities and to create a looping segment in a Process Timeline.

**End Process:** This Activity Type enables you to conditionally end a process.

**Case:** This Activity Type enables you to manipulate the Case instance that is associated with the Process Timeline.

## Managing Process Timeline Instances

When a Process Timeline process is started, a Process Timeline instance is created. This instance is what is routed to the participants of the Process Timeline. The Process Timeline instance contains the **Routing Slip**, the Process Timeline objects (documents, Forms, etc.), optional references, and administrative controls. The Process Timeline instance can be viewed by authorized users for running and completed Process Timelines.

### Routing Slip #

Authorized users have access to the **Routing Slip** in the Process Timeline instance by clicking on the **Timeline Status** Tab. This tab displays a **Routing Slip** that shows the progress of the route. It displays all users that are participating in this Process Timeline and how far it has progressed in the routing process, including all iterations of an Activity.

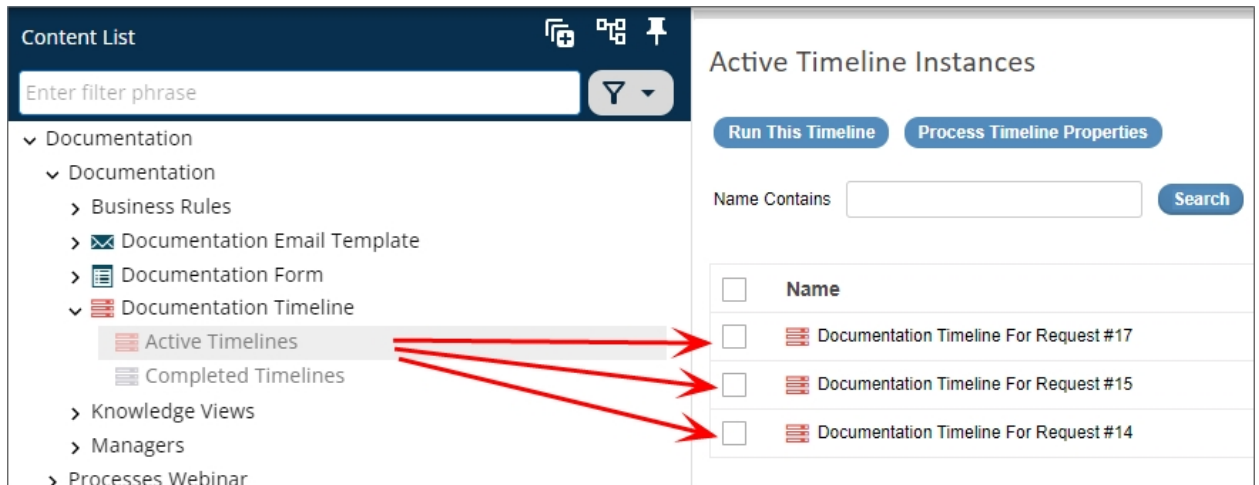
Routing Slip						
Participants	Signature	Completed	Status	Result	Comments	
<b>Initiator</b>						
Ron Harris	<i>Ron Harris</i>	7/28/2015	Completed			
<b>Confirmation</b> 7/28/2015 12:26 PM						
Ron Harris	<i>Ron Harris</i>	7/28/2015	Completed	✔ Complete Application		
<b>Administrative Review</b> 7/28/2015 12:28 PM						
Diana Stuart	<i>Diana Stuart</i>	7/28/2015	Completed	✔ Review Complete		
<b>Confirmation Notification</b> 7/28/2015 12:28 PM						
Ron Harris		7/28/2015	Notified			
<b>Admissions Review</b> 7/28/2015 12:31 PM						
Barb Stanley	<i>Barb Stanley</i>	7/28/2015	Completed	✔ Accept Application		



The active Activity in a running process will be highlighted in Process Director v4.53 and higher.

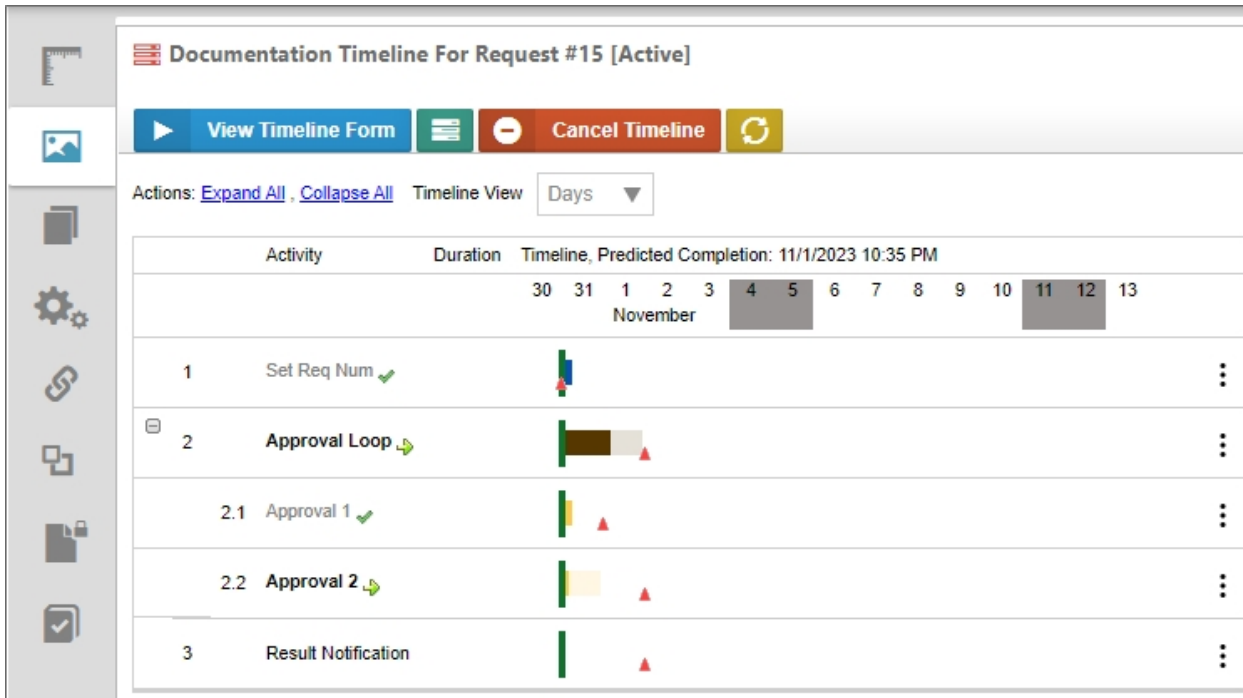
## Active/Completed Timelines #

When you open a Timeline definition, the folder list on the left side of the [Content List](#) screen will automatically expand the Timeline treeview to display the [Active Timelines](#) and [Completed Timelines](#) branches. Clicking on one of the branches will show all instances of the running or completed Timelines, respectively, on the right side of the [Content List](#). Clicking on any running or completed Timeline instance will open the instance in the [Timeline Status](#) view.



## Administration/Timeline

Only authorized users will see this button displayed, which opens the Timeline administration view. The user must have Modify permission for the running Process Timeline or have Modify Children permission for the Process Timeline Definition. The Administration button in the Process Timeline instance allows users to modify the running Process Timeline. From this screen, authorized users can change the running Process Timeline by adding users to an Activity, removing them from an Activity, or jumping to a new Activity.



The Process Timeline administration page displays information about the state of the running Process Timeline instance. It can tell you when tasks are expected to complete, and indicates if that means they'll be late. It can compare the configured run time of an Activity to its actual and predicted run times. To view analyses of average Process Timeline performance, go to the Analyze Timeline page on the Process Timeline definition.

The triangles represent due dates. They can be dragged along the Process Timeline to change the configured due date for a single Process Timeline instance.

The green bar indicates the current date and time, juxtaposed onto the configured dates and times for the Process Timeline.

Green arrows will appear to the left of activities that are currently running.

A checkbox next to an Activity name indicates that Activity has been completed.

1	Travel Approvals ✓
---	--------------------

If a Process Timeline is in an error state, Process Director will display the Activity in which the error occurred highlighted in red. Rolling your mouse over the Activity will also display an error message.

**Documentation Timeline For Request #15 [Active]**

View Timeline Form | Cancel Timeline

One or more Activities are in an error state

Actions: Expand All | Collapsed All | Timeline View | Days

Activity	Duration	Timeline, Predicted Completion: 10/31/2023 3:59 PM
1 Set Req Num ✓		
2.1 Approval 1 ✓		
2.2 Approval 2 ✓		
3 Result Notification		

**Approval 2 [User]**

This activity is in an error state

Message: Putting activity into an error state because too many results met condition and no default setting.

Activity Status: Active | Result: | Configured Start: 10/31/2023 10:35 PM

Activity Started: 10/30/2023 10:35 PM | Predicted Completion: 10/31/2023 3:59 PM | Activity Due: 11/1/2023 10:35 PM

Participants	Signature	Completed	Status	Result	Comments
Approval 2	10/30/2023 10:35 PM				
Dale Franks		-	Active		

## Administering Process Timeline Activities

You can access the properties for a Timeline Activity displayed in the **Administration/Timeline** tab by right-clicking on the Activity, and selecting **Properties** from the pop-up menu.

2.1 Approval 1 →

2.2 Approval 2

Properties  
Cancel this Activity  
Restart this Activity

In the **Timeline Instance Activity Properties** tab, you can access the full range of Activity and user properties by checking the box next to the Activity's user name in the Participants column of the Activity's **Routing Slip**.

Timeline Instance Name: PR/PO Process (Online) (Purchase Request (Online))

### TIMELINE INSTANCE ACTIVITY PROPERTIES

**Timeline**  
Timeline Status: Active | Timeline Started: 9/20/2019 | Timeline Ended: -

**Initial Approval**  
Activity Status: Active | Termination Reason: Not Set | Result: Activity Due: 9/24/2019

**Routing Slip Options**

- Show Assigned On
- Show Signatures
- Most Recent Instances Only
- Show Cancelled Users
- Show Completed Users
- Show Reassigned Users
- Show Pending Users
- Show Running Users
- Show Timed Out Users
- Show Predicted Path

Actions: [Cancel Activity](#), [Restart Activity](#), [Cancel Timeline](#), [Add User\(s\)](#), [Add Me To This Task](#)

Search for users:

Administrative Comments:  Role Name:

<input type="checkbox"/>	Participants	Signature	Completed
▼	Initial Approval	9/20/2019 10:40 AM	
<input checked="" type="checkbox"/>	Diana Stuart		-

With the user check box checked, you now have access to the full range of Timeline and User Actions available for the Activity, displayed in a list of links marked "Actions:". The full list of actions are described in the table below:

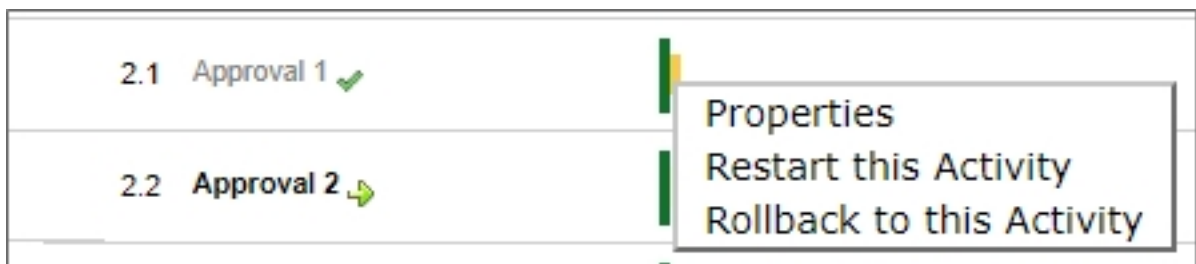
OPTION	RESULT
Cancel Activity	<p>Cancels this Activity in the Process Timeline.</p> <div style="border: 1px solid #0056b3; border-radius: 10px; padding: 10px; background-color: #e6f2ff;"> <p><b>i</b> When a Parent Activity is administratively canceled, the system will ripple the user and cancellation comments to all the child activities that are being canceled because the parent is canceled.</p> </div>
Restart Activity	Restarts this Activity in the Timeline. User, Wait, Custom Task, or Script Activities may all be restarted.
Cancel Timeline	Cancels the entire running Timeline instance.
Add User(s)	Adds additional users to the Timeline Activity.

OPTION	RESULT
Add Me to This Task	Adds you to the Timeline Activity as a participant.
Cancel User	Cancels the user's participation in the task.
Complete User Task	Sets the user's task as complete.
Remove User	Remove the user from the task.
Restart User	Restart this user task.
Resend Email	Resend the task notification email to the user.
Reassign User	Assign a different user to the task.
Open My Task	If you are the user to whom the task is assigned, this link will open the Form that is associated with the Timeline. The Form will open in the context of the user task, enabling you to complete the task.

When canceling a Parent Activity in a Process Timeline, all child activities will also be canceled.

### ***Restart and Rollback***

In addition to viewing an Activity's properties, you also have the option to restart a task, or rollback to a task.

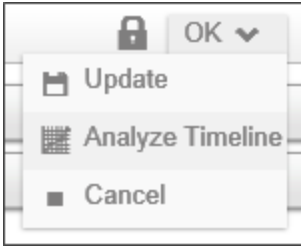


**Restart This Activity:** You can choose an Activity that has already completed, and restart it. This will only restart that Activity, and won't restart any subsequent activities in the Process Timeline. Once the restarted Activity completes, no further action is taken. If you are in a later Activity and you restart, the later Activity will still run normally.

**Rollback to this Activity:** This option only appears when you select an Activity that is on a critical path. If you rollback to an earlier Activity in the same critical path, it will cancel the current Activity, restart the selected Activity, and place all subsequent activities in the critical path into a reset state, so that they can run again. If you rollback to an Activity in a different critical path, it will **not** cancel the current Activity, it will only restart the critical path for the Activity you selected. In that case, both the rollback Activity and the current Activity will run simultaneously. Only if a previous Activity is in the same critical path as the current Activity will the current Activity be canceled upon rollback.

### ***Analyze Timeline***

When viewing a Process Timeline definition, there will be an option to analyze the Process Timeline in the object menu at the upper right portion of the Timeline Definition screen. This analysis view will display data about average predicted run times and start and end dates compared to the configured run times and start and end dates.



A Timeline Analysis page will display the tasks in the Process Timeline definition, and will overlay average times and dates on the times and dates configured in the Process Timeline definition.

<input type="checkbox"/>	Activity	Type	Duration	Actions	Timeline (Days)
<input type="checkbox"/>					1 2 3 4 5 6 7 8 9 10 11 12
<input type="checkbox"/>	1 Set Req Num		0 Day(s)		-83%
<input type="checkbox"/>	2 Approval Loop		2 Day(s)		-67%
<input type="checkbox"/>	2.1 Approval 1		1 Day(s)		-87%
<input type="checkbox"/>	2.2 Approval 2		1 Day(s)		-87%
<input type="checkbox"/>	3 Result Notification		0 Day(s)		-57%

Triangles represent start and end dates for each process. If a triangle is red, it means that the start or end date for an Activity is happening after it was configured to. It doesn't mean that the Activity is taking longer than it was configured to. If a triangle is blue, it means that the start or end date is happening approximately when it was configured to happen, and if it is green, it means that it is happening before it was configured to happen.

A line between two triangles indicates the expected (weighted average historical) duration of the Activity, while its color denotes whether it is generally finishing late, early, or as planned. A red line means that the Activity is finishing later than planned; a blue line means that it is finishing as planned, and a green line means that it is finishing earlier than planned. A percentage number on an Activity bar indicates the percentage of time longer that the Activity takes to run than was planned. For example, a number of 80% indicates that the Activity takes 80% longer to run than planned, and a number of -24% indicates that the Activity completes in a time 24% shorter than planned.

Additional options are available by right-clicking the individual bars on the Timeline to bring up a pop-up menu.



### Properties

Selecting the Properties menu item will display the normal Properties screen for this Timeline Activity.

### Show Dependencies

Selecting the Show Dependencies menu item will display the color-coded view of the Timeline Activity's predecessors and successors.

<input type="checkbox"/>	Activity	Type	Duration	Actions	Timeline (Days)	
<input type="checkbox"/>					1 2 3 4 5 6 7 8 9 10 11 12	
<input type="checkbox"/>	1 Set Req Num		0 Day(s)			Predecessor Activity
<input type="checkbox"/>	2 Approval Loop		2 Day(s)			Selected Activity
<input type="checkbox"/>	2.1 Approval 1		1 Day(s)			
<input type="checkbox"/>	2.2 Approval 2		1 Day(s)			Successor Activity
<input type="checkbox"/>	3 Result Notification		0 Day(s)			

### Analyze Tasks/Users in this Activity

The menu item will change slightly to analyze either tasks or users depending upon the Activity Type. For example, a user Activity will display a user analysis option, while a parent Activity will display a task analysis option. In either case, selecting the menu item will display an analysis screen that provides a snapshot of how the selected Activity is performing.

**Initial Approval: User Analysis**

# Unique Entries 4

Longest Completions Eric Wintemute [ 24.9 Business Days ] Fastest Completions Rick Rob [ 0 Business Days ]

Most Overdue Completions Eric Wintemute [ 1 ] Least Overdue Completions Dale Franks [ 2 ]

All times shown in Business Days

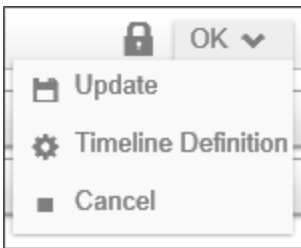
Name	Count	Average Time	Max Time	Min Time	Times Overdue	Times Early
Eric Wintemute	1	24.9	24.9	24.9	1	0
Diana Stuart	1	0	0	0	0	1
Rick Rob	1	0	0	0	0	1

Close

## Process Timeline Items

The Process Timeline Items page is what is presented to users that are participating in a Process Timeline Activity. This page displays all of the objects being routed in this Process Timeline (e.g. documents, Forms, etc.). A Process Timeline instance can contain multiple objects of any type. If the Form has an attach button, users in that Activity will be able to add objects to the Process Timeline instance. Each new object added is added to the Process Timeline Items page. If a user has Delete permission to the running Process Timeline they'll be able to remove objects from the Process Timeline instance.

To return from the Analysis view to the Process Timeline Definition view, click the **Process Timeline Definition** menu option.



## Processes in Error #

It's not always easy for an administrator to determine if any running processes are in an error state. You can, however, easily create a Knowledge View to return only processes that are in an error state. In the example below, you can access a software simulation that walks through the process of creating a Processes In Error Knowledge View. You can run this Knowledge View at any time to see all of the errant processes on your system.

### Use the Simulator

[Build a Processes in Error Knowledge View](#)

## Process Timelines in the Content List #

Process Timeline Definitions are stored in the [Content List](#) just like any other object in the database. When a Process Timeline is started, an entry will be created under the Process Timeline Definition in the [Content List](#) as a child object. It will remain there even after the Process Timeline has completed. To access the child instances of a Process Timeline definition, Users can select the View Children menu

item from the hamburger menu (☰).

<input type="checkbox"/>	 Documentation Form	Dale Franks	10/30/2023	10/30/2023	0 fields	⋮
<input type="checkbox"/>	 Documentation Timeline	Dale Franks	10/30/2023	10/30/2023	5 activities	⋮

Properties

References

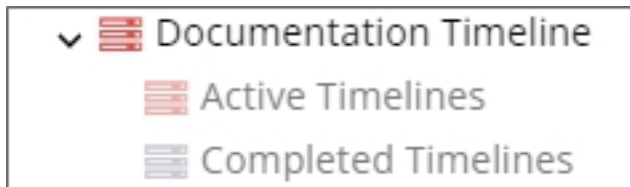
View Children



For versions below Process Director v6.0.300, users can click on the view children (📁) icon next to the Process Timeline Definition name in the Content List. Users can also view the Process Timeline instance for any of the running or completed Process Timelines by clicking on the name or the 📄 icon in the [Content List](#).

### Viewing Running and Completed Process Timelines

If users have View Children permission to the Process Timeline Definition, they'll automatically be given View permission to any running and completed Process Timelines for that definition. The running and completed Process Timelines are displayed in the [Content List](#) showing status, what Activity is running and the user that initiated the Process Timeline.



Running and completed Process Timelines can also be made available to users through a Knowledge View. For example, if users have a need to see only certain types of running Process Timelines, a Knowledge View filter can show a list of the Process Timelines making the actual location of them in the [Content List](#) transparent.

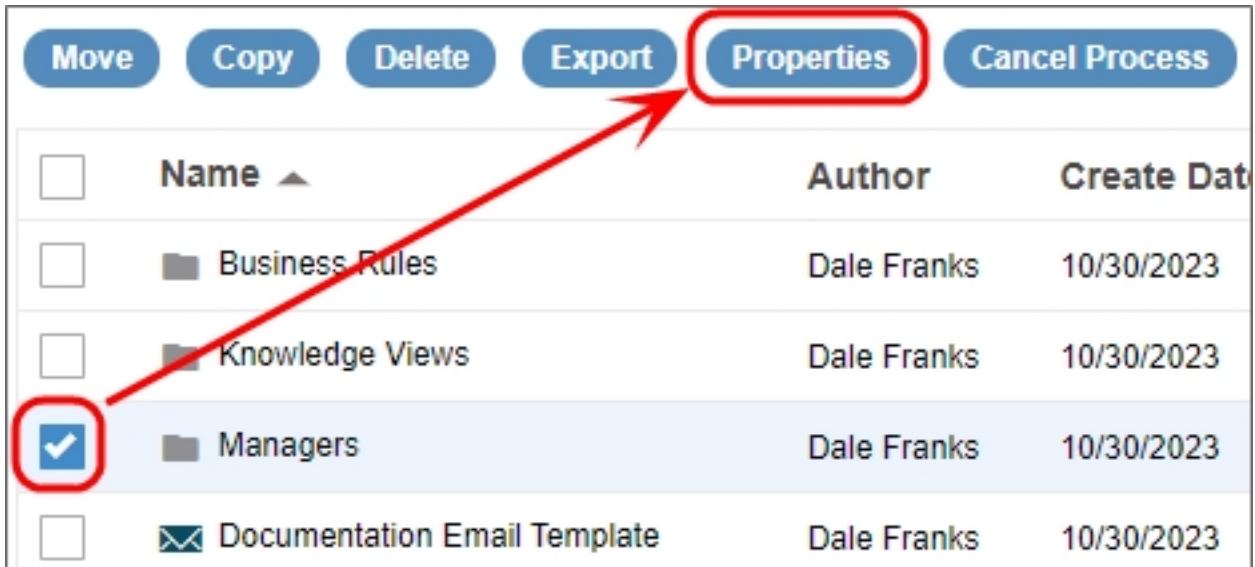
### Attaching New Process Timeline Objects

A running Process Timeline can allow users to add new objects (e.g. documents, Forms). If a user has an attach button on a Form, they'll be able to add new objects to the running Process Timeline. When a new document is added under the Process Timeline instance it will only be contained under the Process Timeline. The document won't exist anywhere else in the [Content List](#).

When a reference (e.g. shortcut) to an existing object in the database is added, only a reference to the object will be created in the Process Timeline. A reference, or shortcut, is a symbolic link to the original object. Any changes made to the reference will also update the object it is pointing to, except in the case of a Form. If a reference to a Form is added to the running Process Timeline, an instance of the Form will be created and placed under the Process Timeline Items button.

### Automatic Process Timelines for Folders

Process Director Process Timeline engine supports the ability to automatically start Process Timelines for certain document events within a folder. Automatic Process Timeline Definitions can be specified for any folder, on the folder's properties page. To see the folder properties, select the check box for the folder, then click the [Properties](#) action button.

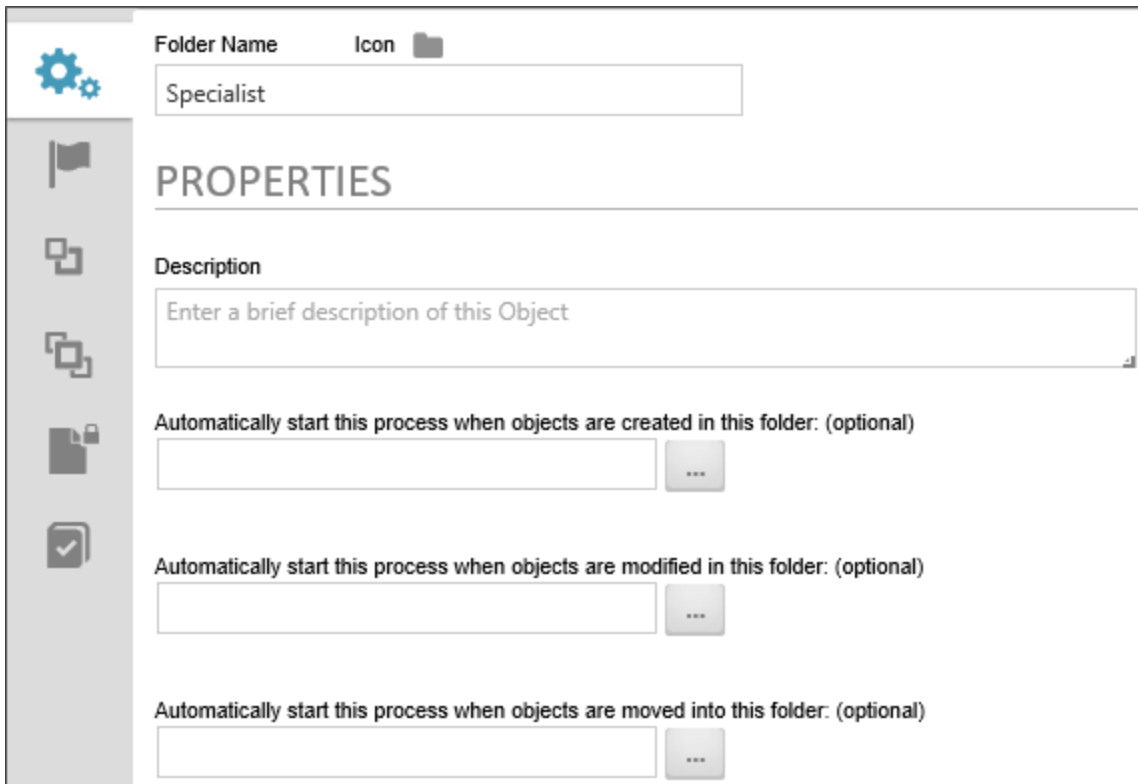


<input type="checkbox"/>	Name ▲	Author	Create Date
<input type="checkbox"/>	Business Rules	Dale Franks	10/30/2023
<input type="checkbox"/>	Knowledge Views	Dale Franks	10/30/2023
<input checked="" type="checkbox"/>	Managers	Dale Franks	10/30/2023
<input type="checkbox"/>	Documentation Email Template	Dale Franks	10/30/2023

A Process Timeline Definition can be associated with one of the following document actions:

- When a new document is added to a folder;
- When a document in a folder is updated and checked in;
- When a document is moved into a folder.

If one of these conditions is met, the appropriate Process Timeline will automatically be started against that document.



Folder Name: Specialist

### PROPERTIES

Description: Enter a brief description of this Object

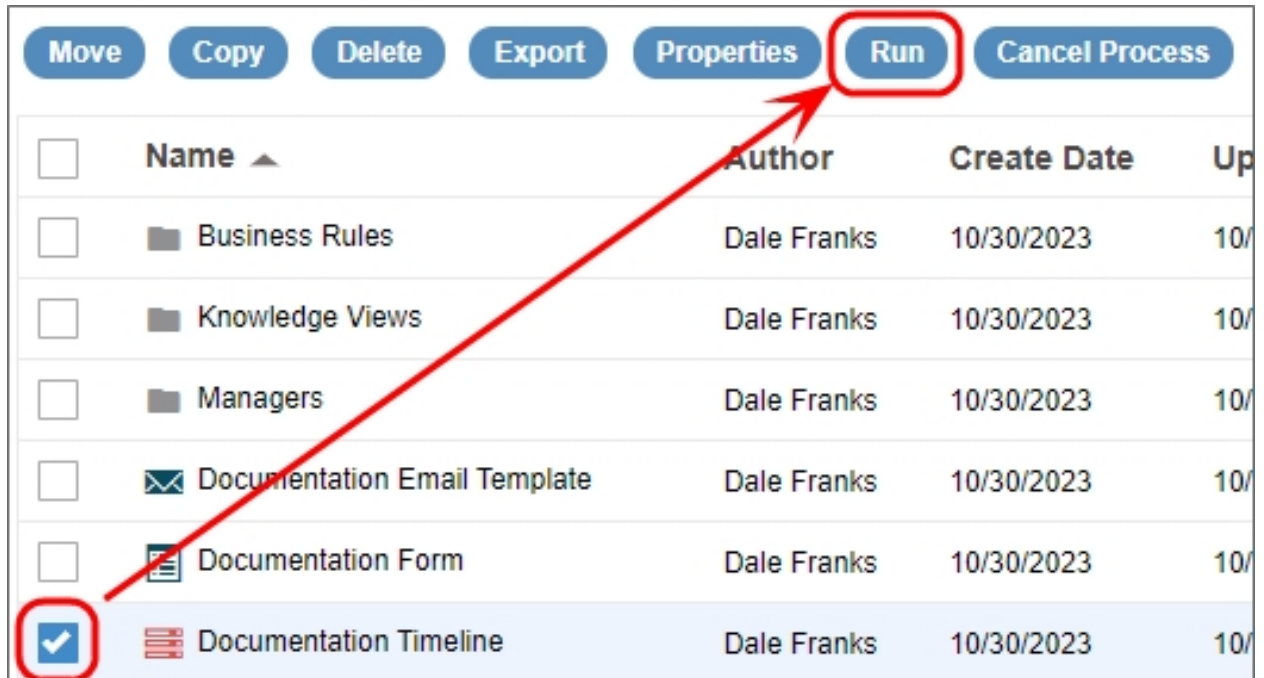
Automatically start this process when objects are created in this folder: (optional)

Automatically start this process when objects are modified in this folder: (optional)

Automatically start this process when objects are moved into this folder: (optional)

## Running Process Timelines

A Process Timeline can be started by selecting the check box next to a Process Timeline definition in the [Content List](#) and choosing the **Run** Action Button. This will start a new instance of the Process Timeline immediately.



Changes to the Process Timeline Definition will affect all running Process Timelines. Please see the section on [Changing Existing Processes](#) for more detailed information.

### Task Lists (To Do Lists)

[Task Lists](#) are fundamental to Process Timeline management systems. The Process Director Process Timeline engine supports an integrated Task List that provides each user with a list of their assigned tasks, according to priority and due date. As a user completes an assigned task, the item is automatically removed from their [Task List](#).

### Permissions and Running Process Timelines

A running Process Timeline will temporarily override the permissions for any object (document, Form, etc.) contained within it, giving the user the necessary access to perform the requested task. When users are assigned to a Process Timeline task, they'll automatically be given Modify permission to the objects until their task is completed. This prevents the object permissions from effecting running Process Timelines.

### Loop Detection

When Process Director detects a possible loop in a running process instance, it delays the process until the next time that the system's timers run. The system will display a message on the [Timeline Status](#) tab for the instance, indicating that the looping condition has been detected.

## Running a Timeline from a URL #

Timelines can be run manually, or scheduled to perform an automatic actions. To schedule these Timelines, a web page (a Web Service web page) is executed to run a specific Timeline definition.

Ensure you have enabled Web Services on the Installation Settings page. You can optionally enable security restrictions and authentication. If you pass credentials on the URL, you'll need to set [fWebServiceAllowCredentialsURL](#) to true in the custom vars file.

### Manual Timeline

Execute the following URL from a browser on the server that has Process Director:

```
https://ServerName.com/services/wsTimeline.aspx/Run?tlid=TLID&bpuserid=USERID&bpPassword=PASSWORD
```



Be advised that using the `bpUSERID` or `bpPassword` parameter requires sending the User ID and Password in clear text, so be mindful of the security implications of transmitting these values.

The `wsTimeline.aspx` command can take the following parameters on the QueryString URL:

- **tlid:** The Timeline ID of the Timeline Definition to run;
- **bpuserid:** A Process Director user ID with permission to run the Process Definition;
- **bpPassword:** A Process Director password with permission to run the Process Definition;

### Scheduled Timeline

You may want to automatically schedule the Timeline to run at regular intervals (for example, every night at midnight). To do this, use the `bputil.exe` utility. This utility enables you to schedule and test commands executed on a regular basis.

Do not schedule `IEXPLORE.EXE` because the web browser will never close. Rather, use the `bputil.exe` command to run the web page. For example, enter this command in the “Run” dialog box to schedule the synchronization:

```
"PATH\bputil.exe" SU  
https://ServerName.com/services/wsTimeline.aspx/Run?tlid=TLID&bpuserid=USERID&bpPassword=PASSWORD
```

(Where `PATH` is the installation directory for Process Director, e.g. `c:\Program Files\BP Logix\Process Director\`). Enter the appropriate credentials in the Windows Scheduler when prompted. Use the “Schedule” tab to set the times to run the command. Consult the Microsoft help for more information on this utility.



Be advised that using the `bpUserID` or `bpPassword` parameter requires sending the User ID and Password in clear text, so be mindful of the security implications of transmitting these values.

## Reports

Users of Cloud Installations, or On-Premise Installations with the Advanced Reporting option, have access to Process Director's Reports component, a sophisticated report generation utility. The report generator itself is fully documented in the [Reports Reference Guide](#).

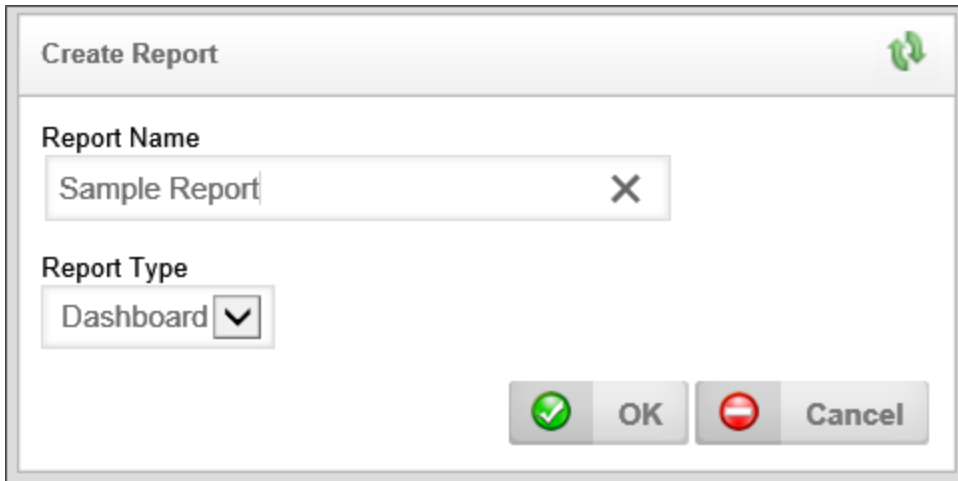
The purpose of this section of the Implementer's Guide is to cover only how the report generator integrates into Process Director. Please refer to the Reports Reference Guide for detailed instructions on how to use the report generation utility to design and create reports.

### Creating a report #



Creating a report in Process Director is as easy as creating any other object. From the [Content List](#), navigate to the folder where you'd like to create the report. Once you have done so, select Report from the Create New... dropdown located in the upper right portion of the screen.

The Create Report screen will appear. In this screen, enter the name of the new report, then select the report type you wish to create.



You have three choices of report type: Dashboard, Multi-page, and Portlet.

### Dashboard

A layout with multiple “panels” in the report designer to assist you with spacing multiple charts on one page. It is approximately 10.75” x 5.5”. This option is best for reports that display infographics.

### Multi-Page

A report laid out on 8.5” x 11” letter sized paper designed for displaying pages of results and printing. This option is best for reports that present lengthy tabular or logical data.

### Portlet

A 5.7” x 2.5” layout designed for a quarter panel of a Process Director workspace.


Keep in mind that you can create custom sizes, but you'd need to make sure that they show up correctly relative to your user's aspect ratios and monitor resolutions.

Once you have selected the desired [Report Type](#), click the **OK** button to create the report, and display the report properties.

## Report Configuration <#>

There are three tabs for configuring a report, the [Properties](#), [Data Sources](#), and [Variables](#) tabs.

## Properties Tab

Report Name    Icon 

Sample Report

---

### PROPERTIES

Description

*Enter a brief description of this Object*

**Design Report**

**View Report**

---

**Options**

Report Type **Multi-Page**    Zoom Mode **Page Width**

Prompt for variables before running report     Do not show this object in 'Items I Can Run' Knowledge Views

Use Flash Viewer

Show ToolBar     Show Export     Show Navigation Buttons     Show Print     Show View Mode     Show Zoom

Report URL  
 :  
 http://dalespc/report.aspx?ID=ee19dd5b-293b-482c-9a10-f6dfc36a6b1f

### Description

The report's description, which will appear in the second line of the [Content List](#) for the Report item.

### Design/View Report

The **Design Report** and **View Report** buttons allow you to create and view your report. See the [Advanced Reporting](#) documentation for details on designing a report.

### Options section

The **Options** section contains several configurable properties, which are listed below

**Report Type:** You have three choices of report type: Dashboard, Multi-page, and Portlet.


**Zoom Mode:** Zoom Mode determines the zoom level at which the report will display.

**Prompt for variables before running report:** The Variables that are sent to the reporting engine via Process Director (covered in section 3 below) may be changed by the end user when running the report, before the results are displayed.

**Do not show this object in 'Items I Can Run' Knowledge Views:** Checking this option will prevent the Report from displaying in the Knowledge Views.

**Use Flash Viewer:** Reports can be created or viewed using HTML or Flash. Using Flash requires the Flash plug-in, available from Adobe. The two types of reports look very similar, but the Flash version is better

suited to displaying Portlet and Dashboard reports, as the Flash viewer automatically resizes the report to the available space.

 This setting remains as a legacy setting for backwards compatibility; however, the use of Flash has been deprecated or eliminated in all modern browsers.

**Show Toolbar:** Make the Bookmarks, Thumbnails, Find, and Full-Screen options available on the report.

**Show Export:** Display an option on the report enabling the user to save it in a wide variety of file types.

**Show Navigation Buttons:** Display page navigation buttons for multi-page reports, allowing the user to jump to a given page location within the report, rather than just viewing pages in succession.

**Show Print:** This will display the print button, allowing the user to print the report.

**Show View Mode:** Display options in the lower right hand corner of the report allowing the user to resize the report on the screen and show multiple pages at once.

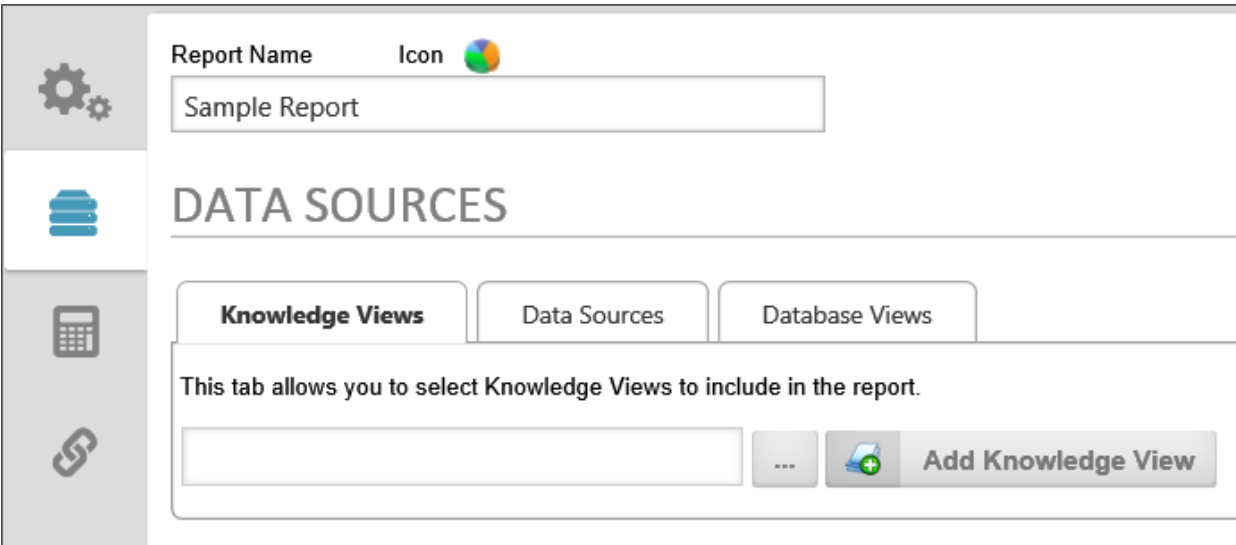
**Show Zoom:** The zoom controls enable the user to automatically size the report to fit his or her display.

## Report URL

The Report URL will allow you to copy and paste a link directly to this report.

## Data Sources Tab

The **Data Sources** Tab allows you to specify how and what information you send to the Report Designer.



The screenshot shows the 'Data Sources' tab in the Report Designer. At the top, there is a 'Report Name' field containing 'Sample Report' and an 'Icon' field with a colorful globe icon. Below this, the 'DATA SOURCES' title is displayed. There are three tabs: 'Knowledge Views', 'Data Sources', and 'Database Views'. The 'Knowledge Views' tab is selected. Below the tabs, a message states: 'This tab allows you to select Knowledge Views to include in the report.' At the bottom, there is an empty input field, a three-dot menu icon, and an 'Add Knowledge View' button with a plus icon.

**Knowledge Views:** You can import data from a previously constructed Knowledge View in Process Director. It makes visually displaying your already constructed reportable information extremely easy.

**Data Sources:** This is the preferred way to retrieve data for reporting purposes. Using a Data Source that has specific tables and views selected and then editing the SQL SELECT statements would help produce the fastest possible reports.



**Database Views:** This Datasource will display not only the default views included in Process Director, but also the views that you can create in Process Director, based on forms.

## Variables Tab #

Process Director sends variables to the reporting engine automatically, including current user information, the name and description of the report, etc. You may also create custom variables and send them to the reporting engine.

Please see the [Report Variables topic](#) for more information about using Variables for the Advanced Reporting Component.

To add a variable to the report, click the **Add Variable** button.

Variables have the following configuration options:

### Name

The name of the variable.

### Variable Type

The data type of the variable. The following choices are available:

- String
- Number
- Yes/No
- Date

- User
- Group
- Dropdown
- Form
- Process
- Content Item

### Default Value

The initial value for the variable, or the value if no other value is available.

### Prompt?

If checked, the user will be prompted for a value.

### Prompt Text

The text that will be used to prompt the user.

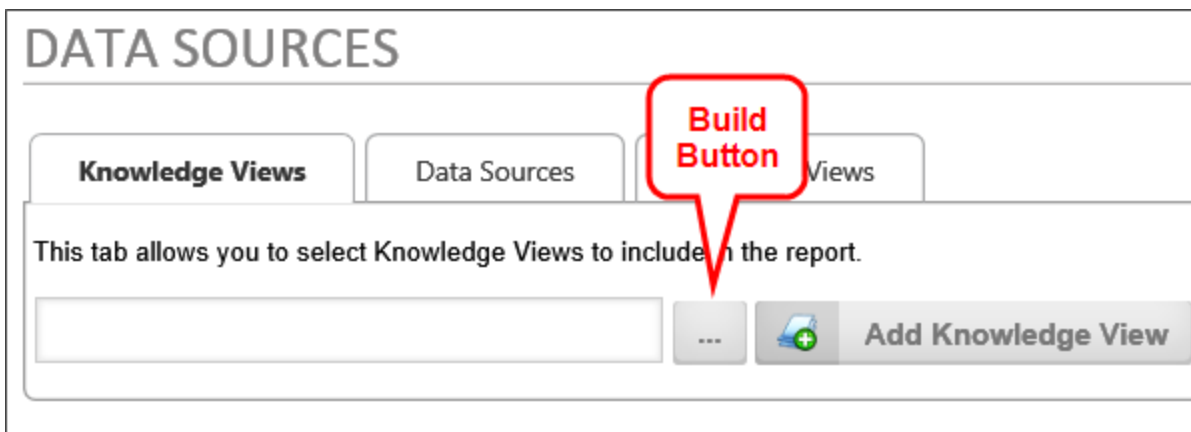
## Selecting Report Data #

The reporting component can use three different types of data to compose reports: Knowledge Views, Data Sources, and Database Views. The fields contained in all of the report data sources you choose from the options screen will be available inside the report component when you design the report.

Once you have chosen the data you want to incorporate into your report, click the Update button to save your selections.

### Knowledge Views

Reports can be constructed using any Knowledge View, and multiple Knowledge Views can be included in a report. To add a Knowledge View to a report, click the Knowledge View tab of the report's options screen. Next, click the picker control's Build button, to open the [Content List](#).



From the [Content List](#), select a Knowledge View to use for the report's data, then click the **OK** button to add the Knowledge View to the picker. The name of the Knowledge View will appear in the picker control's text box.

Now, click the **Add Knowledge View** button to add the Knowledge View to the report. The Knowledge View will appear in the list of Knowledge Views in the report.

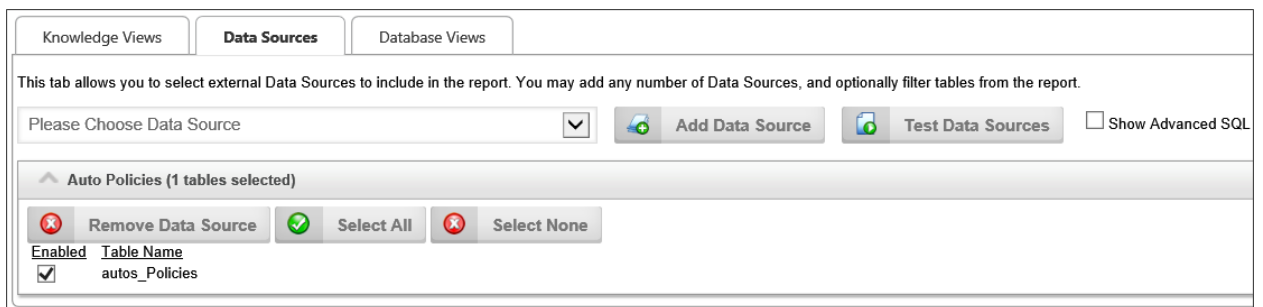


You can view the Knowledge View by clicking the button with the Knowledge View icon, or you can remove the Knowledge View by clicking the **Remove Knowledge View** button.

## Data Sources

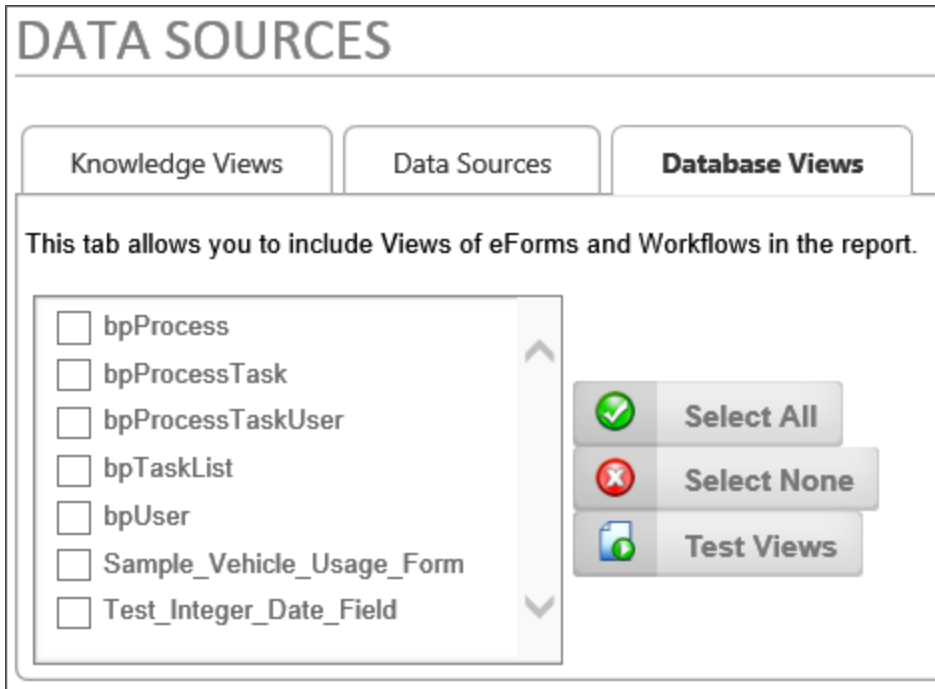
Just like Knowledge Views a report can use multiple data sources, and you can choose any Datasource object in the [Content List](#). This gives you access to data from SQL Server, social media, Microsoft Dynamics, or any other Datasource.

To add a Datasource, select the desired Datasource from the **Please Choose Data Source** dropdown. You can add the Datasource by clicking the **Add Data Source** Button. You can also test the Datasource to ensure the Datasource returns data by clicking the **Test Data Sources** button, which will cause a message to display, which shows the number of rows returned from the Datasource. You can also show the Advanced SQL syntax used to extract the data by clicking the **Show Advanced SQL** checkbox.



## Database Views

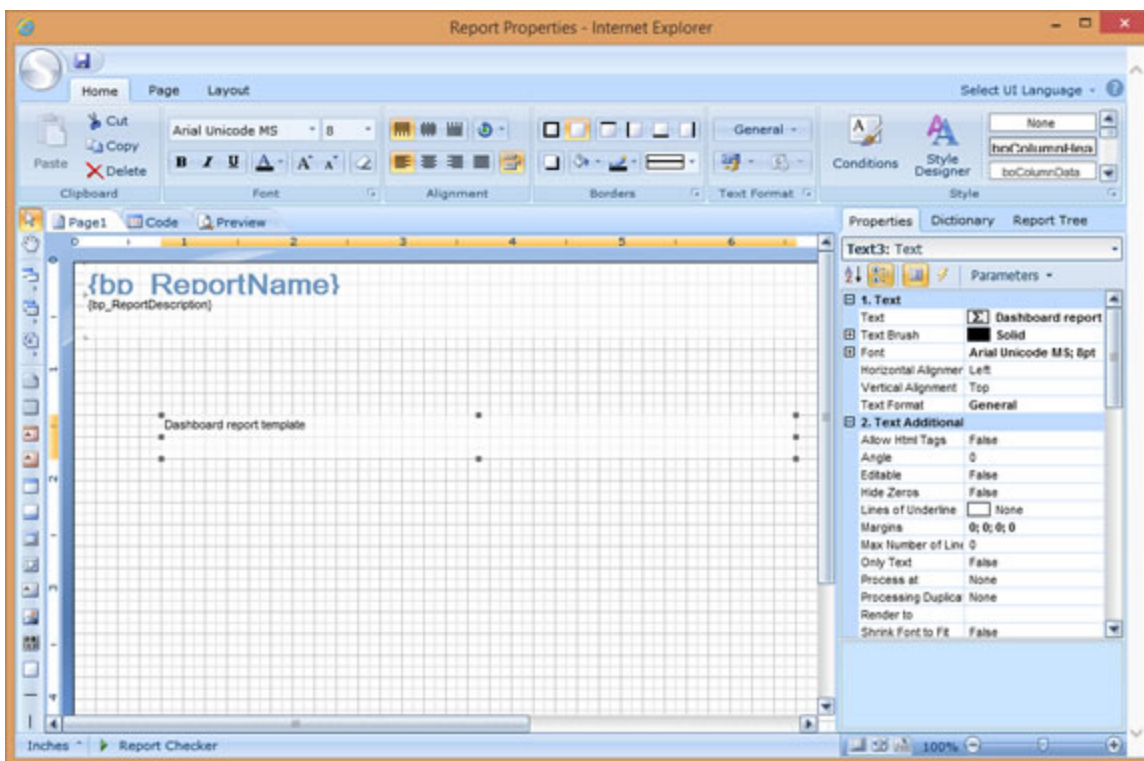
Process Director's internal database views of Forms and Processes may also be used as report data sources.



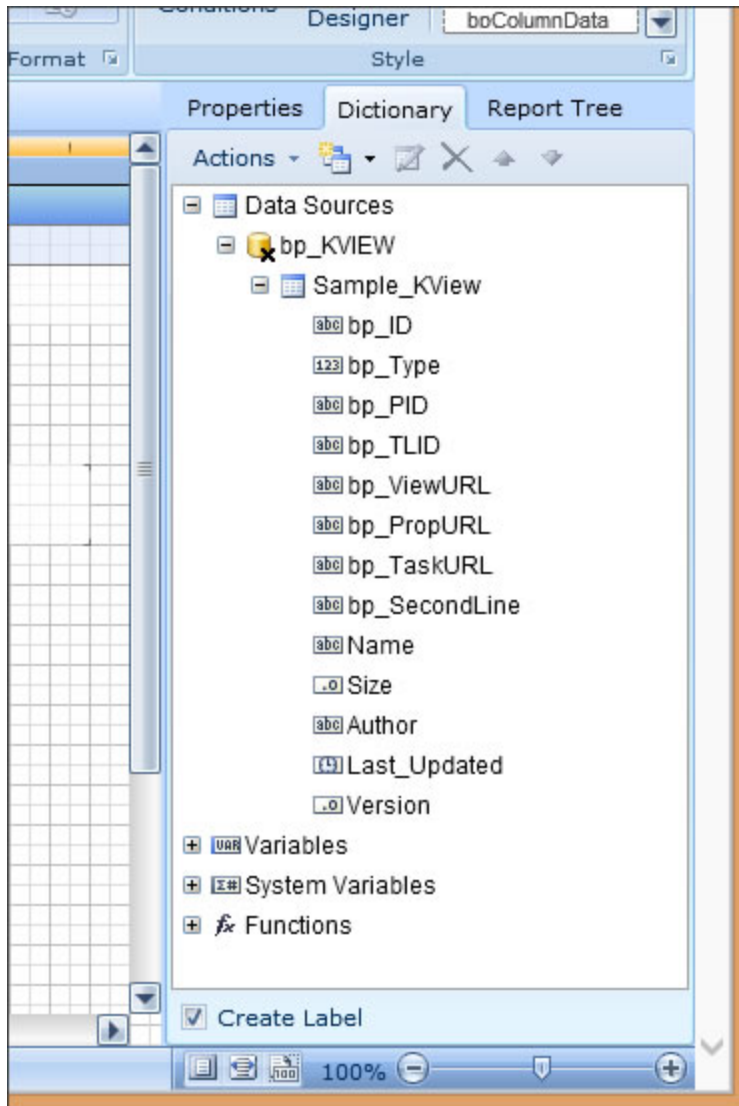
Simply select the views you wish to use in the report by checking the box next to the view's name.

## The Report Designer #

Once you have selected the data sources you wish to use in your report, you can open the report designer by clicking the Design Report button on the options screen.



Inside the report designer, the data sources you've selected will be tied to the Data Band control, and will also be displayed in the **Dictionary** tab of the report designer's options panel, located on the right side of the report designer screen.



Again, for detailed information on designing reports, refer to the [Reports Reference Guide](#).

## Included Views #

We have included multiple pre-constructed Views for use with the reporting and they could also be used in your Knowledge Views. A detailed description of each View is presented in the following topics in the [Database Guide](#):

[bpProcess](#)

[bpProcessTask](#)

[bpProcessTaskUser](#)

[bpUser](#)

[bpTaskList](#)

## Common Report Field Data Types

- **Varchar:** variable length character string using the ANSI character set.
- **Nvarchar:** variable length character string using Unicode UCS-2. This allows for multi-lingual information to be stored in character strings
- **Nvarchar (max):** an nvarchar that can reach up to  $2^{31}-1$  bytes ( $\approx 2.15$  gigabytes).

## The Globally Unique Identifier (GUID)

The Process Director database uniquely identifies every object by assigning it a special identifier known as a Globally Unique Identifier (GUID) as the ID of the object. A **GUID<sup>1</sup>** is a 128 bit, 36 character varchar value. It is a series of **hexadecimal<sup>2</sup>** numbers that are created using algorithms that ensure uniqueness. The GUID is generally represented in the format:

```
aa213c6f-a8aa-454f-a04d-30b56fd2e493
```

What you really need to know about the GUID is that, if you are linking tables together, it is what you should use.

If you search for all Process Instances of the type called “Capital Expenditure Process”, you may get any number of instances of different processes back, just because they are all named “Capital Expenditure Process.” The displayed name isn't unique. But if you search for the “Capital Expenditure Process” instances using the Timeline Instance ID of 00000109-0000-0010-8000-00AA006D2EA4, you are sure to only get that specific Timeline instance.

You can find the ID of a Process Director object by looking at the object definition's properties. For instance, in a Process Timeline, the ID is displayed in the **Options** section of the Process Timeline definition's **Properties** tab. It is displayed as the **PRID** property, as shown below.

---

<sup>1</sup>GUIDs generated from random numbers are so large that the probability of the same number being generated randomly twice is negligible. Assuming uniform probability for simplicity, the probability of one duplicate would be about 50% if every person on earth owned 600 million GUIDs. (Wikipedia)

<sup>2</sup>In mathematics and computing, hexadecimal (also base 16, or hex) is a positional numeral system with a radix, or base, of 16. It uses sixteen distinct symbols, most often the symbols 0–9 to represent values zero to nine, and A, B, C, D, E, F (or alternatively a, b, c, d, e, f) to represent values ten to fifteen. (Wikipedia)

**Documentation**

Options

Process Timeline Name Icon

Documentation

Description

Instantiated Name

{TIMELINE\_DEF\_NAME}{TIMELINE\_ATTACHMENTS,ObjectType=FORM,pre="(",post=")"}

Form for Timeline (will attach Form instance if needed) Documentation ...

Default Email Template in this Timeline ...

Timeline Script ...

Copy objects in Timeline Package to Parent

Only from this group

Copy objects from Parent Process to this Timeline

Only from this group

Remove Timeline instance after process completes  Do not show this object in 'Items I Can Run' Knowledge View

Priority 1 ▼

After a user submits a form which runs this process  
Automatically show user's next task if it is in this process

Show columns:  Index  Activity  Type  Actions  Timeline

Display timeline in Days ▼ Default activity duration Business Days ▼ Reset Predicted Values for In

**PRID: b9546531-974f-4b2e-b691-8be81b29ccd7**

## Datetime

Datetime fields in SQL Server are displayed like this: 2007-10-28 22:11:19.7030000. How you display them while creating a report is up to you. You can just use the date or the time, or just the minutes if you wanted to.

## Nullable vs non-Nullable fields

NULL = “Unknown” or “No value.” There are many fields in a database that don't have values, and some of those have never had values. Some were inserted into the table with the value of “no value” (NULL). What you need to know about null is that null isn't the same as a blank field. If a field is null and you filter out the “blank” fields... you aren't filtering out fields where the value is NULL. Because if you have an SQL statement and you say:

```
SELECT * FROM tableName WHERE fieldName <> '';
```

You aren't filtering out fields where fieldName is NULL. Because NULL isn't the same as '' (a zero length string). Instead, use something like this:

```
SELECT * FROM tableName WHERE fieldName <> '' AND fieldName IS NOT NULL;
```

You can see here that this is for a string value or character field, as I am using the character field delimiter ' (single quote).

The thing you need to know about NULL is how important they are to datetime fields. Dates that appear blank are NULL. There is no such thing as a blank datetime field, that field doesn't exist. So your filtering of date fields (and other nullable fields) is going to use the IS NULL and IS NOT NULL switches in SQL.

```
SELECT * FROM bpProcessTaskUser WHERE StartTime IS NOT NULL AND EndTime IS NOT NULL AND UserID <> '' AND UserID IS NOT NULL;
```

That statement will return all of the tasks that were assigned to a user and completed, as they have both a StartTime, an EndTime and an assigned UserID.

## Report Variables

The **Variables** tab of the report definition enables you to specify variables to use in a Report. Once a variable has been added, it can be used to specify the data returned from the Report's data source, prompt users to supply the variable's value, or accept the variable value as a parameter from a Form control on a Form that uses the **Show Report** control.

Please refer to the [Variables Tab section](#) of the reports topic to see the property values that can be configured for a Report variable.

## Variable Examples

For the examples that follow, we'll use a report that returns information about object instances in the system. This report uses an internal User Database datasource to retrieve the data from the database. This datasource stores a list of Form and Process Timeline instances. Our goal with this example is to create a variable that can be used as a parameter that we can pass to the data source to return all items, only Form Instances, or only Process Timeline Instances. We'll demonstrate two different use cases for using the variable to return the data we want: prompting for the data in the Report when it runs, and passing the variable's value to the report from a Form control. These two use cases will each require a slightly different configuration for the Variable.

### *Variable and Datasource*

The source data we'll use returns both Form and Timeline Instances. Our goal is to create a variable that will enable us to return all objects by default, then based on a value we enter, return only Forms or Process Timelines.

The first thing we need to do is create the variable. The detailed configuration for the variable depends on how we're going to get the variable's value from the user. We don't need to discuss that yet, but we do need to create the Variable and set its **Name** and **Data Type**. For this example, we'll create a string variable named **ObjectType**.



### VARIABLES

This tab allows you to create variables that can be used in the Report or in the associated Knowledge Views.

[Add Variable](#)

Name	Variable Type	Default Value	Prompt?	Prompt Text
ObjectType	String		<input type="checkbox"/>	

In the sections below, we'll come back to configure the **Default Value**, **Prompt**, and **Prompt Text** values, depending on our use case. Form now, though, our variable has been created, so we need to update the Report definition, then navigate to the **Data Sources** tab.

### DATA SOURCES

Knowledge Views | **Data Sources** | Database Views

This tab allows you to select external Data Sources to include in the report. You may add any number of Data Sources, and optionally filter tables from the report.

Please Choose Data Source

Show Advanced SQL

Report Data (1 tables selected)

Enabled	Table Name
<input type="checkbox"/>	bpUser
<input checked="" type="checkbox"/>	Training_Report_Data

In this case, our datasource returns data from a table named **Training\_Report\_Data**. By default, it returns all data in that table, so we'll need to modify this data source to use our variable. To do so, we'll need to check the property labeled, **Show Advanced SQL**, to edit the SQL statement that returns the data from the table.

### DATA SOURCES

Knowledge Views | **Data Sources** | Database Views

This tab allows you to select external Data Sources to include in the report. You may add any number of Data Sources, and optionally filter tables from the report.

Please Choose Data Source

Show Advanced SQL

Report Data (1 tables selected)

Enabled	Table Name	SQL
<input type="checkbox"/>	bpUser	SELECT * FROM [bpUser]
<input checked="" type="checkbox"/>	Training_Report_Data	SELECT * FROM [Training_Report_Data]

As you can see, this SQL statement returns all of the table data. We'll need to add a WHERE clause to this SQL statement to return the specific data that matches the value provided by the user.

```
WHERE [Object Type] LIKE '%{$VAR:ObjectType}%'
```

The **Object Type** field for each record will contain one of two values: "Form Instance" or "Timeline Instance".

Using the **LIKE** operator in our where clause, as well as the % wildcard character, will enable us to enter a filter value such as "form" to return Form instances, or "time" to return Timeline instances. If no value is provided for the variable, this syntax will return all records by default.

The `{ $VAR:ObjectType }` system variable uses the \$ encoding character to ensure that the value we pass to the database is SQL-safe, and identifies the **ObjectType** variable we created on the **Variables** tab as the source of the data to pass as a parameter to our SQL statement.

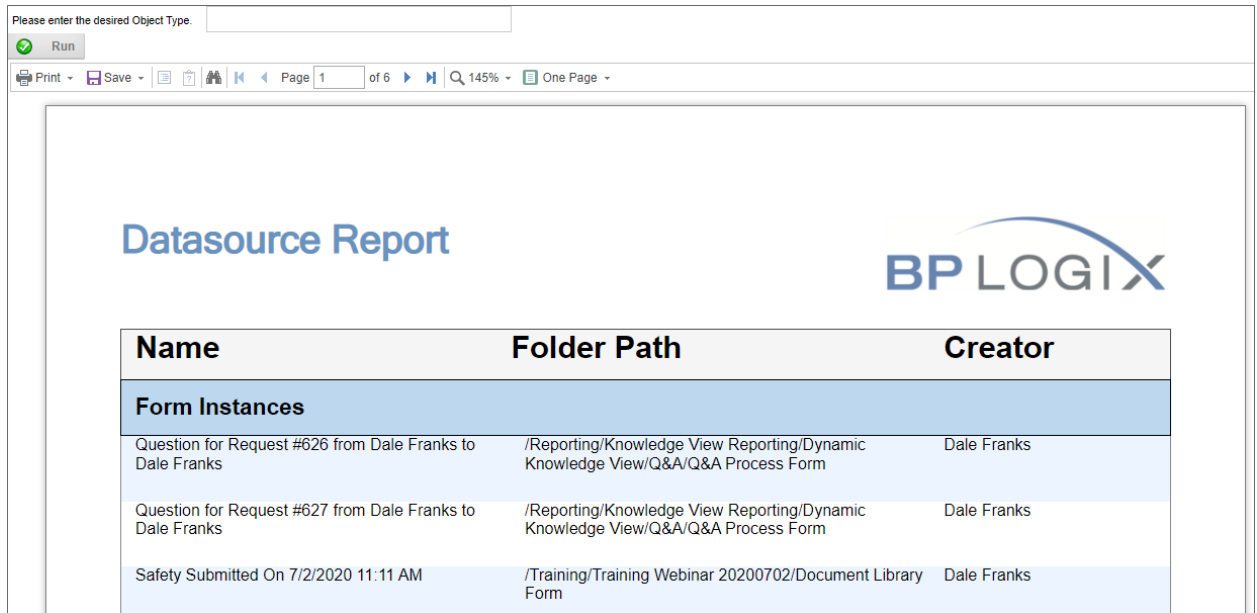
Once we've configured the **Data Sources** tab with the edited SQL statement, we'll need to update the Report definition again to save our changes.

Now, we'll need to return to the **Variables** tab to configure our **ObjectType** variable properly for the use cases below.

### Prompting for the Variable

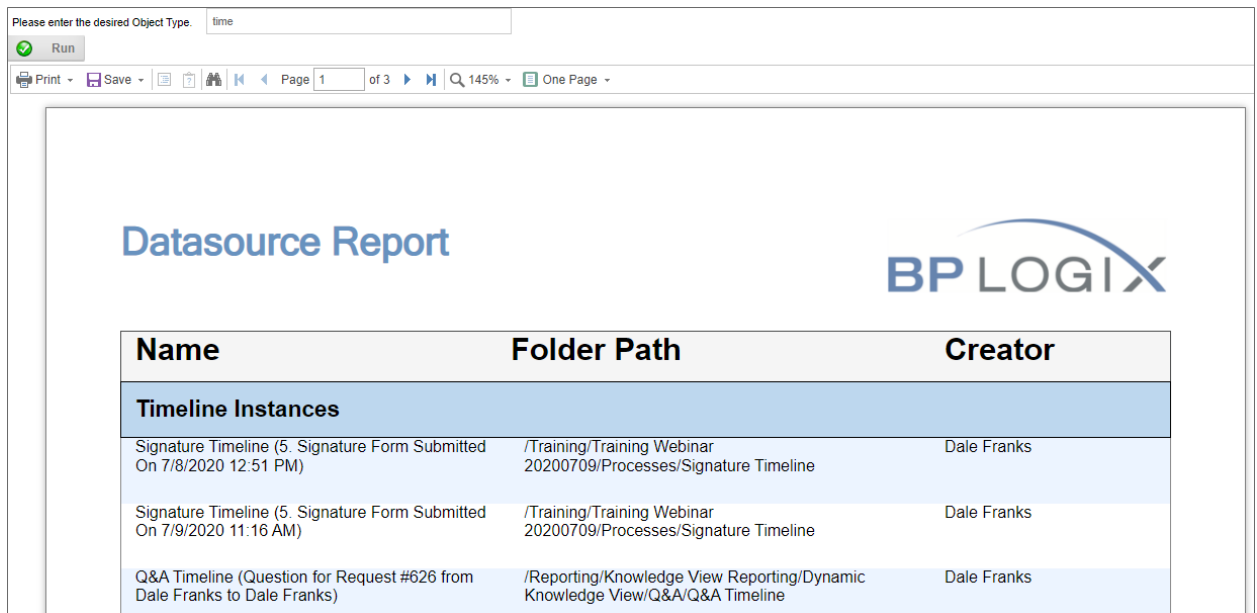
A report can prompt the user to provide a variable's value when the report runs. To make this work, we'll need to configure the **ObjectType** variable to prompt the user.

For this use case, we need to check the **Prompt** property to require the report to prompt the user for the value, then add the **Prompt Text** we want to use to provide instructions to the user. Once we've done so, we can update the report and run it.



When the Report first opens, no value has been supplied for the ObjectType variable, so the report returns all objects, and is 6 pages long.

At the top of the report, the prompt we configured is displayed, along with a text box that enables us to enter the value we want to pass as a parameter. A **Run** button will, once we enter the desired value, run the report to return the data we specify. If we enter the value "time" in the text box, then click the **Run** button, we'll get the specific data that matches our parameter.



As we can see, the report is now only three pages long, and returns only Timeline instances.

### ***Passing a Parameter from a Form***

If a report is displayed on a Form using the **Show Report** control, we'll need to configure the Form to provide the prompt from a Form control, rather than from the report itself.

The screenshot displays a form configuration interface. On the left, there is a form with an 'Object Type:' label and an empty input field. Below the input field is a 'Show Report' button with a colorful icon. On the right, a 'Show Report Control' panel is open, showing three tabs: 'Show Report', 'Comments', and 'Field Properties'. The 'Field Properties' tab is active. It contains the following fields: 'ToolTip' (empty), 'Friendly Name' (containing 'Report 2'), 'Default Value' (containing 'Datasource Report' and a dropdown menu with 'Content Item' selected), 'Set Readonly Options' (dropdown), and 'Set Display Options' (dropdown).

In this simple example, our form has two controls. An **Input** control named **ObjectType**, and a **ShowReport** control named **Report2**. The **ObjectType** field is configured as an **Event Field**, to prompt the report to reload any time we change the field's value. The **Report2** field has the **Default Value** of its Field properties configured to show the **Datasource Report** definition from the **Content Item** selection.

Now, we have to set up the control properties for **Report2** to pass the value of the **ObjectType** control to the Report.

### Show Report Control

Show Report

Comments

Field Properties

**Name**

**Text**

**HTML Height**

**HTML Width**

**Type**

**Image URL**

**QueryString Params (Separated by NewLines)**

```
qs_Obj={:ObjectType}
```

In the **QueryString Params** property of the Report2 control, we'll create a query string to pass to the report, using the syntax:

```
qs_Obj={:ObjectType}
```

This syntax creates a QueryString named **qs\_Obj**, with a value set to the value of the **ObjectType** control, via a Form field System Variable. With the QueryString set, we need to configure the **Variables** tab of the report definition to accept it. So, we'll save and close the Form, open the Report, and edit its **Variables** tab appropriately.

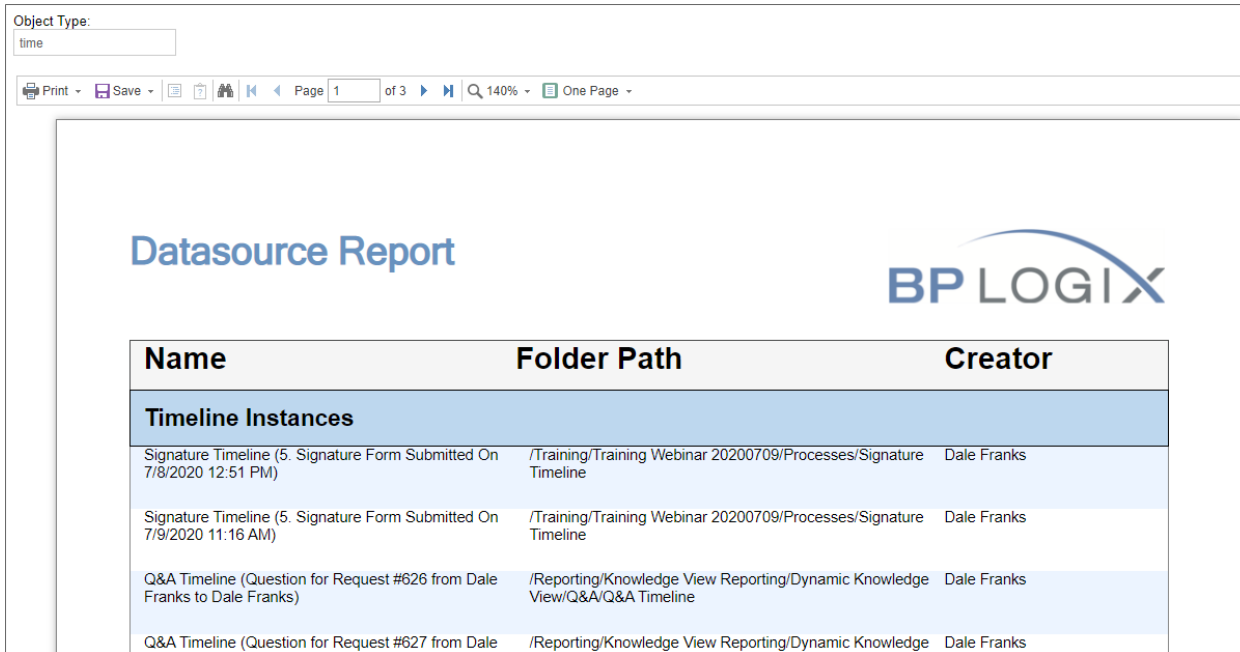
### VARIABLES

This tab allows you to create variables that can be used in the Report or in the associated Knowledge Views.

Name	Variable Type	Default Value	Prompt?	Prompt Text
ObjectType	String ▼	{VAR:qs_Obj}	<input type="checkbox"/>	

In this use case, we need to set the **Default Value** of the variable to use the the **qs\_Obj** QueryString value that the Form will pass to the report. Since a QueryString is a variable, we'll use the VAR system variable to supply the default value, i.e., **{VAR:qs\_Obj}**. We can now save and close our report, and run the Form.


When the Form first opens, it'll display the report showing all of the objects, but if we enter "time" into the **ObjectType** field on the form, the Report will, once again, show us only Timeline instances.



Object Type: time

Print Save Page 1 of 3 140% One Page

### Datasource Report



Name	Folder Path	Creator
<b>Timeline Instances</b>		
Signature Timeline (5. Signature Form Submitted On 7/8/2020 12:51 PM)	/Training/Training Webinar 20200709/Processes/Signature Timeline	Dale Franks
Signature Timeline (5. Signature Form Submitted On 7/9/2020 11:16 AM)	/Training/Training Webinar 20200709/Processes/Signature Timeline	Dale Franks
Q&A Timeline (Question for Request #626 from Dale Franks to Dale Franks)	/Reporting/Knowledge View Reporting/Dynamic Knowledge View/Q&A/Q&A Timeline	Dale Franks
Q&A Timeline (Question for Request #627 from Dale	/Reporting/Knowledge View Reporting/Dynamic Knowledge	Dale Franks

## Stream Actions Object

Users of Process Director v5.26 and higher have access to the Stream Actions object. This object will read a recordset stream from a Datasource, and enable you to submit a form instance for each record in the stream. Stream Actions can be scheduled and run on a recurring basis to import data on any desired schedule.

A common use case for the Stream Actions Object would be to import data from an Excel spreadsheet, external data source, or Business Value, then create a new form instance for each row in the recordset, fill out fields on the form instance, then submit it to begin a process for each record.

## Properties Tab #

The **Properties** tab provides the basic object configuration settings.

The following Properties are configurable.

**Name:**

The name of the Stream Actions object.

**Description:**

The description that you'd like to display in the second line of the object's entry in the [Content List](#).

**Stream Action is currently:**

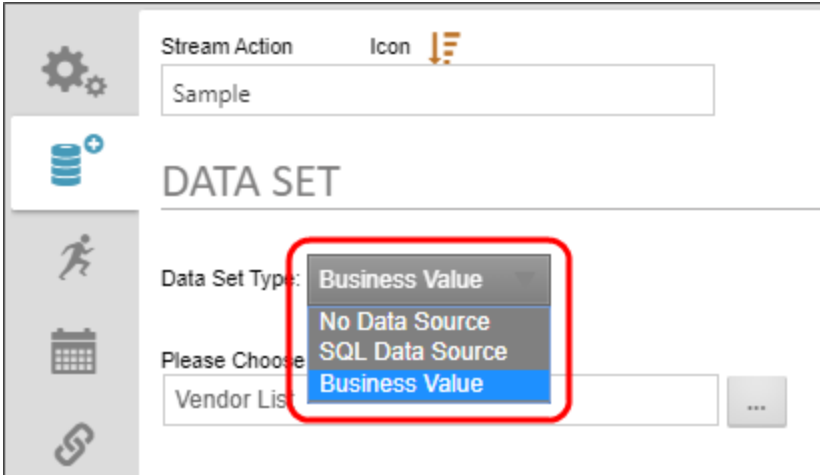
For Stream Actions that you wish to schedule, you can activate or deactivate the Stream Actions object from running by selecting the appropriate radio button.

**Group in Value Picker:**

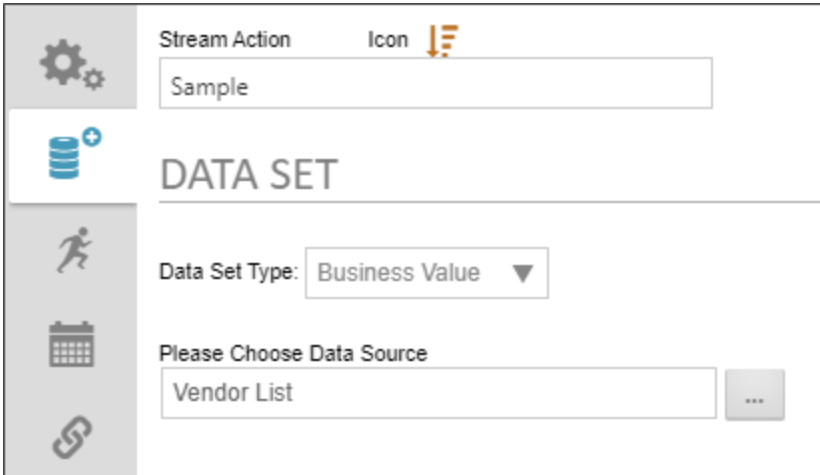
Enables you to set the category in which this Stream Actions Object will appear in the dropdown selector of the [Choose System Variables](#) dialog box.

**Data Set Tab #**

The **Data Set** tab enables you to configure the data source you wish to use for this Stream Actions object. The configuration properties available on this tab will depend on the **Data Set Type** you select to use. You can select either a Business Value as a data source, which is the most common user case, but you can also configure a SQL data source directly as well.



### Configuring a Business Value Data Source



When you set the **Data Set Type** to Business Value, you can use the **Please Choose Data Source** Object Picker to browse for the Business Value object you wish to use as the data source.

### Configuring a SQL Data Source



The screenshot shows the 'DATA SET' configuration panel. At the top, there's a 'Stream Action' dropdown menu with 'Sample' selected. Below it is the 'DATA SET' section. The 'Data Set Type' is set to 'SQL Data Source'. Underneath, there's a 'Please Choose Data Source' field with 'Vendor List' selected, followed by an ellipsis button and a 'Test Connection' button. Below that is an 'SQL Command' text area. To the right of the text area is a '[Select Table to Inspect Columns]' dropdown and a 'Show Columns' button. At the bottom left of the panel is a 'Test Query' button.

When you set the **Data Set Type** to SQL Data Source, the following properties are configurable.

**Please Choose Data Source:** An Object Picker that enables you to choose the Datasource object that connects to the database.

**Test Connection:** Once the Datasource object has been selected, you can click this button to test the connection to the specified database.

**SQL Command:** Enables you to type in the SQL Statement that is required to return the desired data.

**Select Table.../Show Columns:** Enables you to select a table in the Datasource to which you've connected, then show the columns contained in the selected table. This will assist you in writing your SQL statement, in case you don't which columns you want to include.

**Test Query:** Once you've written the SQL Statement, you can test its operation by clicking this button.

## Start Process Tab #

The **Start Process** tab enables you to specify the form to use to import the data and start a process when the form is submitted by the Stream Actions Object. When the object runs, it will import the data, create a new form for each row of data in the dataset, then immediately submit the form you specify here. If the Form Definition is configured to start a process when a new form instance is submitted, the process will begin immediately, with the data you've imported via the Stream Actions object.

The screenshot shows the configuration interface for a stream action. At the top, there is a 'Stream Action' dropdown set to 'Sample' and an 'Icon' dropdown set to a list icon. Below this is a 'START PROCESS' section with a 'Submit Form and start Process when:' label and a link to 'Click to create condition...'. The 'Form Definition' section has a dropdown set to 'Purchase Request (Online)'. The main area is a table with two columns: 'Form Field' and 'Value'. The 'Form Field' column contains dropdowns for 'VendorName', 'VendorAddress1', 'VendorAddress2', 'VendorCity', 'VendorState', 'VendorZip', and 'VendorContact'. The 'Value' column contains corresponding system variables: 'Variable VendorName', 'Variable Vendor Address 1', 'Variable VendorAddress2', 'Variable VendorCity', 'Variable VendorState', 'Variable VendorZip', and 'Variable VendorPOContact'. Each row has a three-dot menu, up/down arrows, and a red 'X' icon. At the bottom, there is an 'Add Field Mapping' button and a 'Distinct Dataset Column:' label with an empty text box.

Form Field	Value
VendorName	Variable VendorName
VendorAddress1	Variable Vendor Address 1
VendorAddress2	Variable VendorAddress2
VendorCity	Variable VendorCity
VendorState	Variable VendorState
VendorZip	Variable VendorZip
VendorContact	Variable VendorPOContact

The following properties are configurable.

### Submit form and start process when

This property invokes the [Condition Builder](#) to enable you to set whatever conditions you desire to determine whether the stream for each record should run.

### Form Definition

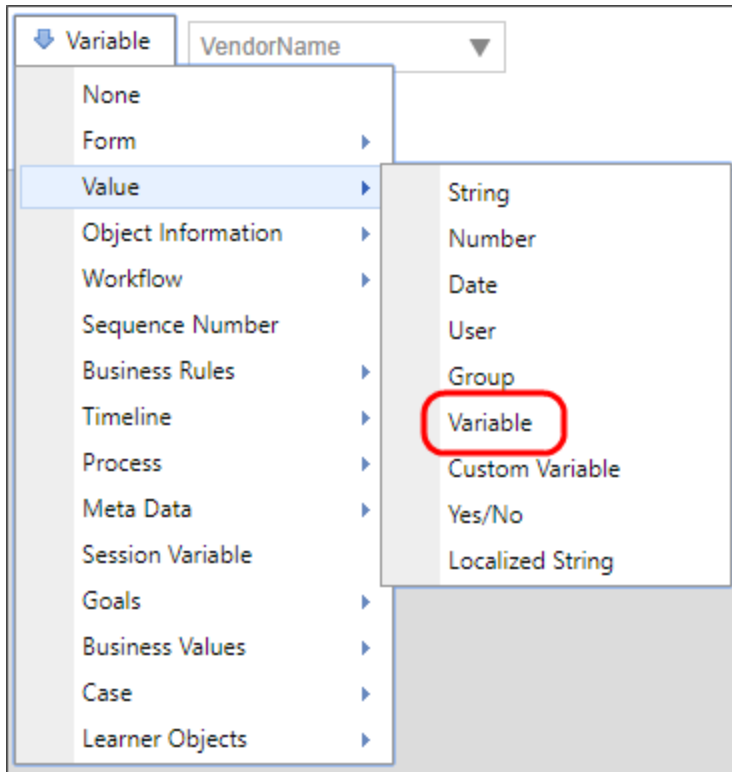
This Object Picker enables you to select the form definition to use when creating a form instance for each record in the dataset stream.




### Form Field

A dropdown containing all of the available form fields on the specified form. Select the field you want for each field in the stream's record.

### Value

A System Variable Picker that enables you to choose the record field to import into the form field. In the Picker, you can select the Variable type to display a dropdown list containing all of the fields in the stream record. The dropdown list will populate automatically with the fields in the stream record.



An **Add Field Mapping** button enables you to add as many **Form Field/Value** rows as necessary. Additionally, Up () , Down () , and Delete () buttons on each row enable you to order or delete rows.

## Schedule Tab <#>

The **Schedule** table enables you to create the schedule you'd like to use to import the Stream data on a periodic basis.

The screenshot shows a configuration window for a Stream Action. At the top, there is a 'Stream Action' dropdown menu with 'Sample' selected and an 'Icon' dropdown with a downward arrow. Below this is a 'SCHEDULE' section. A dropdown menu is set to 'Run Stream Action on a Schedule'. The 'Repeat Every' field is set to '1' with a 'Days' dropdown. The 'Repeat Interval Starts At' field is empty, with a 'Days of Lead Time' field set to '0'. The 'Scheduler Period End' field is empty. There is an unchecked checkbox for 'Limit to Business Hours Only' and a 'Process Now' button at the bottom.

The following properties are configurable.

### Run Stream Action.../No Automatic Stream Action

This dropdown enables you to turn scheduling on or off for the Stream Actions object.

### Repeat every

This text/box, dropdown combination enables you to specify the number of time intervals, and the type of time interval (i.e., hours, days, months, etc.) between each run of the Stream Actions object. The default is to run every 1 day.

### Repeat Interval Starts At

This DateTime Picker enables you to specify a start date for the Stream Actions Object to begin the a scheduled import frequency.

### Days of lead time

This text box enables you to specify the number of days lead time you need to complete the process. For instance, let's say the processes that are started by the Stream Actions Object take 5 days to complete, and all need to be completed by the 15th of every month. You might set the **Repeat Every** property to run every month, and set the **Repeat Interval Starts At** property to the 15th of the coming month. If you set the lead time to 5 days, the data will be imported and the processes started five days prior to the 15th.

### Scheduler Period End

This DateTime Picker enables you to set an end date to stop running the Stream Actions Object's scheduled runs.

### Limit to Business Hours Only

Checking the property will ensure that the Stream Actions object runs only during the business hours you've set in the system.

### Process Now

Clicking this button will run the Stream Actions object immediately.

## Workflow



The Workflow object is the legacy process model used in early versions of Process Director. BP Logix recommends the use of the [Process Timeline](#) object, and not the Workflow object. The Workflow object remains in the product for backwards compatibility, but doesn't receive any new functionality updates, other than required bug fixes. No new features have been added to this object since Process Director v4.5. All new process-based functionality is solely added to the Process Timeline.

Workflow is a complicated term. In Process Director, there is a Workflow object that enables you to create a process definition. The Workflow Object is an older method of creating process definitions that pre-dates the Process Timeline.

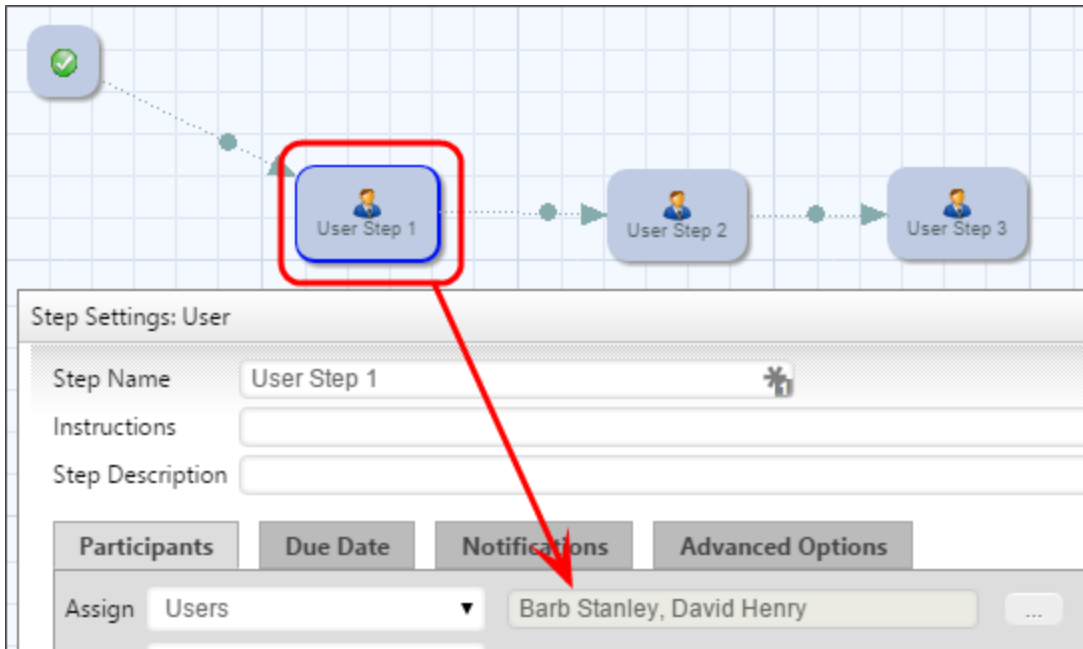
In a larger sense, "Workflow" is also a general process management term that is largely synonymous with the term "process". A Workflow is the automation object of a business process in which documents, information or tasks are passed from one participant(s) to another for action. A Workflow is made up of many functions and activities such as a review process, [Task Lists](#), notifications, alerts/triggers, reminders, context sensitive tasks, an approval process, status/tracking, due dates and reporting.

### What is a Workflow Definition?

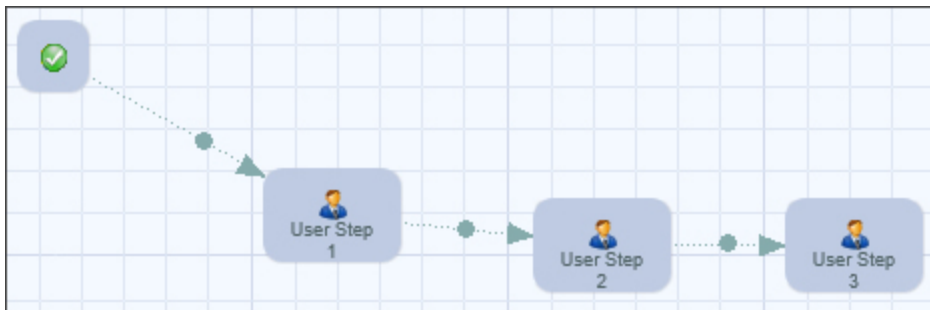
Workflow definitions in Process Director are a series of logical steps, each with a specific task and assigned participants. The Workflow defines the path or route that a Form or document must take. Each step in the Workflow path defines the task type, the participants, and the rules that govern how the Workflow will advance or transition to the next step.

### Users vs. Groups

Users and/or groups can be assigned to a step in a Workflow. When a Workflow definition is run, the group will be expanded and all users that are members of the specified groups will be added to the appropriate Workflow Steps. The Process Director Workflow engine supports parallel and serial reviews. To set up a parallel review process, add multiple users or groups to the same Workflow Step.



To establish a serial review process, add each user or group to a separate step.



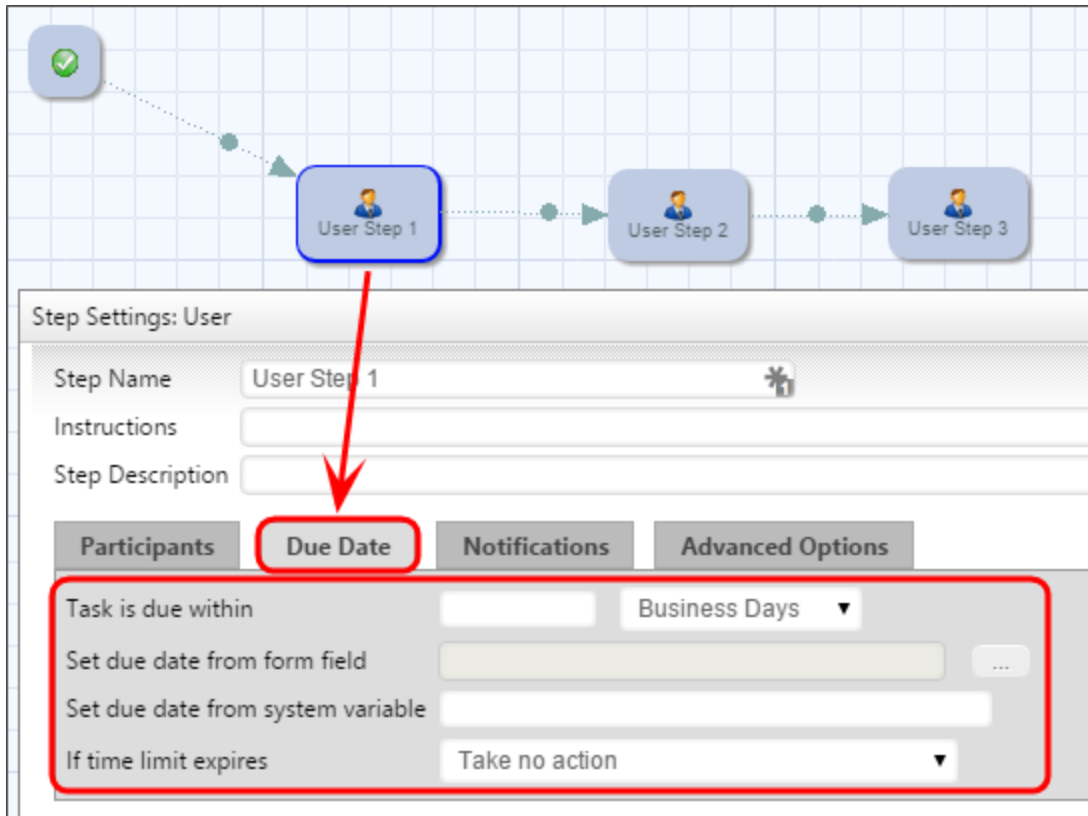
Generally a review will be a combination of a parallel and serial process.

## Task Email Notifications

As a Workflow advances to a step, the assigned users are automatically notified using your corporate email system. A custom email can be sent that provides users with special instructions that are relevant to this specific task.

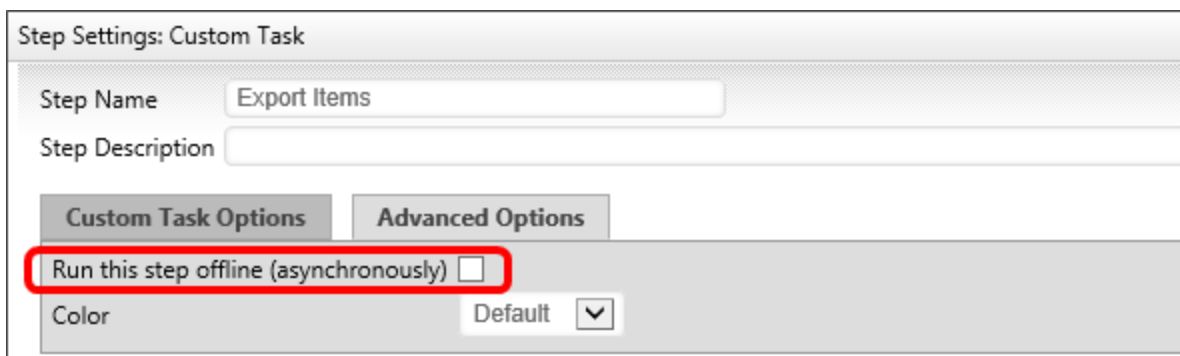
## Due Date Management

Due date management functions allow due dates to be set for the entire Workflow, as well as for each step in a Workflow. Periodic email reminders can be automatically sent to users that have not completed their task. When a due date expires for a step the due date escalation rules determine if the system should automatically advance the Workflow to the next step, notify another user, start a new Workflow process or jump to another step within the Workflow.



## Asynchronous Operation

A Workflow Step of the Custom Task, Script or Sub-Process types can be configured to run asynchronously/offline by checking the check box labeled "Run this step offline (asynchronously)" in the Advanced Options tab. Setting the operation to run asynchronously can be used to prevent a long-running, machine-centric task in the Workflow from hanging a user session while the processing occurs. The step will be run in a different context than the current user that caused the transition of the step. Additionally, if you have a [Rendering Server](#) enabled, the asynchronous processing will be conducted by the Rendering Server when the asynchronous option is checked.



Please note that, when this option is selected, if the user's task is followed (after one or more intervening "background" tasks) by another task assigned to the same user, the current behavior in which the window

remains open and is refreshed with the Form for the subsequent task will no longer be seen. Instead, the user will have to click the appropriate link in their [Task List](#) or email notification to open the new task.

On some systems, when starting a subprocess using the asynchronous option, the system can mark the calling task as complete before the called subprocess completes. This may be especially true if the subprocess contains complex rendering operations. To avoid this, a wait time, in seconds, can be set using the [nAsyncSubProcessWaitSecs](#) variable in the Custom Vars file. The default setting for this variable is 5 seconds.

## Creating Workflow Definitions



The Workflow object is the legacy process model used in early versions of Process Director. BP Logix recommends the use of the [Process Timeline](#) object, and not the Workflow object. The Workflow object remains in the product for backwards compatibility, but doesn't receive any new functionality updates, other than required bug fixes. No new features have been added to this object since Process Director v4.5. All new process-based functionality is solely added to the [Process Timeline](#).

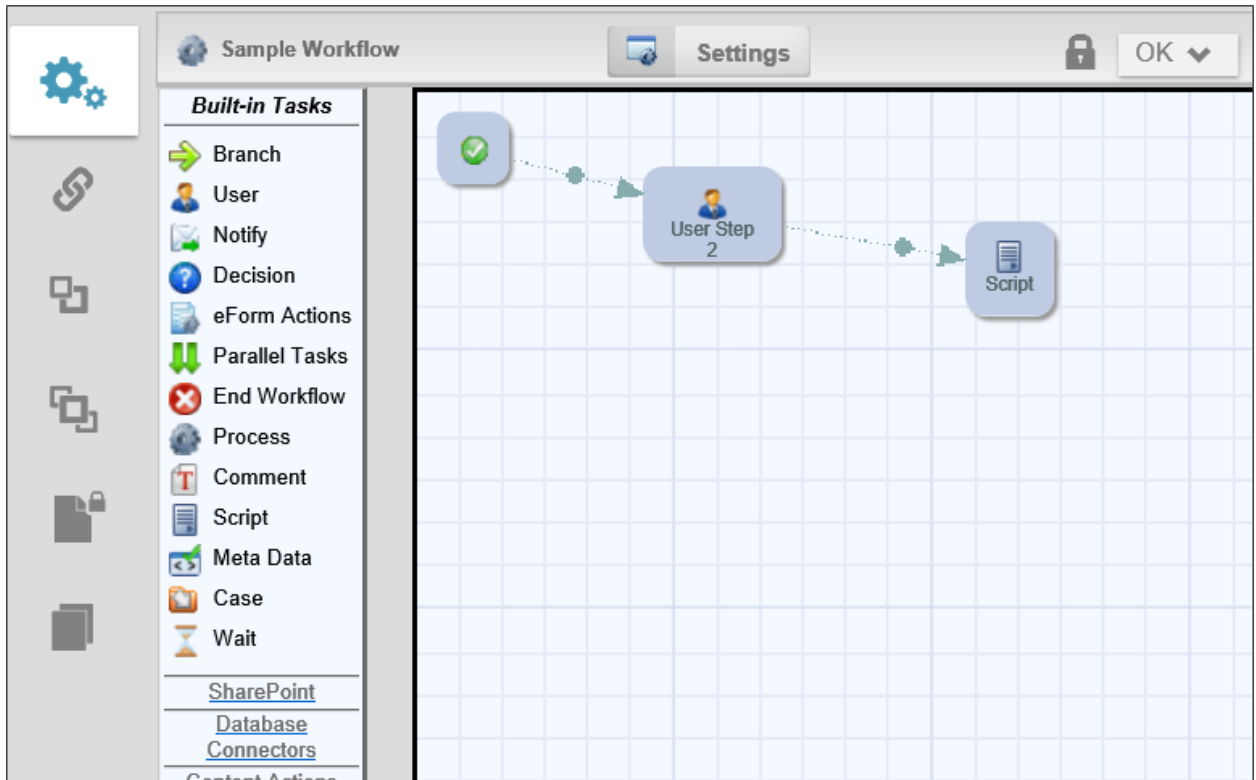
Workflow definitions can be created and modified by the business users. Click the [Content List](#) tab in the Process Director navigation bar to view the [Content List](#). You must have Modify permission in the folder you are viewing to be able to create a new Workflow definition. Use the [Create New](#) menu item and select [Workflow Definition](#) from the list.

Workflow definitions can be stored anywhere in the [Content List](#), under any folder structure.

### Workflow Settings #

To configure the Workflow definition settings, click on the Settings icon in when viewing the Workflow definition. This will display a dialog that allows the Workflow information to be set.





## Workflow Name and Description

**Workflow Settings**

Workflow Name

Description

The Workflow definition name and description are important because this is what a user will see when selecting a Workflow to run. The description should describe why and when this Workflow definition should be run.

## Workflow Options

**Workflow Settings**

Workflow Name

Description

**Workflow Options**

**Advanced Options**

Priority

After a user submits a form which runs this process

Remove workflow instance after workflow completes

Do not show this object in 'Items I Can Run' Knowledge Views

Workflow Icon

### Priority

The Workflow priority is used to show participants the importance. The higher priority items are displayed first in a user's [Task List](#). The Workflow's Priority column can be resorted in the user's [Task List](#).

### After a user submits a form which runs this process

In some cases, a user may be assigned to two or more activities in a row. The default behavior for Process Director is to immediately display the Form to the user to perform the second task once the first task is completed. This may be confusing, as the user generally expects the Form to close when a task is completed. This property enables you to change the default behavior so that, if a user is assigned to two activities in a row, the Form won't immediately reload to complete the second task. To perform the second task, the user will have to manually re-open the form from the [Task List](#).

### Remove Workflow instance after Workflow completes


When a Workflow is started it is stored under the Workflow definition in the [Content List](#). Completed Workflows will remain there until deleted or moved. If a completed Workflow isn't needed for reporting or auditing, you can have them removed automatically when they complete. To automatically delete completed Workflows set the Remove inactive Workflow after it completes flag.

To delete a Workflow instance manually, click on the Workflow definition in the [Content List](#). From there, two options should appear under the Workflow in the Content List: Active Workflows and Completed Workflows. Click on Completed Workflows, select the Workflow you want to delete, and click on the "Delete" action.

### Do not show this object in 'Items I Can Run' Knowledge Views

Checking this option prevents users from accessing the Workflow from the [Items I Can Run](#) Knowledge Views.

### Workflow Icon

A custom icon can also be configured for this Workflow definition. This icon will be displayed for all running Workflows for this definition (e.g. [Task List](#), [Content List](#), Knowledge View, etc.). To change the icon used for this Workflow definition, click on the  icon.

## Advanced Options

The screenshot shows the 'Workflow Settings' dialog box with the 'Advanced Options' tab selected. The 'Workflow Name' is 'Employee Workflow' and the 'Description' is 'This workflow routes an employee through the new hire process.' The 'Instantiated Workflow Names' field contains the text '{WORKFLOW\_DEF\_NAME}{WORKFLOW\_ATTACHMENTS,pre="(",post:'. Below this are fields for 'eForms Associated with Workflow (optional)', 'Default Email Template in this Workflow', and 'Workflow Script', each with a '...' button. There are also two checkboxes for 'Copy objects in Workflow Package to Parent' and 'Copy objects from Parent Process to this Workflow', each with an 'Only from this group' field. At the bottom, the WFID is '43cfe9a6-466c-4491-afbe-58202f9b7cef'.

### Instantiated Workflow Names

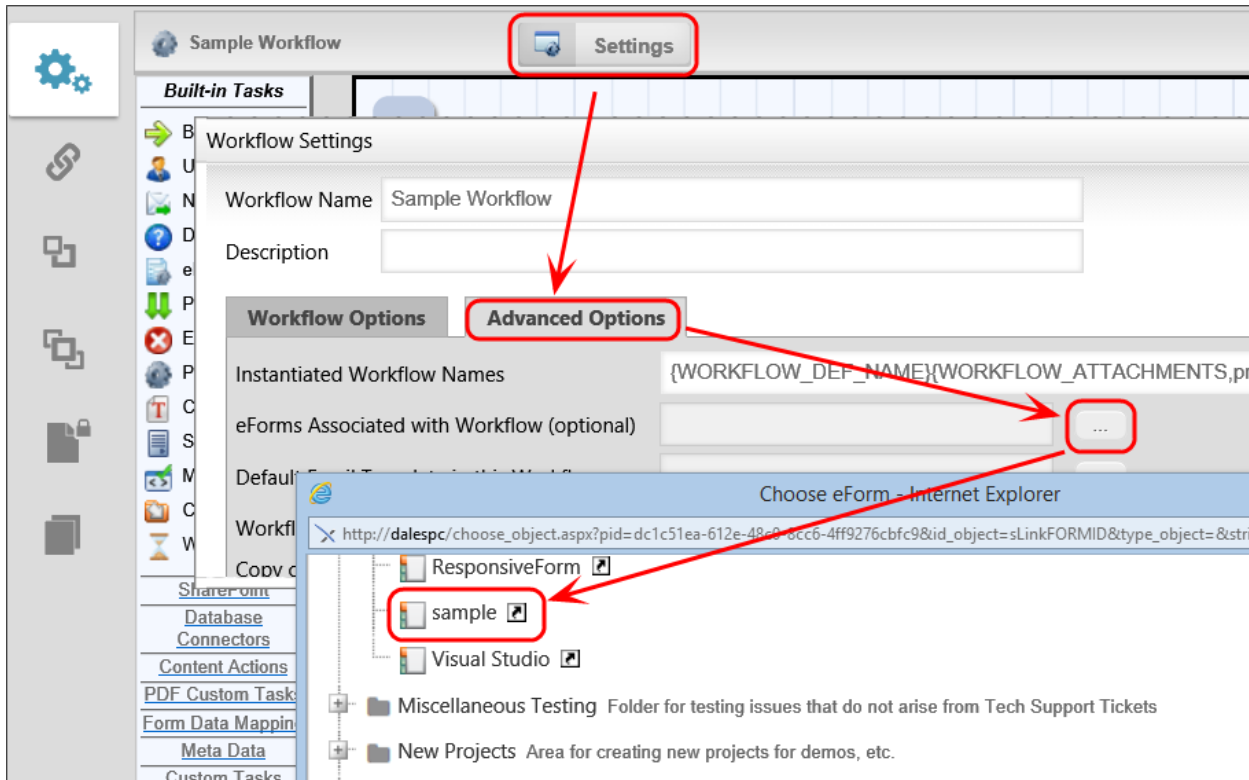
This field contains an optional name that should be used when a Workflow is started (i.e. instantiated). The default name of a running Workflow is the same name as the Workflow definition. This name can contain system variables using the {sysvar} tag. This allows the running Workflow names to contain more meaningful information (e.g. PR Review – started by {CURR\_USER}). The running Workflow name is re-evaluated after each Workflow Step completes allowing variable data (e.g. Forms field values) to be used to construct the name. Refer to the section named System Variables in this document for more information.

When using an instantiated Workflow name, a user won't be able to override the name when starting a Workflow against an object.

### Form Associated with Workflow

You can optionally link a Workflow to a specific Form definition in the [Content List](#). When this is configured you can choose a form field from a dropdown list of all fields in that form instead of choosing a form then a form field.

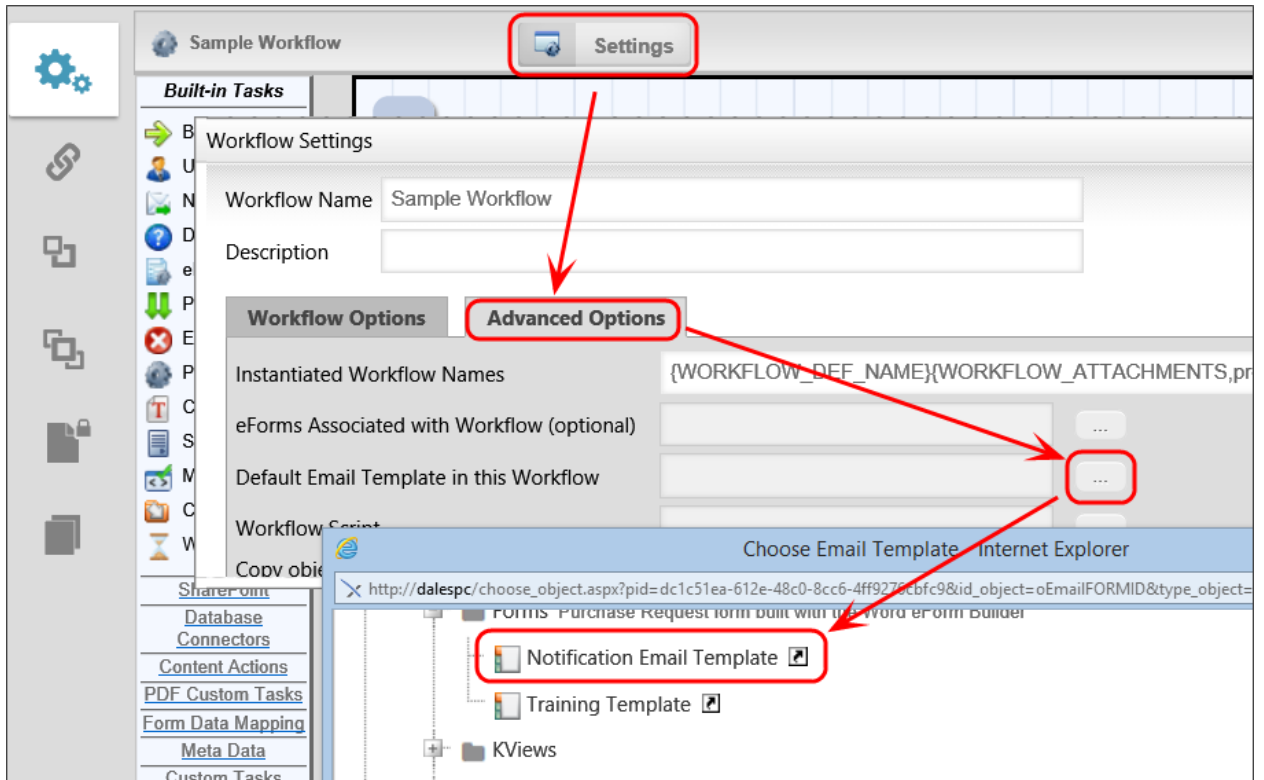
To do this, click on Settings, then on the Advanced Options tab in the window that appears. Use the Pick List [. . .] button to select the Form you want associated with the Workflow.



### Default Email Template in this Workflow

This is an optional default email template to use when sending email notifications to users in a step. If an email template isn't specified in a step, this default email template will be used. If no default email template is assigned, the system default email template will be used that was shipped with Process Director. For more information on email templates refer to the section named Using Email Templates in this document. Email Templates are stored in the [Content List](#) as a Form uploaded as an Email Template Form.

To set an email template, click on "Settings" then go to the Advanced Options tab in the Window that appears. Use the Pick List [ . . . ] button, then select the email template you want to use.



### Copy Objects in Workflow Package to Parent

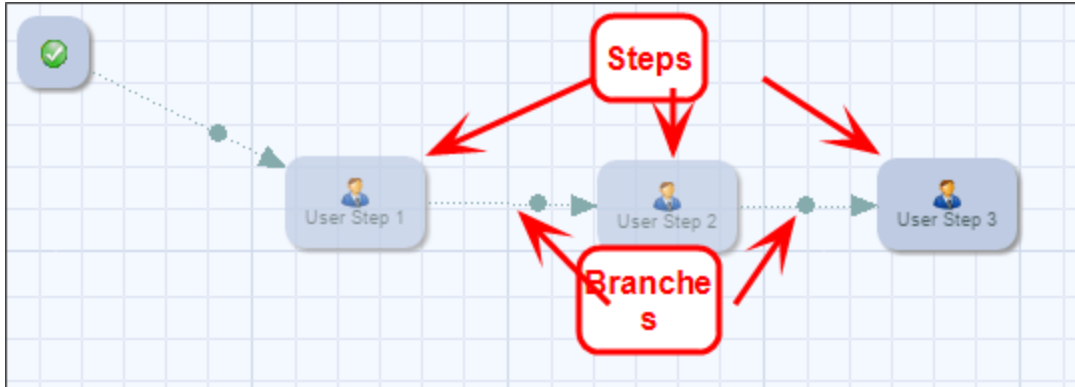
If checked, this option will copy objects in this Workflow package to the package's parent. You can optionally limit the objects copied by group. The parent process may be either a Workflow or a Process Timeline.

### Copy Objects from Parent Process to this Workflow

If checked, this option will copy objects from this Workflow's parent process to this Workflow. You can optionally limit the objects copied by group. The parent process may be either a Workflow or a Process Timeline.

### Workflow Steps #

A Workflow is made up of a series of steps. Each step in a Workflow identifies the task to be performed and the users that should perform the task. Steps are run from the beginning (start step) and follow the branches (route). Steps can be easily added, moved and deleted from a Workflow definition and the route or path is defined using branches between steps.



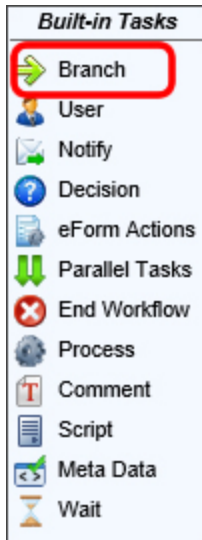
## Workflow Task Toolbar

The Workflow builder has a toolbar displayed on the left side of the screen. This contains all of the Workflow tasks available to you. Each task type identifies the type of Workflow function to be performed for this step and provides the participating users with different capabilities. Tasks are grouped in the toolbars (e.g. Built-In Tasks). To view tasks in the different groups click on the group name (e.g. Other Tasks). To use a task, click on the icon, drag it to the right and drop it on the Workflow palette.

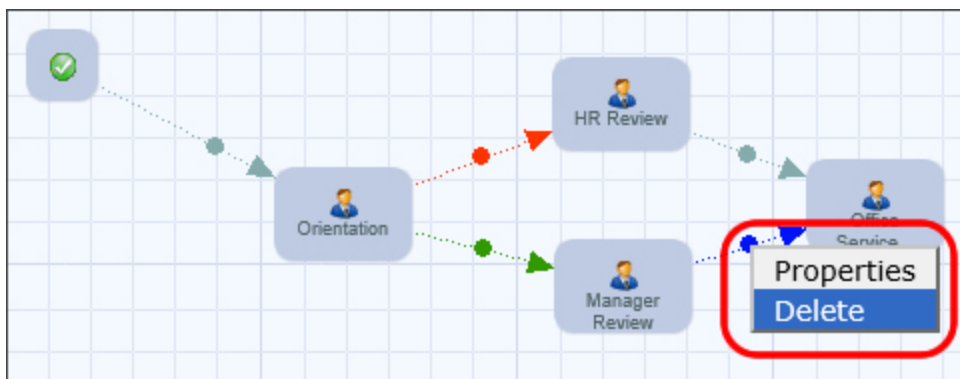
The built-in Workflow tasks are in the Built-in Tasks groups in the toolbar. If any Custom Tasks are installed, they'll appear as new groups in the toolbar. The group names are the folder names where the Custom Task is installed in the [Content List](#).

### Branch Steps (the route)

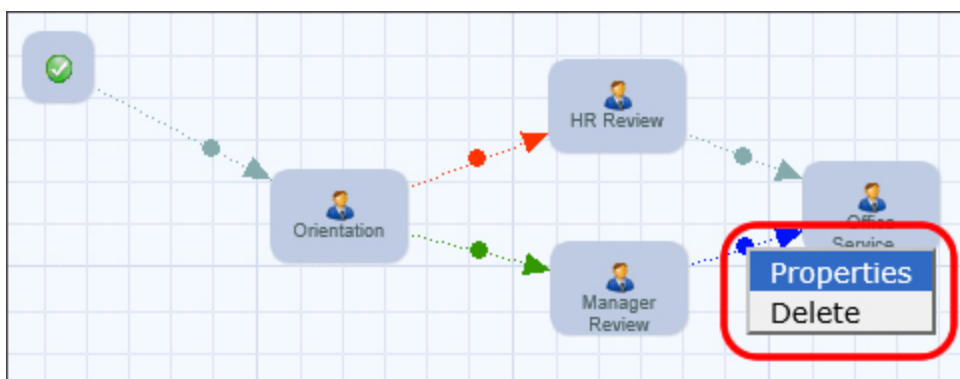
To define the route that the Workflow will take, you must connect the steps. This is done by using the Branch in the Built-in Tasks group. To branch two steps together, click on the Branch in the toolbar. Then click on a step in the Workflow and drag it to the step you want to branch to, lifting the mouse over that step. This will draw a line between the two steps showing the direction of the flow.



To delete the line (branch), right click on the handle in the line (displayed as a circle in the middle of the line), then select "Delete" from the pop-up menu.



When using the Branch task you can branch from a step to multiple steps which allow you more flexibility with your process. Each branch that is created has branch settings. Right click on the handle in the line and select Properties.



### Branch Settings

Branches in a Workflow can have the following information associated with them.

Branch Settings

Branch Name

Branch Description

Conditions Options

% of participants that choose this branch

# of participants that choose this branch

Confirmation prompt for this branch

Take this branch when [Click to create condition ...](#)

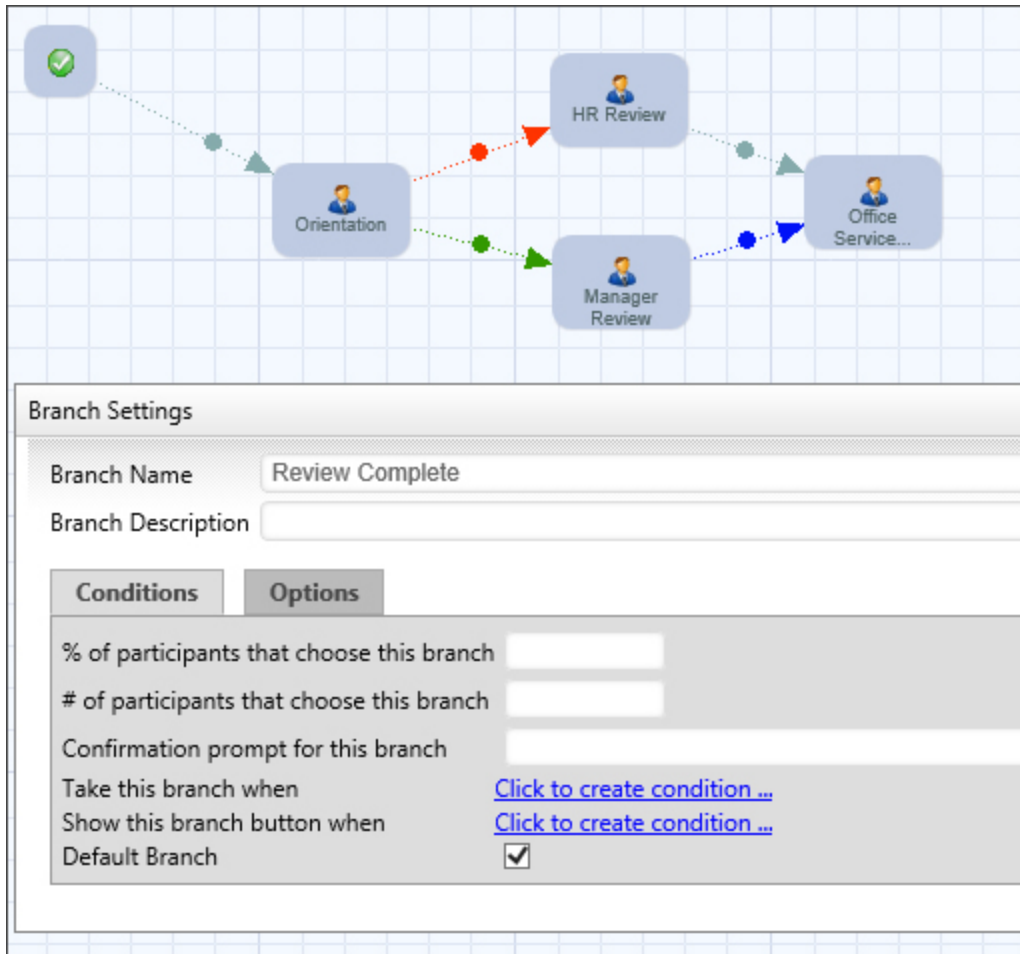
Show this branch button when [Click to create condition ...](#)

Default Branch

### Branch Name

The first field on this dialog identifies the name of the branch. This information is important for the users because it is used to display a button on the Form with the branch name as the label. This will determine the route the user will take in the Workflow process. The button will only display during the step the branch is connected from. For example: The image below displays the branch name as Reject which will display a button, labeled “Reject”, on the Form in the Form Approval step.





You can leave this field blank to display the default button “Complete Task”.

### Branch Description

This field contains an optional description of this task. This can be used by the builder of the Workflow to document what this branch is used for and why it is here. The end-users won't see this field.

### Branch Conditions

A branch in the Workflow can have conditions assigned to it. This provides the flexibility to conditionally route a Workflow based on information from the Workflow process (Form, references, user, etc.).

#### % of participants that choose this branch

You can specify the percentage of participants you want to have clicking on this button before continuing on in the process.

#### # of participants that choose this branch

You can specify the number of participants you want to have clicking on this button before continuing on in the process.

#### Take this branch when

You can build a condition to be met in order for the Workflow process to continue on this branch. Refer to the [Condition Builder topic](#) of this guide.

## Default Branch

You can specify the default branch to take if no condition or multiple conditions are met.

## Branch Options

Each branch has its own options. You can configure them by clicking on the Options tab.

The image displays a workflow diagram and a configuration window. The workflow starts with a green checkmark icon, followed by an 'Orientation' task. From 'Orientation', two branches emerge: a red dotted line leading to 'HR Review' and a green dotted line leading to 'Manager Review'. Both 'HR Review' and 'Manager Review' lead to an 'Office Service...' task. The 'Branch Settings' dialog box is open, showing the 'Review Complete' branch name. The 'Options' tab is selected, showing settings for 'Assign users from previous step' (set to 'Never'), 'Color' (set to 'Default'), and several checkboxes for 'Button Order on eForm', 'Button Icon on eForm', 'Signature Comments Required', 'Allow this branch to be completed from the users Task List', and 'Skip Validation on Form when Selected'.

### Assign users from previous step

This will give you the following options to assign users from the previous step:

- Never
- Always
- Users that have completed their task
- Users that selected this branch (button)

### Color

The “Color” option sets the color of the branch graphic displayed in the Workflow. The “Button Order on Form” option tells the Form in what order to draw the buttons, from left to right. The “Button Icon on

Form” option changes the icon displayed on the Form button.

### Button order on Form

Specify the button order by number that controls the order of the buttons displayed on the form. A button order of -1 will prevent the button from being displayed on the form. This can be used when a condition is associated with the branch (e.g. based on the value of a form field) but you don't want the button to be displayed to the user on the form.

### Button Icon on Form

You can insert an icon on your button which will display before the button label.

### Signature Comments Required

Check this box to require the user to enter comments upon completion of task.

### Allow this branch to be completed from the users Task List

Check this box to allow users to complete the task from their [Task List](#) screen, without having to open the Form associated with the Workflow. When the user completes a task from the [Task List](#), Process Director will display a small popup that indicates the task completion is in progress. Additionally, if the branch selected is configured to require completion comments, Process Director will prompt the user for the comments before completing that task.

The size of the prompt and confirmation dialogs won't usually need to be changed, however, if, for some reason the size does need to be changed, there are custom variables available to change the width and height of the dialogs: `nTaskCompleteDialogWidth`, `nTaskCompleteDialogHeight`, `nTaskCompletePromptDialogWidth`, and `nTaskCompletePromptDialogHeight`. Documentation for these four custom variables can be found in the [Miscellaneous Custom Variables](#) topic of the Developer's Guide.

### Skip Validation on Form when Selected

Check this box to specify that a given branch will cause validation to be skipped. The feature is especially useful for steps/activities that include an option for the user to abandon or reject a form, which the user should be able to accomplish without, for example, filling in all required fields.

### Optional list of strings for email complete

If this branch can be completed by email, users can complete it by replying to the task assignment email with a specific string. This text box is used to set the list of response strings the user can enter into the reply email. For example, if this is a rejection branch, you might enter the list of allowable responses as "Reject, Disapprove, Disapproved, No, Rejection". If the user enters any of these words into the reply email, the task will be rejected.

### Built-in Task Settings

These settings are included in all task types. To configure a step in the Workflow, right click on the step and choose the Properties menu item, or double click on the icon. All steps in a Workflow definition will have the following information associated with them.

The screenshot shows a dialog box titled "Step Settings: eForm Actions". It contains the following fields and options:

- Step Name:** A text input field containing "eForm Actions".
- Step Description:** An empty text input field.
- Options:** Two tabs are visible: "eForm Options" (selected) and "Advanced Options".
- Color:** A dropdown menu with "Default" selected.

## Step Name

The first field on this dialog identifies the name of the step. This information is important for the users and the Workflow owners because it is used to determine what step the Workflow is currently running on, and what function is being performed. This is also displayed on the [Routing Slip](#) to show where the process is currently running.

## Description

This field contains an optional description of this task. This can be used by the Workflow builder to document what this step is used for and why it is here. The end-users won't see this field.

## Instructions

This contains instructions for the users telling them how to perform the task assigned to them. This information will be contained in the email sent to the users, it will be displayed on their [Task List](#), and it is displayed in the Process Timeline Package.

## Color

This field contains an optional color of this step. This can be used by the Workflow builder to document help visualize the Workflow process.

## Workflow Step Task Types



The Workflow object is the legacy process model used in early versions of Process Director. BP Logix recommends the use of the [Process Timeline](#) object, and not the Workflow object. The Workflow object remains in the product for backwards compatibility, but doesn't receive any new functionality updates, other than required bug fixes. No new features have been added to this object since Process Director v4.5. All new process-based functionality is solely added to the Process Timeline.

Users are assigned a task when they are added to a Workflow Step. A context sensitive web page or Form is displayed that is specific to the task type when the user selects the entry in their [Task List](#), or the link in their email notification. Each task type defines a set of functions or actions that can be performed, as well as custom instructions for the participants.

**i** If your process includes tasks for unauthenticated users, please refer to the documentation on [Anonymous Users](#) for special concerns that may apply to some settings.

The Workflow engine embedded within Process Director supports the following tasks:

### User Task #

The user task is any actionable item assigned to a user. If a user task has been assigned to a user, a Task will appear in their [Task List](#). They must complete the task before it can be removed from the [Task List](#). A user who is assigned a task will be automatically given permission to any object in the Workflow package so they may complete the task. Examples of this task can be an approval task or update task.

When a Workflow Step is assigned to a user with an invalid UID or user ID, Process Director will immediately stop the process, and place the Workflow Step into an error state. This is also true for anonymous user assignments if the email address isn't a valid format (Process Director can't validate the email address itself, only the format).

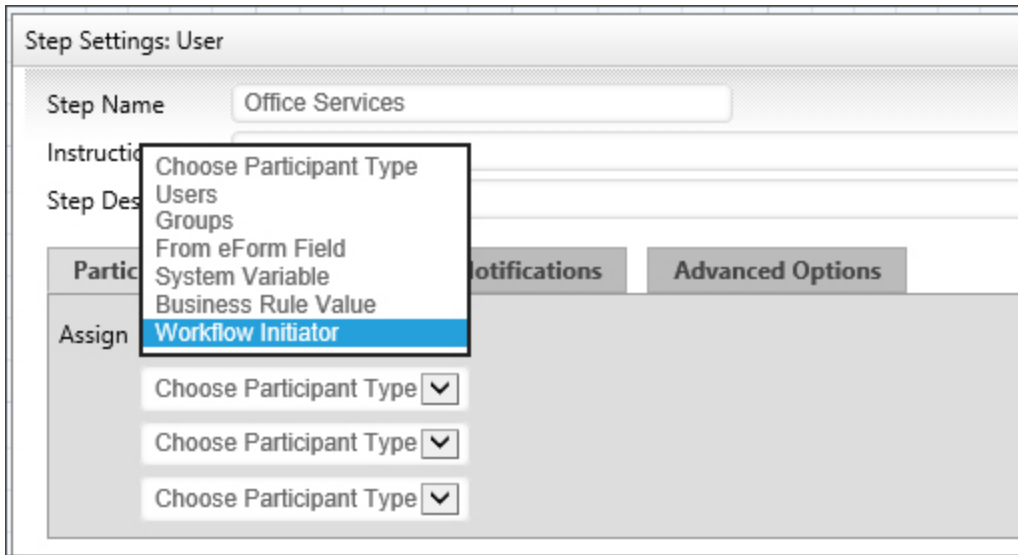
### Participants Tab

A user step in the Workflow can have participants assigned to it. Some Workflow Steps can run without any users (e.g. Form Actions, Parallel Tasks, Sub Workflow, etc.).

The screenshot displays the 'Step Settings: User' configuration interface. At the top, there are three text input fields: 'Step Name' (containing 'Office Services'), 'Instructions', and 'Step Description'. Below these fields is a tabbed interface with four tabs: 'Participants', 'Due Date', 'Notifications', and 'Advanced Options'. The 'Participants' tab is selected and highlighted. Under the 'Participants' tab, there is an 'Assign' section with a dropdown menu currently set to 'Workflow Initiator'. Below this are three additional dropdown menus, each labeled 'Choose Participant Type'.

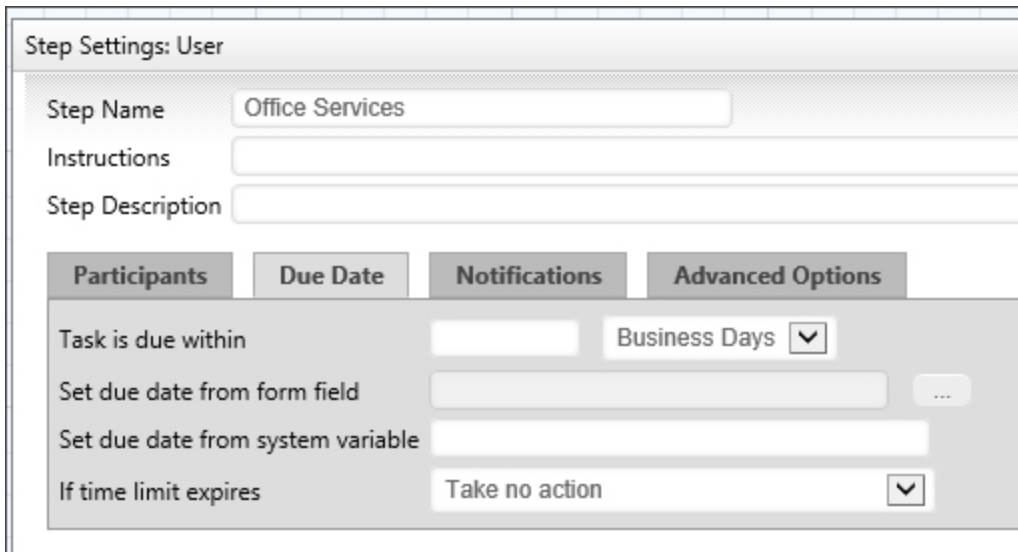
### Assign

A user step can assign participants by user, group, From Form Field, Business Rule Value, and the Workflow Initiator. Any number of users can be assigned to a task.



## Due Date

Each step in the Workflow can have unique time limit/due date options. Process Director runs as a virtual directory in IIS and processes all time related events like due dates when Activity is present on the system (e.g. a login). To force this Activity checking more often, you can schedule a command in the bputil.exe utility.



## Task is Due Within

This specifies a due date for this step. The step due date is optional. If a step doesn't complete by the due date specified it can automatically transition to the next step in the Workflow. The due date can also be used to prioritize items in a user's [Task List](#).

### Set Due Date to Form Field

This will allow the user to specify a due date on the form using a Date Picker for that step in the Workflow process

### If Time Limit Expires

If the time limit (due date) passes before this step is complete these are the actions that can be taken.

OPTION	DESCRIPTION
Take no action	This lets the step continue running.
Cancel the Step	This will cancel the step and automatically advance to the next step in the Workflow.
Cancel the entire Workflow	This will immediately cancel the Workflow.
Add These Users from Business Rule	This will add users identified in the Business Rule.
Replace with These Users from Business Rule	This will add the user from the Business Rule and cancel any running users.
Jump to Workflow Step	This will cancel the step and automatically jump to the step Name that is specified

Step Name: Office Services  
Instructions:   
Step Description:

**Participants** | **Due Date** | **Notifications** | **Advanced Options**

Task is due within:  Business Days   
Set due date from form field:  ...  
Set due date from system variable:   
If time limit expires: 

- Take no action
- Cancel the Step
- Cancel the entire Workflow
- Add these users from Business Rule
- Replace existing users with those in Rule
- Jump to Workflow Step

## Notifications Tab

Each step in the Workflow can have unique notification options.

Step Settings: User

Step Name: Office Services  
Instructions:   
Step Description:

**Participants** | **Due Date** | **Notifications** | **Advanced Options**

Notify participants when step starts (Task Assigned)   
Email Notifications: 

- Choose Notification Time
- Choose Notification Time
- Choose Notification Time

  
Using email template:  ...

## Notify Participants When Step Starts

This will allow the Workflow to notify all users that have been assigned from the Participants tab.

## Notify

This will allow you to select a user or users from various places in the Workflow and form to notify in that current step. Once you have made your selection you'll have to select from the Choose Notification Time. You have up to three selections to notify.

You can tell the step how to determine what users to notify with the following options:



### ***Notification Time***

- When step starts.
- When step completes.
- When users complete: You can choose specific users or groups whom you wish to complete the task prior to notification.
- When step is past due date.

You may also choose various reminder times, to send notifications a varying number of days before or after a task is due. Sending these reminders are useful for creating escalation notifications, or notifying managers that tasks are slipping past due dates.

### ***Notify Type***

- All Active Users in Step: this will send a notification to all users who have yet to complete a user step
- All Users in Step: this will notify all the users involved in this User step.
- Users: this allows you to select specific users to notify.
- Groups: this will notify all users in a specific group.
- Business Rule Value: this will notify the user result of a Business Rule.
- Workflow Initiator: this will notify the initiator of this Workflow instance.
- From Form Field: this will notify a User specified by a form on the container Form.
- External Users: this will send a notification to a specified email address.

### **Using Email Template**

This is an optional email template to use when sending email notifications to users in this step. If this isn't specified, the default email template will be used that is shipped with Process Director. For more information on email templates refer to the section named Using Email Templates in this guide.

### **Advanced Options Tab**

Depending on the task type for the step different options will be available. Some of these options will appear under the Advanced Options tab, others will appear under a tab that is unique to the task type.

Participants	Due Date	Notifications	Advanced Options
Use Form in this Step			<input type="text"/> ...
Step is complete when			All Users Have Completed ▼
Restart Users			Start configured and users all previously run in this task ▼
Restart All Users When			<a href="#">Click to create condition ...</a>
After a user completes this task			Automatically show user's next task if it is in this process ▼
Re-authenticate users		<input type="checkbox"/>	
Prompt for Signature Comments		<input type="checkbox"/>	
Allow task to be completed from email		<input type="checkbox"/>	
Allow Task Sharing		<input type="checkbox"/>	
Assign users to this task			All in parallel ▼
Enable email invitations		<input type="checkbox"/>	
Require at least one user to be assigned and complete this task		<input type="checkbox"/>	
Assign task to first user to accept		<input type="checkbox"/>	
Allow anonymous users to be assigned tasks by email address		<input type="checkbox"/>	
Switch the user in Case Folder View when opening this task		<input type="checkbox"/>	
Color			Default ▼

### Use Form in this Step

This will allow you to specify a Form to use in this step. This controls the form displayed when a user opens their [Task List](#).

### Step is Complete When

This will allow you to select when the step is complete. You can choose when “All Users Have Completed”, “First User Completes Task”, or “When any branch condition is met”.

### Restart Users

This option determines how participants are assigned to the step when it must be restarted. This is particularly relevant when an implementer changes the participants assigned to an Activity in the Workflow definition. An existing instance may have been created when different participants were configured to participate in the task. This option enables you to determine how to handle user assignments when the configured users have changed.

It is also important to note that the term "restart" is used for both:

- Cases in which a running task is restarted while the task is still running, and
- Cases in which a task has already been completed, but must be re-run, as in an iterative process, or as a result of a change to some condition.

In either case, the [Restart Users](#) options you select will be applied.

The following options are available:

OPTION	DESCRIPTION
Start only configured task participants	This option will assign the task to the currently configured users, irrespective of which users may have been previously assigned to the Activity.

OPTION	DESCRIPTION
Start configured and users all previously run in this task.	By choosing this option, you ensure that both the currently configured users, as well as any users who were previously assigned the task, are assigned the task.
Start only users that previously completed this task	This option will assign the task only to users who completed the task when it was previously run for this instance. Those users will be re-assigned the task irrespective of whoever is currently configured for task assignment in the Workflow definition.
Start from last completed user	If multiple users are assigned to the Activity instance, this option will re-assign the Activity to the last user who completed it when it was initially run.
Start users that did not complete previously	If multiple users are assigned to the Activity instance, this option will assign the task to users who did not previously complete it. For instance, if the Activity uses a round-robin assignment, the Activity will be assigned to one of the users who were not assigned it on the initial run.

### Restart all users when

This option enabled you to define a condition, or set of conditions under which all users in this task will be reassigned.

### After a user completes this task

This option determines how a user's tasks are displayed when a user is assigned two sequential tasks in a process. The default in Process Director is to automatically show the user's next task if it is in the current process.

For example, if a user completes a task in a Form, and is also assigned the next task in the sequence, once the user clicks the OK or other task completion button on the Form, the Form won't close, but will simply redisplay itself to allow the user to complete the next task in the sequence. For various reasons, you may wish to override this behavior, in which case the Form will close once the first task has been completed, and the user will have to re-open the Form to complete the next task in the sequence.

You have the following options for configuring the behavior of Process Director:

OPTION	DESCRIPTION
Automatically show user's next task if it is in this process	This is the default behavior, and will automatically redisplay the Form once the user completes the first task in the sequence.
Automatically show user's next task if it is for this step in a different process	This selection will automatically show the user's next task if the user is assigned a task in another process. For instance, if the current task starts a subprocess, and the user is assigned a task in the subprocess, the subprocess task will automatically be displayed.

OPTION	DESCRIPTION
Do not show the user's next task	Once a user completes the task, the Form will close, and the user must reopen the Form from the <a href="#">Task List</a> to begin performing the next task.

### Re-authenticate users

This indicates that the participants must re-enter their User ID and password when attempting to complete their task. This is provided for regulatory compliance when an electronic signature is required for non-repudiation.

### Prompt for Signature Comments

This indicates that the participants will receive a prompt to provide signature comments when the task is completed. The prompt will appear as a dialog box when the user attempts to complete the task in the Form without providing signature comments.

### Allow task to be completed from email

Checking this box will allow users to complete a task by responding to the email notification sent by the user step.

### Allow Task Sharing

Checking this option will enable task sharing through the [Shared Delegation](#) feature. If this option isn't checked, this task won't be available for Shared Delegation.

### Assign users to this task

This option enables you to select how wish to assign a task when multiple users are associated with the same task. You have the following options:

OPTION	DESCRIPTION
All in parallel	All users will be assigned the task at the same time, and will complete their actions at the same time as the other users.
All in series	Each user will be assigned the task individually, and each user must complete the assigned action before the task is assigned to the next user. The process will continue until all users have completed their actions.
One (Round Robin)	Only one user will be assigned the task each time an instance of the Timeline is run. Each time an instance is run, a different user will be assigned the task until all users have been assigned an instance of the process. This will divide the assignments equally among all assigned users.
One (fewest tasks in this process)	Only one user will be assigned the task each time an instance of the Timeline is run. Process Director will assign the task to the user who

OPTION	DESCRIPTION
	has the fewest number of active tasks pending in this process. This method assigns the task to the user who has the smallest workload in this process.
One (fewest tasks overall)	Only one user will be assigned the task each time an instance of the Timeline is run. Process Director will assign the task to the user who has the fewest number of active tasks in their <a href="#">Task List</a> . This method assigns the task to the user who has the smallest overall workload.

### Enable email invitations

By checking this box, a user will be able to invite another user to a task by forwarding the notification email to him. The receiver of the forwarded email will then be able to follow a link and accept or decline the invitation.

### Require at least one user to be assigned


Depending on how the task assignment is configured, it's possible that no users will be assigned to the task. For instance, if a task restarts after all users have completed their assignment, and the assignment is set to "Start users that did not complete previously" there will be no available user to which the task can be re-assigned. Checking this option will assign the task to one of the previously completed users.

### Assign task to first user to accept

This satisfies a scenario in which multiple users are configured for the task, but only one user is required to complete the task. This option will notify all of the step users that a task is pending, and then assign the task to the first user who accepts it.

### Allow anonymous users to be assigned tasks by email address

Checking this option will enable anonymous users to complete a task based on the user's email address. This is a required option when using the [Email Anonymous Task List](#) system variable. In the vast majority of use cases, however, the email address will be taken from a Form field identified on the Participants tab.

 Access to Process Director for unauthenticated or anonymous users is enabled by an optionally licensed component.

### Switch the user in Case Folder View when opening this task

If this task is part of a Case Management application, checking the property will automatically display the Case Folder view when the user opens the task.

### Color

This field contains an optional color of this step. This can be used by the Workflow builder to document help visualize the Workflow process.

## Decision Task #

The Decision Step is used to take a single branch coming out of this step. Assign conditions on the actual branches (use right-click Properties on the actual branches). Only a single branch can be taken. You can designate a single branch to be the default if (Default Branch property of the branch) the conditions of multiple branches could evaluate to true.

The screenshot shows the 'Step Settings: Decision' dialog box. It has a title bar with a close button. The main area contains the following fields and options:

- Step Name:** A text box containing 'Decision'.
- Step:** An empty text box.
- Description:** An empty text box.
- Advanced Options:** A tabbed section containing:
  - A text box with the following text: "The Decision Step is used to choose one branch of the one or more branches leading from this step. Assign conditions on the actual branches (use right-click Properties on the actual branches). Only a single branch can be taken. You can designate a single branch to be the default if (Default Branch property of the branch) the conditions of multiple branches could evaluate to true."
  - Color:** A dropdown menu currently set to 'Default'.

At the bottom of the dialog, there are left and right navigation arrows.

## Form Actions #

The Form Options task allows the Workflow package to control which Form will be the default when multiple forms are contained within the Workflow and allows new forms to be attached.

The screenshot shows the 'Step Settings: eForm Actions' dialog box. It has a title bar with a close button. The main area contains the following fields and options:

- Step Name:** A text box containing 'eForm Actions'.
- Step:** An empty text box.
- Description:** An empty text box.
- eForm Options:** A tabbed section containing:
  - Form Definition:** A text box with a browse button (...).
  - Set this as the default eForm Instance for the Workflow:** A checkbox that is currently unchecked.
  - Add new eForm Instance:** A dropdown menu set to 'Only if the eForm does not already exist in the Package'.
  - Group Name:** An empty text box.

At the bottom of the dialog, there are left and right navigation arrows.

## Parallel Tasks #

This allows you to run your steps in parallel. The Parallel Task Step will start all branches coming out of this step at once. You can optionally add conditions to any branches. The Wait step can be used to wait for all parallel branches to complete.

The screenshot shows the 'Step Settings: Parallel Tasks' dialog box. It has a title bar with a close button. The main area contains the following fields and options:

- Step Name:** A text box containing 'Parallel Tasks'.
- Step:** An empty text box.
- Description:** An empty text box.
- Advanced Options:** A section with a grey background containing:
  - A text box with the text: "The Parallel Task Step will start all branches coming out of this step at once. You can optionally add conditions to any branches (use right-click Properties on the actual branches). The Wait step can be used to wait for all parallel branches to complete."
  - Color:** A dropdown menu currently set to 'Default'.

At the bottom of the dialog, there are left and right navigation arrows.

## Process #

The screenshot shows the 'Step Settings: Process' dialog box. It has a title bar with a close button. The main area contains the following fields and options:

- Step Name:** A text box containing 'Process'.
- Step:** An empty text box.
- Description:** An empty text box.
- Advanced Options:** A section with a grey background containing:
  - Start this Process:** A text box with a dropdown arrow on the right.
  - Copy objects in Workflow Package to Parent:** A checked checkbox with a text box below it labeled 'Only from this group'.
  - Copy objects in Workflow to Child Process:** A checked checkbox with a text box below it labeled 'Only from this group'.
  - Color:** A dropdown menu currently set to 'Default'.

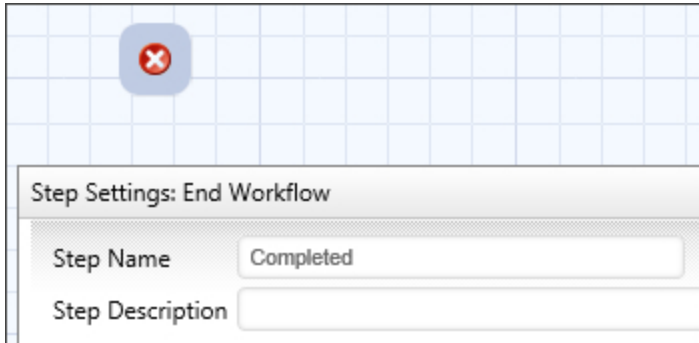
At the bottom of the dialog, there are left and right navigation arrows.

## Subprocesses and End Process Steps

The Process step requires that a Workflow definition be configured. The configured process will be started as a subprocess (child process) of the main (parent) Workflow. A subprocess can consist of a Workflow or a Process Timeline. You may copy objects from the parent process to the child process and vice versa.

When a subprocess contains an End Process step, it will return the name of the End Process step as a result for the parent Process Activity. Process Timelines and Workflows use the End Process function differently, so the results can be slightly different depending on the type of subprocess that is called by the Process step.

### *Workflow*



In the example above, the Workflow End Process step is named completed. If this Workflow runs as a subprocess, then, when the process reaches this step and completes, the name "Complete" will be passed to the parent Workflow as the result of the Process step that started the sub-process.

Workflows do NOT automatically end when an End process step is reached. Workflows end when all of the required steps in a Workflow have completed. This enables you to have two Workflow paths running concurrently, or in parallel, each of which may have its own End Process step. If a Workflow contains two parallel paths that each have a different End Process step, then, when the subprocess completes, it will return a comma-separated string containing the names of BOTH end process steps. In the parent Workflow, the returned string will then be set as the result of the Process step that started the subprocess.

### *Timeline*

For information on how Process Timelines work in this configuration, please refer to [the Results Tab section](#) of the Process Timeline Activities topic.

## **Comment #**

This task type is only for documenting the Workflow. It allows text to be entered that can be used to describe the Workflow process.



## **Script #**

The Script task supports the ability to call custom script functions. For more information on the custom scripts refer to the Scripting section of this guide.



Step Settings: Script

Step Name

Step Description

**Script Options**    **Advanced Options**

Script File  ...

Script Parameters (optional)

Use eForm in this Step  ...

## Script Options Tab

### Script File

You may use the [Object Picker](#) to select a script that will automatically start when the Activity starts.

### Script Parameters (optional)

You may enter a parameter string to pass to the selected script.

### Use Form In this Step

This property enables a Script Step in a Workflow to specify a different Form to use a current form for the Activity. If this property isn't configured, Process Director will revert to passing the script to the current form instance that is associated with the running process.

## Meta Data #

The purpose of this task is to assign Meta Data to the Workflow objects. The Meta Data is extracted from the default form instance, and can be applied to all Workflow objects, or to objects in a specific group.

Step Settings: Meta Data

Step Name

Step

Description

**Advanced Options**

Push the meta data from the default form instance to the other objects in this workflow package

Only to objects in this group (optional)

Color  ▼

## Wait #

This task will wait for certain conditions before completing (e.g. Parallel Tasks to complete, Sub-Workflows to complete, etc.). This can also be used for Workflow synchronization to synchronize events between different processes.

Step Settings: Wait

Step Name: Wait

Step Description:

Wait Options | Notifications | Advanced Options

Wait for event string to be posted: [input field]

Wait for Parallel Steps to Complete:

Wait for Sub Processes to Complete:

Only this Process: [input field] ...

Stop Waiting When Any Branch Condition Is Met:

Maximum amount of time to wait: [input field] Business Days [dropdown]

Set maximum date to wait from form field: [input field] ...

Set maximum date to wait from system variable: [input field]

For the wait step to be satisfied these conditions must be true:

- If Wait for Parallel Steps to Complete is checked, all parallel tasks complete.
- If Wait for Sub Processes to Complete is checked, all child sub-processes (or a specific sub-process) must be complete.
- If Wait for event string to be posted is checked it will wait until that event is posted (from an external source or process).

Or

The Stop Waiting When Any Branch Condition Is Met is checked, and the condition on any branch leaving the wait task is satisfied (note: if no conditions are specified on the branch, this will cause the wait task to complete immediately).

Or

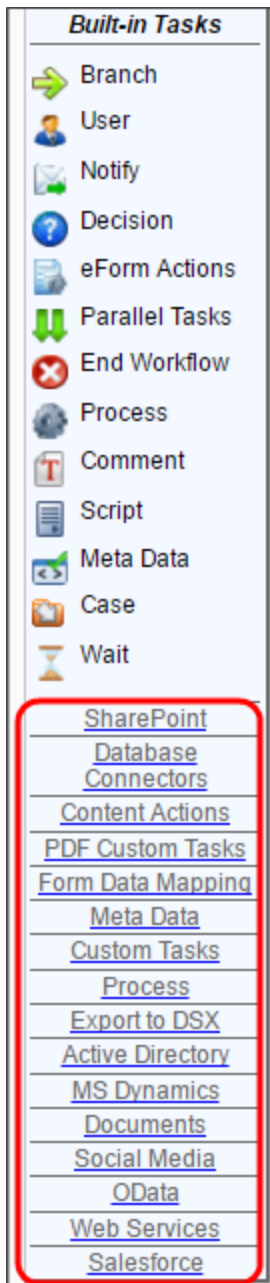
If the Set maximum date to wait from form field occurs before the wait event is satisfied the step will be canceled and it will automatically advance to the next step in the Workflow.

Or

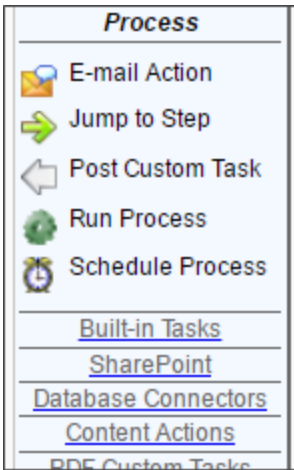
If the time the step has waited exceeded the result of the value specified in the Set maximum date to wait from system variable field, the step will be canceled and the Workflow will automatically advance to the next step.

## Custom Task #

Process Director Custom Tasks can be used as a Workflow Step. The different types of Custom Tasks appear on the left side of the Workflow definition screen, below the Built-In Tasks.



Clicking on a Custom Task Type will display the Custom Tasks of that type at the top of the sidebar. In the example below, the sidebar displays the Process Custom Tasks.



Simply drag the Custom Task onto the flowchart, just as you'd any Built-In Task. Open the properties of the Custom Task, and use the **Custom Task Options** tab to configure the task. On the **Custom Task Options** tab, click the **Configure** button to open the configuration screen for the task. Each Custom Task will have a specific configuration screen. For details on the configuration settings for a Custom Task, please refer to the [Custom Tasks Reference Guide](#).

## Workflow Package



The Workflow object is the legacy process model used in early versions of Process Director. BP Logix recommends the use of the [Process Timeline](#) object, and not the Workflow object. The Workflow object remains in the product for backwards compatibility, but doesn't receive any new functionality updates, other than required bug fixes. No new features have been added to this object since Process Director v4.5. All new process-based functionality is solely added to the Process Timeline.

When a Workflow process is started, a Workflow Package is created. This is what is routed to the participants of the Workflow. The Workflow Package contains the **Routing Slip**, the Workflow objects (documents, Forms, etc.), optional references, and administrative controls. The Workflow Package can be viewed by authorized users for running and completed Workflows.

### Routing Slip #

Authorized users have access to the **Routing Slip** in the Workflow Package by clicking on the Workflow Package Tab. This page displays a Routing Slip that shows the progress of the Workflow process. It displays all users that are participating in this Workflow and how far it has progressed in the routing process, including all iterations of a step.

^ Routing Slip						
Participants	Signature	Completed	Status	Result	Comments	
▼ Initiator						
Ron Harris	<i>Ron Harris</i>	6/2/2014	Completed			
▼ (Stage 2) Country Site Review Team 6/2/2014 9:48 AM						
Ron Harris	<i>Ron Harris</i>	6/2/2014	Completed	✓ Country Site Review Complete	done	
▼ MHRP Executive Review 6/2/2014 9:49 AM						
David Henry	<i>David Henry</i>	6/2/2014	Completed	✓ Complete	done	
▼ HQ Contracts Executive Review 6/2/2014 9:53 AM						
Bob Barker	<i>Bob Barker</i>	6/2/2014	Completed	Complete	all done	
▼ (Stage 4) HQ Contracts Review 6/2/2014 9:53 AM						
Barb Stanley	<i>Barb Stanley</i>	6/2/2014	Completed	✓ Generate Draft Contract		
▼ (Stage 5) HQ Revise Draft 6/2/2014 9:55 AM						
Barb Stanley	<i>Barb Stanley</i>	6/2/2014	Completed	✓ Contract is Complete		
▼ Agreement is Ready 6/2/2014 9:56 AM						
Ron Harris		6/2/2014	Notified			

This [Routing Slip](#) is very similar to the [Routing Slip](#) control that is displayed on Forms.

### Signature Comments #

Administrators can choose to require and display comments on the [Routing Slip](#). Checking the “Prompt for Signature Comments” checkbox in the Advanced Options tab of a User control will automatically put a Signature Comments control at the bottom of the Form for users completing that task.

The screenshot shows the 'Step Settings: User' dialog box with the 'Advanced Options' tab selected. The 'Step Name' is 'Office Services'. The 'Instructions' and 'Step Description' fields are empty. The 'Advanced Options' tab contains the following settings:

- Use eForm in this Step: [Empty field]
- Step is complete when: All Users Have Completed
- Restart Users: Start configured and users all previously run in this task
- Restart All Users When: [Click to create condition...](#)
- After a user completes this task: Automatically show user's next task if it is in this process
- Re-authenticate users:
- Prompt for Signature Comments:**  (highlighted with a red rectangle)
- Allow task to be completed from email:
- Assign users to this task: All in parallel
- Enable email invitations:
- Require at least one user to be assigned:
- Assign task to first user to accept:
- Color: Default

Simply checking this checkbox won't require that the user enter comments, it will only prompt them to. To require that the user enter comments, check the "Require Signature Comments" checkbox in the Options tab of Branch settings. This way, you can configure it such that users only need enter comments when one branch is chosen and not another.

**Branch Settings**

Branch Name: Review Complete

Branch Description:

**Conditions** | **Options**

Assign users from previous step: Never

Color: Default

Button Order on eForm:

Button Icon on eForm:

**Signature Comments Required**

Allow this branch to be completed from the users Task List

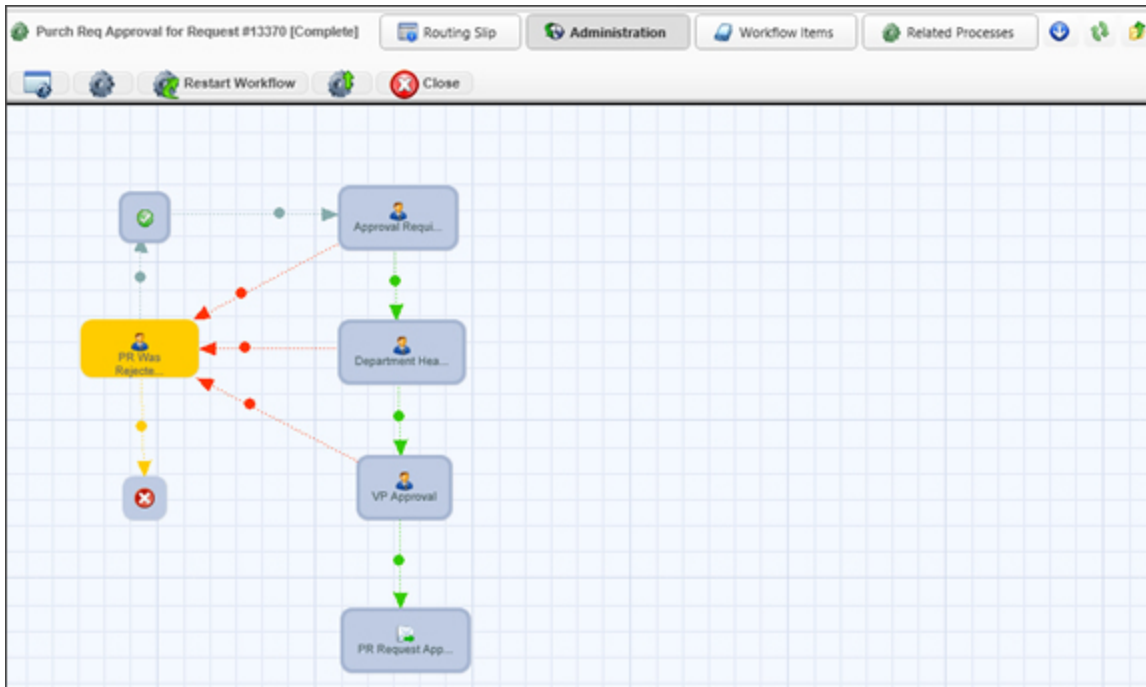
Skip Validation on Form when Selected

Optional list of strings for email complete:

You can also prompt users to enter signature comments in a Process Timeline task. See the Process Timeline section for instructions.

### Graphical Administration #

Only authorized users will see this button displayed. The user must have Modify permission for the running Workflow or have Modify Children permission for the Workflow definition. The Administration button in the Workflow Package allows users to modify the running Workflow. From this screen, authorized users can change the running Workflow by adding users to a step, removing them from a step, or jumping to a new step.



## Administering Tasks

From the Graphical Administration screen, you can double-click on a task to open the administration screen for that task.

Task Type: User						
Step Status	Termination Reason	Result	# Times Step Run	Step Started	Step Ended	Step Due
Active	Not Set		1	10/27/2014	-	10/30/2014

Actions: [Cancel Step](#), [Restart Step](#), [Cancel Workflow](#), [Add User\(s\)](#), [Add Me To This Task](#)

You have a number of options to choose when administering a task.

Option	Result
Cancel Step	Cancels this step in the Workflow.
Restart Step	Restarts this step in the Workflow. User, Wait, Custom Task, or Script steps may all be restarted.
Cancel Workflow	Cancels the entire running Workflow instance.
Add User(s)	Adds additional users to the Workflow Step.
Add Me to This Task	Adds you to the Workflow Step as a participant.
Cancel User	Cancels the user's participation in the task.
Complete User Task	Sets the user's task as complete.



Option	Result
Remove User	Remove the user from the task.
Reassign User	Assign a different user to the task.

## Workflow Items #

The Workflow Items page is what is presented to users that are participating in a Workflow Step. This page displays all of the objects being routed in this Workflow (e.g. documents, Forms, etc.). A Workflow Package can contain multiple objects of any type. If the Form has an attach button, users in that step will be able to add objects to the Workflow Package. Each new object added is added to the Workflow Items page. If a user has Delete permission to the running Workflow, they'll be able to remove objects from the Workflow package.

Name	Added By	Last Updated	Group Name	Actions
Small Doc.doc	Ron Harris	8/25/2014	Justification	
Purchase Requisition #13370 Submitted On 8/25/2014	Ron Harris	8/25/2014		

## Related Workflows #

This page in the Workflow Package will only appear if the Workflow process is related to any other Workflow processes. If this Workflow is a parent or child process of another Workflow then the Related Workflows button is automatically displayed. This displays the relationship between the Workflows allowing users to click on the different Workflow entries and view the status (i.e. Workflow Package) for each of the other Workflows.



## Workflows in the Content List #

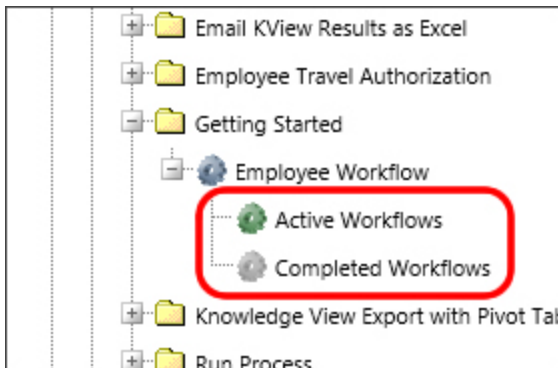
Workflow definitions are stored in the [Content List](#) just like any other object in the database. When a Workflow is started, an entry will be created under the Workflow definition in the Content List as a child object. The entry will remain there even after the Workflow has completed. To view the running and completed Workflows for a Workflow definition, users can click on the “Active Workflows” or Complete Workflows” icons next to the Workflow definition name in the [Content List](#).

Name	Author	Create Date	Update Date	Size	Actions
Employee Workflow	Dale Franks	9/19/2014	9/19/2014	6 steps	
Getting Started eForm	Dale Franks	3/21/2014	9/11/2014	22 fields	
Getting Started Timeline	Dale Franks	3/21/2014	9/12/2014	5 activities	
GS-ConditionalApprovalSample	Dale Franks	3/21/2014	3/21/2014	0 values	
Job and User Information	Dale Franks	9/18/2014	9/18/2014	-	
Sample Dropdown	Dale Franks	9/18/2014	9/18/2014	0 entries	
test	Dale Franks	9/2/2014	9/2/2014	0 values	

## Viewing Running and Completed Workflows

If users have View Children permission to the Workflow definition, they'll automatically be given View permission to any running and completed Workflows for that definition. The running and completed

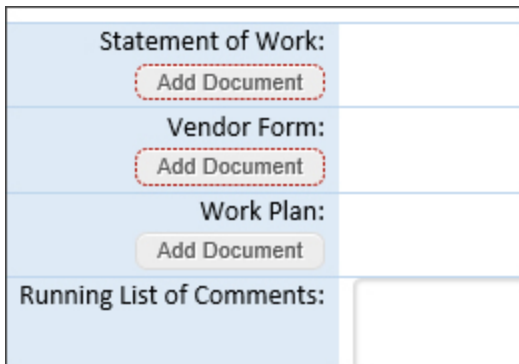
Workflows are displayed in the [Content List](#) swing status, what step is running and the user that initiated the Workflow. Users can view the Workflow Package for any of the running or completed Workflows by clicking on the name or the icon (  ) in the Content List. To view the objects that are part of a Workflow Package, click on the icon (  ) for the appropriate Workflow name.



Running and completed Workflows can also be made available to users through a Knowledge View. For example, if users have a need to see only certain types of running Workflows, a Knowledge View filter can show a list of the Workflows making the actual location of them in the [Content List](#) transparent.

### Attaching New Workflow Objects #

A running Workflow can allow users to add new objects (e.g. documents, Forms). If a user has an attach button on a Form, they'll be able to add new objects to the running Workflow.



When a new document is added under the Workflow Package it will only be contained under the Workflow. The document won't exist anywhere else in the [Content List](#).

When a reference (e.g. shortcut) to an existing object in the database is added, only a reference to the object will be created in the Workflow. A reference, or shortcut, is a symbolic link to the original object. Any changes made to the reference will also update the object it is pointing to, except in the case of a Form. If a reference to a Form is added to the running Workflow, an instance of the Form will be created and placed under the Workflow Items button.

### Automatic Workflows for Folders #

The Process Director Workflow engine supports the ability to automatically start Workflows for certain document events within a folder. Automatic Workflow definitions can be specified for any folder. A

Workflow definition can be associated with one of the following document actions:

- When a new document is added to a folder;
- When a document in a folder is updated and checked in;
- When a document is moved into a folder.

If one of these conditions is met, the appropriate Workflow will automatically be started against that document.

To set Workflows to start automatically, click on the “properties” icon for the folder to open the folder's properties screen.

Use the [Object Picker](#) to select a Workflow from the [Content List](#) that you want to start under the given conditions.

## Running Workflows <#>

A Workflow can be started by selecting one or more objects in the [Content List](#) and choosing the Start Workflow link. This will display a popup with a list of Workflow definitions the user has permission to start. Changes to the Workflow definition will affect all running Workflows.

## Task Lists

[Task Lists](#) are fundamental to Workflow management systems. The Process Director Workflow engine supports an integrated [Task List](#) that provides each user with a list of their assigned tasks, according to priority and due date. As a user completes an assigned task, the item is automatically removed from their [Task List](#).

Tasks List Knowledge Views with a “Complete Task” column allow the user to complete the task from the [Task List](#). A column will appear in the Task List with icons representing every option the user can choose. Clicking the icon will complete the user’s task.

### Permissions and Running Workflows

A running Workflow will temporarily override the permissions for any object (document, Form, etc.) contained within it, giving the user the necessary access to perform the requested task. When users are assigned to a Workflow task, they’ll automatically be given Modify permission to the objects until their task is completed. This prevents the object permissions from effecting running Workflows.

### Advanced Form Options in the Workflow Package #

When a Form is attached to a Workflow (via the Form Actions) it creates a new form data instance in the Workflow package. The Workflow package may contain multiple forms attached to it. When using multiple forms in a Workflow package, you can control which form is the default that will be displayed to the user when they click on the entry in their [Task List](#) or their email notification.

In Process Director, the Form is the viewer which controls how information is displayed to the user and the form instance contains the actual form data. This means that you can use different Forms to view the same form data. This is controlled through the Workflow processing, allowing you to present a user with a different Form, but still have it associated with and displaying the same form instance data.

### Choosing the Current Form Instance

When multiple form instances are attached to a Workflow package, you can select the instance that should be used as the current or default form instance for the Workflow. This is performed in the Form Actions Workflow task. If a current form instance isn’t set using this task, the first form instance that was attached to the Workflow will remain the current form instance. In the Form Actions task, you can choose the default form instance by selecting the actual Form Definition that was used to create or view the form data and check the “Set this as the default Form Instance for the Workflow” box. The running Workflow will determine the appropriate form instance to use based on the “Set this as the default Form Instance for the Workflow Form Instance” field below.

The screenshot shows a dialog box titled "Step Settings: eForm Actions". It has a "Step Name" field containing "eForm Actions". Below it are "Step" and "Description" fields. There are two tabs: "eForm Options" and "Advanced Options". The "Advanced Options" tab is active. A red rectangular box highlights the "Form Definition" field, the "Set this as the default eForm Instance for the Workflow" checkbox (which is unchecked), and the "Add new eForm Instance" dropdown menu (which is set to "Only if the eForm does not already exist in the Package"). Below these are "Group Name" and "Add new eForm Instance" fields. The dialog has a close button in the top right and navigation arrows at the bottom.

## Running a Workflow from a URL #

Workflows can be run manually, or scheduled to perform an automatic actions. To schedule these Workflows, a web page (a Web Service web page) is executed to run a specific Workflow definition.

Ensure you have enabled Web Services on the Installation Settings page. You can optionally enable security restrictions and authentication. If you pass credentials on the URL, you'll need to set [fWebServiceAllowCredentialsURL](#) to true in the custom vars file.

### Manual Workflow

Execute the following URL from a browser on the server that has Process Director:

```
http://localhost/services/wsWorkflow.asmx/Run?wfid=
WFID&bpUserid=USERID&bpPassword=PASSWORD
```



Be advised that using the bpUSERID or bpPassword parameter requires sending the User ID and Password in clear text, so be mindful of the security implications of transmitting these values.

The wsWorkflow.asmx command can take the following parameters on the QueryString URL:

- wfid: The WFID of the Workflow Definition to run;
- bpUserid: A Process Director user ID with permission to run the Workflow Definition;
- bpPassword: A Process Director password with permission to run the Workflow Definition;

### Scheduled Workflow

You may want to automatically schedule the Workflow to run at regular intervals (for example, every night at midnight). To do this, use the bputil.exe utility. This utility enables you to schedule and test commands executed on a regular basis.

Do not schedule IEXPLORE.EXE because the web browser will never close. Rather, use the bputil.exe command to run the web page. For example, enter this command in the “Run” dialog box to schedule the synchronization:

```
"PATH\bputil.exe"
```

SU

```
http://localhost/services/wsWorkflow.asmx/Run?wfid=
```

```
WFID&bpUserid=USERID&bpPassword=PASSWORD
```



Be advised that using the bpUSERID or bpPassword parameter requires sending the User ID and Password in clear text, so be mindful of the security implications of transmitting these values.

(Where PATH is the installation directory for Process Director, e.g. c:\Program Files\BP Logix\Process Director\). Enter the appropriate credentials in the Windows Scheduler when prompted. Use the “Schedule” tab to set the times to run the command. Consult the Microsoft help for more information on this utility.

## Implementation Guidelines

While knowledge of the various [Process Director objects](#) is vital to creating any application, there are several Best Practices we recommend you follow. Additionally, knowing how to import and export data, how to organize your [Content List](#), and other equally important topics will make your job as an implementer much easier. The topics in this section will provide more in-depth knowledge of how your applications will work, and provide valuable information for both initially creating applications, and for managing them over a longer term.

The topics in the section can be accessed via the Table of Contents that is displayed on the upper right corner of this page, or by clicking one of the links below.

[Best Practices For implementation](#): Following these guidelines will make your implementations work more smoothly, be easier to maintain, and will enable you to make necessary changes more quickly, as personnel or processes change and evolve.

[Managing Content](#): Recommendation on how to organize the [Content List](#), create, import and export objects, use the Process Director import and export utilities, use the Template Library feature, and more.

[Managing Users](#): Information about how users are created, user profiles, and external/anonymous users.

[Permissions](#): Instructions for setting and managing object permissions in Process Director, and recommendations for implementing an effective permissions methodology on your Process Director installation (s).

[Changing Existing Processes](#): An explanation of how Process Director handles changes to existing applications, what to expect when importing a modified application, and how in-flight processes will respond to various changes.

[Accessibility](#): A detailed explanation of how WCAG accessibility guidelines apply to Process Director, important legal considerations about accessibility, and recommendations on how to implement accessibility guidelines into your Process Director applications.

[Data Flow Analyzer](#): A description of, and instructions for using, the built-in [Data Flow Analyzer](#) to see how your application is using data in real-time, as you develop data-driven applications.

[BP Logix Mobile Application](#): Instructions for using the additionally-licensed mobile application component that is available for Process Director.

## Best Practices for Implementations

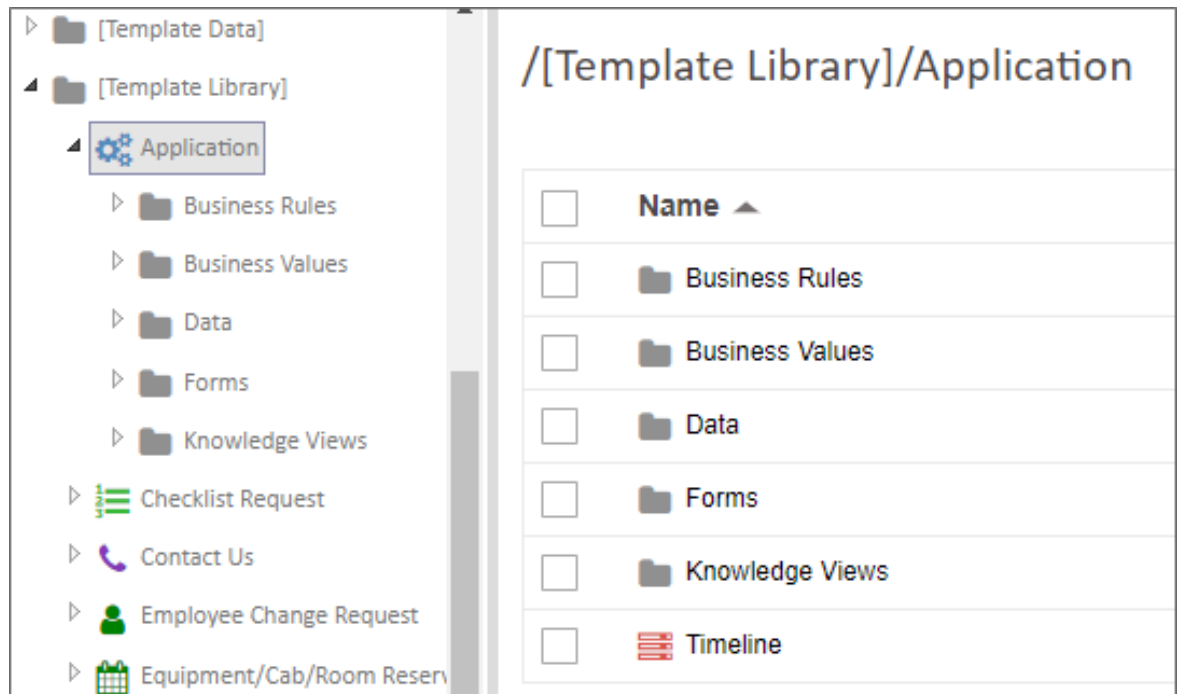
There are a number of Best Practices that you should incorporate into your implementation projects. Following these guidelines will make your implementations work more smoothly, be easier to maintain, and will enable you to make necessary changes more quickly, as personnel or processes change and evolve. In addition to these best practices, you should also familiarize yourself with the basic design concepts that are explained in the [Getting Started Guide](#).

### General

1. When you delete a form instance, ensure that you delete the corresponding Process Timeline instance as well.
2. Always attach objects as Process references instead of Form references, except in those cases where a form-specific attachment is required. Always use a Group Name for attachments. The Group Name is the only reliable handle we have for selecting attached objects.
3. As a general rule, you should create only one Datasource per database back end, and reuse that Datasource for all relevant implementation projects. Only create multiple Datasources when a) you are pointing to two or more databases, or b) You want to specifically restrict a Datasource to specific objects. For example, you might create a Datasource with a smaller subset of tables for use in a specific project, to ensure a class of users don't have access to more sensitive tables that might be included in the main Datasource.
4. Organize Datasources in root folders of the partition, separate from any implementation folders, so that implementers don't have to export Datasources when the implementation is exported from development to production. Exports and Imports are name-based, and relative folder position-based. Consequently, the root folder containing database sources should be mirrored on development and production servers, so that both the names and relative folder positions are exactly the same in both environments. If the mirroring is correct, implementations will be imported/exported between development and production without losing contact with the Datasources.
5. When creating conditions that require a specific value, use the "equals" (=) condition rather than the "contains" condition when possible. Only use 'contains' when a null value is desired as a return proxy for "show me everything" When building conditions testing matching strings, think hard about when to use "=" versus "contains". Note that if the right hand side of the condition is blank, then "contains" will match all strings, whereas "=" will match none. If you're building a condition, for example, testing the result of a Process Timeline Activity, and the result might be "Accept" or "Do Not Accept", then using a condition like "contains 'accept'" will match either result—probably not what you intended. So it's also a good idea to avoid results that are substrings of one another, like "Approve" and "Disapprove".
6. When configuring Process Timeline activities, enter actual instructions in the **Instructions** property, and descriptions into the **Description** property.
7. When creating new applications, place all of the application's objects, i.e., Forms, Process Timelines, etc., into a single parent folder. This makes importing/exporting applications between development and production environments much easier, as described in the [Importing/Exporting Content topic](#) of the Implementer's guide.
8. Create an application stub in the [Template Library](#) that contains all of the base objects you'd normally create for any new application, e.g., a Form and Process Timeline that are correctly linked, along with a default email template. This enables you to provide all of the initial applications objects, along with any desired Form styling or layout, to enable all new applications to inherit a desired look and feel. This stub should also include the common organizational folders for applications objects, such as Knowledge Views, Forms, etc. Create new applications directly from this



stub. The example below shows an application stub built into the Template Library.



## Email Templates

1. As a guideline, use fewer email templates by using conditional sections and multiple email data controls, when applicable.
2. Use system variables to display activity names, instructions, and descriptions, rather than hard-coding them in the body of the email, so that little customization has to be done to the email template.
3. Configure a default email template in the Process Timeline definition. You can override it and use a different email template if a particular Timeline Activity requires it, by configuring the **Using Email Template** property on the activity's **Notifications** tab.
4. Use the `{EMAIL_USER}` System Variable to specify the email recipient instead of `{CURR_USER}`. Email messages do not have a Current User context.
5. On Forms, use the System Variable Control rather than typing System Variables as text into the Online Form Designer. IT makes the Form's design look cleaner, and protects against inadvertent editing of the System Variable name.

## Development/Testing

1. Your Development and Staging systems should be near-replicas of your production system, including the same users, Meta Data, etc. Global objects, such as Datasource objects should have the same names, and be in the same folder locations on all environments, though they may not be otherwise configured the same. For instance, an "Employee" datasource might connect to real-time data in an employee database on the production system, but a sanitized database on a development system. But, as long as the two objects have the same name, and are stored in the same

file location on both servers, imported applications that refer to these objects will import correctly, and return the appropriate data on each system.

2. Datasource objects, by the way, should not be imported between systems. They should be manually created on each system.
3. When developing, use real users as assignees for Timeline activities. In testing, use impersonation to perform the assignees' tasks. Do not assign tasks to yourself, as you need to see the process as the actual assignees will see the process.
4. In field properties for Process Timeline activities, enter actual instructions in the **Instructions** field, and descriptions into the **Description** field. (Remember, somebody else might be modifying this project after you—help them out!)
5. When you delete a form instance, ensure that you delete the corresponding Process Timeline instance and vice versa. Failure to do leaves orphaned objects in Process Director, i.e., Form instances without their Timelines. All related instance objects need to be deleted together.
6. You shouldn't ever perform testing in the production environment, except in unusual circumstances. Applications should be rigorously tested in the development environment before being imported to Production.

## Forms

1. Use meaningful instance names for each form definition, by setting the Form definition's **Instantiated Form Name** property. By the same token, use simple, meaningful names for object and field definitions. Do not use Hungarian notation or other programming-style conventions. Names should be logical and recognizable. This is very helpful when the objects are used in other places, such as when defining conditions. Object names will appear sorted alphabetically.
2. Think carefully about when the Form's data fields should be editable by users. There are two properties on the Properties tab of the Form definition that control this: **Form Fields are Enabled Only When** and **Entire Form is read-only when**. In general, Form fields should be enabled for new form instances, but disabled in all other contexts. If form information does need to be edited, specify the field that should be edited during the process, and when editing is to be allowed, via visibility conditions on the appropriate fields. **Section** controls are extremely helpful here, because you can place the editable controls in their own **Section**, and enable or disable the **Section** control to enable/disable its constituent controls. Similarly, once the process is complete, the entire Form should be set to read-only, so that it cannot be edited after the process has completed.

Form Fields are Enabled Only When ... ▼	<a href="#">(New Form Instance? = Yes)</a>
Entire form is read-only when	<a href="#">(Timeline Status = Complete)</a>

3. When using tables to provide Form layout, be sure to set the Role attribute, [as described here](#).
4. Instructions should tell users to do something. For example, “Please review this form and approve or reject it using the buttons below”. Note that it’s nice to begin instructions with “Please”. Descriptions should explain something. Always use friendly names for form fields that are required, have validation rules associated with them, or may be used in Knowledge Views, so that Knowledge

Views and other derived uses of the field display human-friendly field names. Keep in mind that Knowledge View column titles will use the friendly name, so brevity is important, as a friendly name like "This is the name of the user who filled out this form" might not be the best choice.

5. Always add the assigned user and task instructions at the top of forms and email templates. In a task context; the easiest way to be sure of that is to use the System Variable control, and set it to use the [Task Instructions System Variable](#), which only appears in that context. You can append the task user's name to the control by typing "{curr\_user}: " in the "Pre" formatter of the variable's attributes, as shown below.

The screenshot shows a dialog box titled "System Variable Control". It has two tabs: "SysVar" and "Comments". The "SysVar" tab is selected. The dialog contains two input fields. The first is labeled "System Variable" and contains the text: `{TASK_INSTRUCTIONS, pre="<strong>{CURR_USER}: </strong>"}`. The second is labeled "Label" and contains the text: "Task Instructions". At the bottom right of the dialog are two buttons: "OK" (green) and "Cancel".

6. As a guideline, on each Form template, there should be three standard fields: the form submitter, submission date, and a unique request number. The three fields should be disabled for all users. In some cases, the submitter may be submitting on behalf of another user: those fields should be separate, so that the submitter is always automatically set (usually by setting its default value to [Form Submitter](#)), and the on-behalf-of user can be selected by the submitter using a user picker or other mechanism. The unique request number can be set via the use of the [Sequence Number](#) System Variable. It's best to set this value in the very first activity of the Process Timeline, so the Sequence Number is generated only after the Form has been submitted.

7. When possible, choose the **Show disabled fields as text** option in the Form definition, which will render disabled fields as plain text, rather than disabled controls.

The screenshot shows the 'Options' configuration window for a form. It contains several sections with various settings:

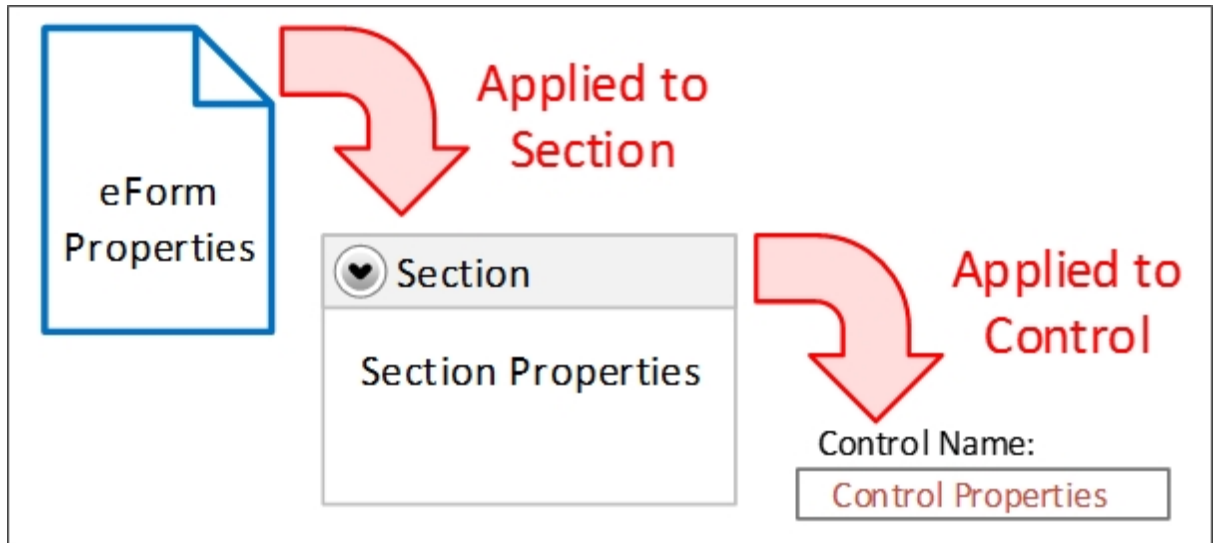
- Options** (header)
- Automatically start this process after Form is submitted (optional): [text input] ...
- Workflows Associated with Form (optional): [text input] ...
- Timelines Associated with Form (optional): [text input] ...
- Default Groupname for Form Instances: [text input]
- Form Script (optional): [text input] ...
- Automatically create an instance of this Case (optional): [text input] ...
- Select a Content List background image for this form (optional): [text input] ...
- Select a background image from this list for this form (optional):
  - Choose background: [dropdown menu]
  - Cover [dropdown menu]
  - Opacity Level 1-100 0 [slider]
- Form Lock Group (optional): [text input]
- Display validation errors with alert boxes:
- Do not show this object in 'Items I Can Run' Knowledge Views:
- Use Visual Basic for scripts (leave unchecked for C#):
- Audit Form field changes:
- Enable Form Locking:
- Display warning if user Cancels form changes:
- Show disabled fields as text:**
- Show only the first validation error (instead of all):
- Hide disabled buttons:
- Remove form from existing Case on submission:
- Enable Form Data Cache:

8. Form control appearance logic can be complicated. The following rules should apply:

- Think about when form control appearance logic should be implemented within a Business Rule, rather than explicitly defined within the field properties. As a rule of thumb, consider a Business Rule when there are three or more conditions controlling the field's appearance, or when the same appearance logic is used on multiple fields. This is true of task assignment in general as well. If you put the assigned user in a Business Rule, you can more easily find and change the user later, as users often change in the course of regular promotions and attrition. Don't name the Business Rule after the person, but rather, the role the person plays in that Process.
- Do not tie a specific user to the appearance logic for a field unless absolutely necessary.
- Avoid using the "Step Running" condition for appearance logic, because the fields you desire to show or hide will only be shown or hidden while that specific activity is running. In all probability, the actual condition you wish to use is "Step Reached", because you wish the fields to be available when the process reaches a specific point, and at all points in the process thereafter. It's very common to have form control logic (enabled/disabled, visible/hidden, required/optional) driven by the state of the process, and the context in which the user is viewing that form (i.e., whether the user is in a task or not).

As a general guideline, use “Activity Reached” when you want appearance based on whether or not the process has reached a certain point, and use “Task Name =” when you want appearance based on whether the user is interacting with the form in the context of a specific task assigned to that user.

- Appearance logic is strongly driven by inheritance. Inheritance is the ability of a container control to automatically incorporate, or inherit, properties of its parent container control. In the case of Form controls, for example, the "required" and "enabled" properties are inherited. The Form is the highest-level container control, and every control on the form is a child control of the Form. A Section control placed onto a Form will inherit the properties of the Form, while an individual control placed inside of a Section control will, in turn, inherit the section's properties.



For example, if the **Required** property of a **Section** control is set to "true", then inheritance ensures that all of the controls contained in that section will be required. You can override inheritance, however, by explicitly setting the **Required** property of a control inside the section to "False". If there is conflict between the **Required** property of a container control, and the controls contained inside it, the control property at the lowest level wins, and overrides the inheritance. With the "lowest level wins" model of inheritance in mind:

- Disable forms when the form isn't a new instance, then enable individual sections as necessary for data input in succeeding activities.
  - Do not use the **Otherwise disabled** or **Otherwise enabled** conditional appearance settings in field properties unless you specifically want to override the appearance logic for the parent container in all cases.
    - Setting a control to "Otherwise Xxxx" when the parent container has its own appearance logic can result in unexpected behavior when the parent container's appearance logic conflicts with that of the control.
9. **Label** control values don't get saved to the database; therefore, labels should be used only to show text on forms. **Label** controls are display vehicles only; don't try to use them in conditions, for example. When possible, associate each label with an **Input** control for [Accessibility](#) compliance.

10. Hide debug sections and other development conventions in forms from the non-developer form users. Users should never see anything on a form that isn't relevant to the process on which they are working.
11. The **Routing Slip** should appear below any user action buttons so that users don't have to scroll below the **Routing Slip** to perform their assigned task.
12. Use the branch/result order properties to ensure that options are presented consistently to users throughout a process. Approvals are always displayed first in the button areas.
13. [Sequence numbers](#) should always be formatted as text, and should always use the "digits" formatter. There are a couple of reasons for this.
  - First, there's a general rule that, if you aren't going to do math on a data item, it's not actually a number, even if all of the characters are numeric. This is why zip codes, or social security numbers are always formatted as text. Additionally, if a data item is formatted as a number, then you can't use the "Contains" operator in the [Condition Builder](#), and can only use the "=" operator. This limitation can be troublesome, because if you leave the value blank in, say, a Knowledge View filter condition that uses the "=" operator, Process Director returns no results, but leaving a "contains" condition blank returns all results. Most of the time, you want to return all results as a default, and then allow the user to filter the results to a smaller set by entering a value for the sequence number, which requires using the "contains" operator in the filter.
  - Formatting the sequence number as text, of course, changes the way sorting works. A numeric field is sorted in numerical order, while a text field is sorted alphabetically, so the same set of sequence numbers will be sorted in different order, depending on whether they are text or numeric data.

NUMERIC SORTING	ALPHABETIC SORTING
1	1
2	100
11	11
21	2
100	21

- This is where the "digits" formatter for the sequence number comes in. If you set the digits formatter to "digits=4", the sequence number will always be a fixed length of four digits, and will use leading zeros for smaller numbers, so that sequence number "1" will be stored as "0001". Adding the leading zeros will fix the Alphabetical sorting issue, so that you can sort by the sequence number in the expected order, e.g., 0001, 0002, 0011, 0021, 0100.

## Knowledge Views

In general, you should always return Form Instances with Knowledge Views unless you have a specific reason to return other objects such as Attachments, Timeline Instances, etc. This is especially true if you are creating filters that use the value of some form field for filtering. Returning, say, Timeline Instances will cause your filter criterion to fail, whereas form field filters will work as expected when returning Form Instances.

## Business Rules, Business Values, and Goals

1. For the purposes of this discussion, we'll refer to Business Rules, Business Values, and Goals collectively as "value objects", since they primarily return a value or values. These value objects occupy a slightly different space than other Process Director objects. Most objects, like Forms or Timelines, are tied to a specific implementation project. Conversely, value objects are often not tied to a specific project at all. For instance, a Business Value that extracts some data from an external ERP or CRM system may be widely used across a number of projects. Similarly, a Goal that specifies some universal system state may be used by all projects. On the other hand, a Business Rule might return a global value that is not tied to a specific application, while a different Business Rule might not work outside the context of a specific Form or Process Timeline definition. This configuration issue raises the question of where the value objects should be placed in the [Content List](#), and when—or if—they should be exported/imported between development and production systems.
2. You may wish to consider whether to create a folder at the root level of the Partition to store value objects that are used by multiple projects, while storing project-centric objects within the relevant project folders. It's often quite easy to determine whether an object is project-centric or not: If the object can be used by any application, such as a Business Value, Dropdown Object, or DataSource object, it should probably be stored in a central location in the [Content List](#), separate from any specific implementation project. BP Logix uses folder names set off with square brackets, e.g., [Business Values] to organize these global objects together in the [Content List](#).
3. Indeed, you might wish to consider this method of centralizing storage for some other objects, as well. For instance, some Business Rules may be used in multiple projects, because they have no dependency on specific forms or other application objects. These Business Rules should also be stored in a central location, while application-specific Business Rules would be stored within the application's parent folder.

## Best Practices for Accessibility

When using Process Director, you should make end user accessibility a priority. Not only is it a nice thing to do in general, but most governments mandate some form of accessibility be implemented so that disabled persons can access web-based content more easily. In the US, this mandate comes from Section 508 of the Americans with Disabilities Act, while, more recently, Canada passed the Accessible Canada Act, which, along with various provincial accessibility laws, now mandates accessibility for all federal public institutions, Crown Corporations, and all federally regulated organizations. Fortunately, most accessibility laws specify conformance with the [Web Content Accessibility Guidelines](#) (WCAG) promulgated by the World Wide Web Consortium (W3C).

In this document, we will examine the WCAG accessibility standards, and discuss some guidance for implementing these standards within Process Director. Process Director provides accessibility tools for the creation of objects that will be accessed by end users—which primarily means Forms—and not for the administrative functions of the product itself. In order to organize these various practices, we will take a look at each of the WCAG standards, and describe the built in accessibility features of Process Director, along with some best practices for implementing each standard.

More detailed information about Process Director's compliance with accessibility standards is provided by the BP Logix Process Director Accessibility Conformance Report (VPAT© Version 2.3) as a [PDF download](#).



Many accessibility features are built in to the product already, and govern specific control behaviors. These control behaviors can be reviewed in [this downloadable PDF document](#).

## Checking for Accessibility #

There are a number of tools you can use to check for accessibility while designing your Process Director forms, but one of the easiest methods for running accessibility checks is to use a browser extension (or "add-on" for Microsoft browsers). Browser extensions are installed into the web browser, and run inside the browser without needing any other external software, so they are fairly easy to use.

Keep a couple of things in mind:

- *We can't specifically recommend* any third-party tool, or browser extension, only inform you about *some* of the available options.
- None of these browser extensions are perfect, and all of them have their own strengths and weaknesses.
- You should choose the accessibility checking tool that best meets your needs, after an appropriate evaluation.

With the above in mind, here are some browser extensions that are used fairly widely.

**Lighthouse:** This isn't actually an extension, but one of the Developer Tools that are already built into Chrome and Chrome-based browsers like Brave. It can perform a variety of checks, one of which is an accessibility check. In Chrome-based browsers, the Developer Tools can be accessed by pressing the [F12] key. Lighthouse has its own tab in the Developer Tools UI.

**WAVE Accessibility Extension:** Available for both Chrome and Firefox. This is one of the oldest and most widely-used browser extensions.

**axe - Web Accessibility Testing:** Available for Chrome, Firefox and Edge.

**Accessibility Insights for Web:** Available for both Chrome and Edge.

**a11yTools - Web Accessibility:** Available for Safari. Safari has a more limited choice of extensions, unfortunately.

**IBM Equal Access Accessibility Checker:** Available for Chrome and Firefox.

As mentioned previously, each of these tools have pros and cons to their use, and some experts recommend using more than one tool to check for accessibility issues.

## Accessibility Principles

Please continue to see the [Accessibility Principles](#) that apply to no code/low code application development with Process Director.



## Accessibility Principles

The W3C has specified three levels of accessibility in the [WCAG Standards](#): Levels A, AA, and AAA. **Most legal requirements specify compliance with Level AA of the WCAG guidelines.** Since that is so, this document won't attempt to provide a listing of all Level AAA requirements. Also, keep in mind that some of the Level AA requirements aren't especially relevant to Process Director either, so we will concentrate on the Level AA requirements that are most relevant. Finally, remember that some requirements simply aren't related to the controls or other features of Process Director, but must be implemented manually, such as giving logical names to form controls, or to the link text you use for hyperlinks. Process Director will automatically implement accessibility standards where it can, but much of the burden remains upon you, as a form designer, to implement accessibility standards as part of your design.

So, with the above in mind, let's take a look at the WCAG standards, and see if we can't provide some helpful guidance.

The WCAG standards are defined by four principles, with guidelines for each principle that specifies how it should be implemented in a web application. The guidelines for each principle are linked below.

**[Principle 1 - Perceivable](#)**: This principle relates to making information and user interface components presentable to users in ways they can perceive.

**[Principle 2 - Operable](#)**: This principle relates to how user interface components and navigation operate.

**[Principle 3 - Understandable](#)**: This principle relates to making information and the operation of the user interface understandable to users.

**[Principle 4 - Robust](#)**: This principle relates to making content robust enough to be interpreted by a wide variety of user agents, including assistive technologies.

### Accessibility Principle 1 – Perceivable

This principle relates to making information and user interface components presentable to users in ways they can perceive.

#### Guideline 1.1 – Text Alternatives

*Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language.*

##### 1.1.1 Non-text Content

*Level A - All non-text content that is presented to the user has a text alternative that serves the equivalent purpose.*

In the context of a Process Director form or Dashboard, this requirement is primarily applicable to the use of images, including logos and other branding images. Image controls in Process Director's Online Form Designer contain an **Alternative Text** property. This property should be used to provide descriptive text for all images you display on forms.

Similarly, **Button** controls have an **Alt Text** property that you should use to provide descriptive text to buttons that only present images or icons.

All data entry controls should have a **Label** control associated with them. In the properties dialog of the **Label** control, the **Associated Control ID** property should contain the **Name** of the control to which the

**Label** is associated. For instance, if you have an **Input** control with the **Name** "FirstName", the **Associated Control ID** of the **Label** would be "FirstName".

Tables have a **Caption** property to provide text-based table descriptions in the **Table Properties** dialog box.

## Guideline 1.2 – Time-based Media

*Provide alternatives for time-based media.*

This isn't generally relevant to Process Director, as forms don't display video or audio presentations.

## Guideline 1.3 – Adaptable

*Create content that can be presented in different ways (for example simpler layout) without losing information or structure.*

### 1.3.1 Info and Relationships

*Level A - Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text.*

You should use tables to present tabular data, and use the appropriate table headers. When doing so, Process Director's Online Form Designer will automatically insert the appropriate table, th, tr, td, etc., tags. You can specify the headers as the first row, first column, or both, using the **Headers** property of the Table's properties Dialog. Additionally, the **Cell Properties** dialog box has a **Cell Type** property you can use to specify if a table is a data or header cell.

Process Director automatically provides the HTML "scope" attribute for headers to associate header cells and data cells in data tables.

### 1.3.2 Meaningful Sequence

*Level A - When the sequence in which content is presented affects its meaning, a correct reading sequence can be programmatically determined.*

This isn't generally relevant to Process Director.

### 1.3.3 Sensory Characteristics

*Level A - Instructions provided for understanding and operating content don't rely solely on sensory characteristics of components such as shape, color, size, visual location, orientation, or sound.*

You should always provide textual identification of items that otherwise rely only on sensory information to be understood. In other words, image-only buttons, images used as buttons, or other UI conventions should always provide a textual identification in addition to the sensory-based convention, e.g., using explicit text labels for buttons, in addition to images or icons, where possible. There are exceptions when space constraints may not allow the use of such explicit text, in which case you must use the **Alt Text** property of the object to ensure a text alternative is always available for it.

### 1.3.4 Orientation

***Level AA - Content doesn't restrict its view and operation to a single display orientation, such as portrait or landscape, unless a specific display orientation is essential.***

As a best practice, you shouldn't fix widths for layout elements if doing so isn't essential. By default, for example, the default width of tables in the Online Form Designer is 100%, which enables them to scale to the screen width of the viewport. You shouldn't impose a specific screen size or viewport on your users, or attempt to replicate the exact layout of a printed form. Users who require a larger font, for example, may find such forms more difficult to read or use because of the layout restrictions you've imposed.

### 1.3.5 Identify Input Purpose

***Level AA - The purpose of each input field collecting information about the user can be programmatically determined***

As a best practice, all input fields should have logical, recognizable **Name** properties that reflect the type of data being collected by the input control, rather than use among conventions that make sense only to developers or other IT personnel. For instance, "FirstName" tells you fairly specifically what the purpose of the input control is, while "frm11a\_txt\_fname" is substantially more opaque.

## Guideline 1.4 – Distinguishable

***Make it easier for users to see and hear content including separating foreground from background.***

### 1.4.1 Use of Color

***Level A - Color isn't used as the only visual means of conveying information, indicating an action, prompting a response, or distinguishing a visual element.***

As a best practice, any information conveyed by color differences should also be available in text. Use of color cues alone to display a status, or prompt a response isn't an acceptable accessibility practice. Color should be an enhancement to, and not the primary method of, communication.

### 1.4.2 Audio Control

***Level A - If any audio on a Web page plays automatically for more than 3 seconds, either a mechanism is available to pause or stop the audio, or a mechanism is available to control audio volume independently from the overall system volume level.***

This is generally not relevant to Process Director.

### 1.4.3 Contrast (Minimum)

**Level AA - The visual presentation of text and images of text has a contrast ratio of at least 4.5:1, except the following:**

**Large Text: Large-scale text and images of large-scale text have a contrast ratio of at least 3:1;**

**Incidental: Text or images of text that are part of an inactive user interface component, that are pure decoration, that aren't visible to anyone, or that are part of a picture that contains significant other visual content, have no contrast requirement.**

**Logotypes: Text that is part of a logo or brand name has no contrast requirement.**

As a best practice, text should be easily distinguishable from any background colors or images through the use of highly contrasting colors. Without sufficient contrast such text will be difficult for to read by users who are visually impaired. The WCAG specification links to [a number of web-based tools](#) that are available online for analyzing color contrasts to ensure that your design meets the required contrast ratios. In addition, we've provided [a set of sample colors](#) you might want to consider for creating the appropriate contrasts with white or black text, for use with UI elements like buttons.

### 1.4.4 Resize text

**Level AA - Except for captions and images of text, text can be resized without assistive technology up to 200 percent without loss of content or functionality.**

Most modern web browsers have 200%+ text resizing built in. As a form designer, you should remember that the text size in which you are designing the form may not be the size in which your users are viewing it. As a best practice, you should view your form at different text sizes, up to 200%, to ensure that the layout and usability of your form remains intact at high zoom levels. This also relates to the best practice identified for WCAG specification 1.3.4, above, as this requirement affect the proposed layout of your form at high magnifications.

### 1.4.5 Images of Text

**Level AA - If the technologies being used can achieve the visual presentation, text is used to convey information rather than images of text except for the following:**

**Customizable: The image of text can be visually customized to the user's requirements;**

**Essential: A particular presentation of text is essential to the information being conveyed.**

**Note 1: Logotypes (text that is part of a logo or brand name) are considered essential.**

As a best practice, use text, rather than images of text, wherever possible.

### 1.4.10 Reflow

**Level AA - Content can be presented without loss of information or functionality, and without requiring scrolling in two dimensions.**

Though there are some minor exceptions to this requirement in the WCAG 1.4.10 standard in general, you should ensure that your users don't have to scroll in more than one dimension, i.e., vertical scrolling alone is acceptable, as is horizontal scrolling alone, but requiring users to scroll both vertically and horizontally isn't acceptable. (As a best practice, horizontal scrolling should be eliminated entirely if it is possible to do so.)

When using installations with the mobile device component, Process Director will automatically reflow content, such as tables, to ensure it fits within smaller viewports without having to scroll horizontally. Otherwise, some relatively simple CSS styling applied to the form in the **Properties** tab of the Form Definition can accomplish the same refactoring to control the reflow of a table. A sample of this sort of reflow CSS is provided below.

### Code Sample

```
@media
only screen and (max-width: 860px),
(min-device-width: 860px) and (max-device-width: 1024px) {
  /* Force table to not be like tables anymore */
  table, thead, tbody, th, td, tr {
    display: block;
  }

  /* Hide table headers (but not display: none;, for accessibility) */
  thead tr {
    position: absolute;
    top: -9999px;
    left: -9999px;
  }
  tr {
    border-top: 1px solid #e0e0e0;
    margin-top: 10px;
  }
}
```

For many UI elements, such as tables, or controls like the **Section** control, Process Director sets the default **Width** of the element to 100% of screen width automatically. In most use cases, this ensures that the element fills the screen horizontally, but doesn't extend beyond the screen's width. Images, however, can be particularly troublesome in this regard, since an image that displays well on a desktop screen may be too wide to display properly on a tablet or phone. These large images will push the effective screen width beyond the horizontal size of the screen, which will also force the vertical size of other elements off the edge of the screen. You may wish to set the CSS style for images (i.e., the `img` element, to a **max-width** of `100%` to ensure that, as screen widths change, the image never extends beyond the screen's width. This CSS style forces the width of the element to always be no larger than 100% of screen width, and enforces this setting by forcing the image to resize to the screen width when the screen is resized. This style can easily be applied in the **Image** control's properties by adding this style setting to the image in the **Style** property.

For the standard **Image** control, the **Style** property can be found on the **Advanced** tab of the **Properties** dialog box for the control.

### Image Properties

Image Info   Link   Upload   Advanced   ◀ ▶

Id    Language Direction    Language Code

Long Description URL

Stylesheet Classes    Advisory Title

Style

For the BP Logix-specific **Image** control that's accessed via the **Other Controls** menu, the **Style** property is displayed on the **Image Control** tab of it's Properties dialog box.

### Image Control

Image Control   Field Properties   Comments   ◀ ▶

Name  ?

Image URL

URL

Css Class

Style

HTML Height



Once set, the image will always display properly, as shown in the examples below.

### Image Example

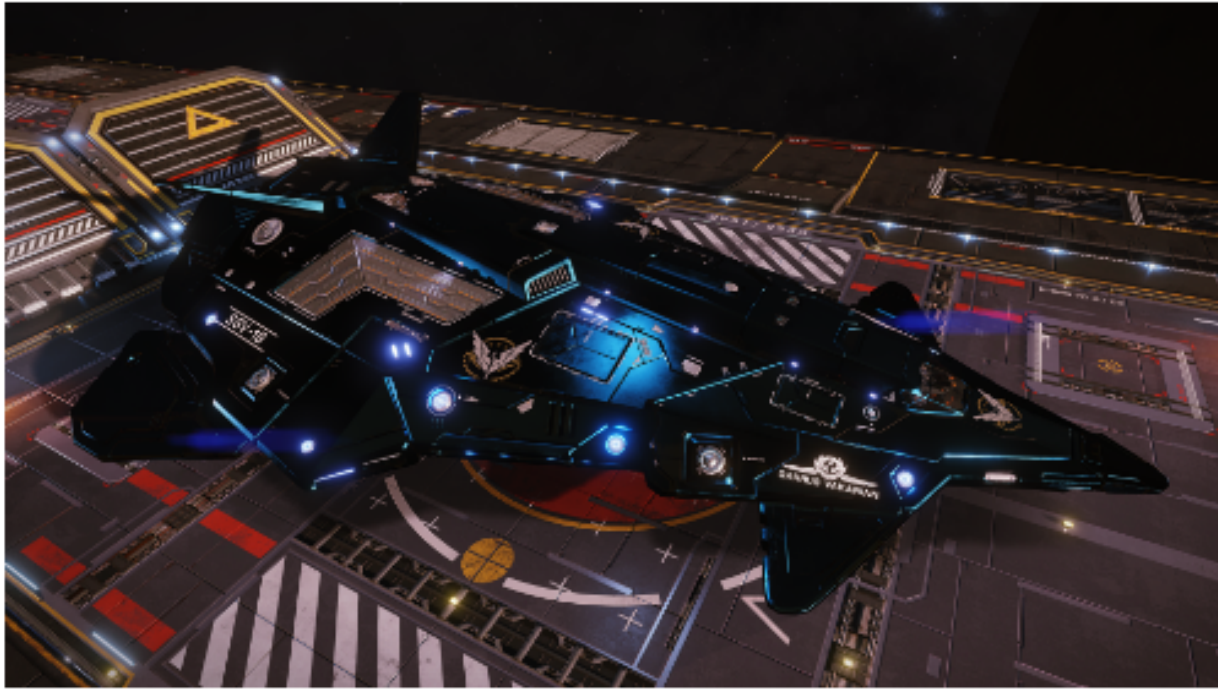
Wide Screens:



Narrow Screens:

Text 5

Text 6



#### 1.4.11 Non-text Contrast

*Level AA - The visual presentation of the following have a contrast ratio of at least 3:1 against adjacent color(s):*

*User Interface Components: Visual information required to identify user interface components and states, except for inactive components or where the appearance of the component is determined by the user agent and not modified by the author;*

*Graphical Objects: Parts of graphics required to understand the content, except when a particular presentation of graphics is essential to the information being conveyed.*

As a best practice, and as with the text contrast requirements of 1.4.3, above, user interface components and other graphical object must have a relatively high contrast. For example, a button with blue text and a slightly darker blue background might look nice, but will probably not have enough contrast between the button's background color and text to meet the requirements of this specification. In Process Director, button background, text, and icon colors can all be specified b the designer, so ensure that the color contrasts are large enough for visually impaired users to easily differentiate between them. Again refer to the contrast analyzer tools linked in 1.4.3, above.



### 1.4.12 Text Spacing

**Level AA** - In content implemented using markup languages that support the following text style properties, no loss of content or functionality occurs by setting all of the following and by changing no other style property:

*Line height (line spacing) to at least 1.5 times the font size;*

*Spacing following paragraphs to at least 2 times the font size;*

*Letter spacing (tracking) to at least 0.12 times the font size;*

*Word spacing to at least 0.16 times the font size.*

**Exception:** Human languages and scripts that don't make use of one or more of these text style properties in written text can conform using only the properties that exist for that combination of language and script.

Though this is generally not relevant to Process Director, you should, as a best practice, not make the organization or presentation of information dependent on line, letter, or paragraph spacing. This is also a layout consideration that affects the requirements of both 1.3.4, and 1.4.4, above.

### 1.4.13 Content on Hover or Focus

**Level AA** - Where receiving and then removing pointer hover or keyboard focus triggers additional content to become visible and then hidden, the following are true:

**Dismissible:** A mechanism is available to dismiss the additional content without moving pointer hover or keyboard focus, unless the additional content communicates an input error or doesn't obscure or replace other content;

**Hoverable:** If pointer hover can trigger the additional content, then the pointer can be moved over the additional content without the additional content disappearing;

**Persistent:** The additional content remains visible until the hover or focus trigger is removed, the user dismisses it, or its information is no longer valid.

**Exception:** The visual presentation of the additional content is controlled by the user agent and isn't modified by the author.

While not generally relevant to Process Director in the specific context of a mouseover or hover action, there are times when you want information to appear or disappear based on the value of some evaluable condition, or when you select a control. Process Director provides both [Section](#) and [Embedded Section](#) controls that enable you to hide or reveal form areas. When shown, these sections are persistent until the condition changes or it is hidden manually. Also, this hiding or displaying of information is generally done via events other than mouseover, in order to provide both persistence, and user-level manual control over when the information is hidden or displayed. As a general rule, attempting to show or hide information based on the position of the mouse isn't a good practice.

### Other Accessibility Principles

[Principle 2 - Operable](#): This principle relates to how user interface components and navigation operate.

[Principle 3 - Understandable](#): This principle relates to making information and the operation of the user interface understandable to users.

**Principle 4 - Robust:** This principle relates to making content robust enough to be interpreted by a wide variety of user agents, including assistive technologies.

### Sample Accessible Colors

For your convenience, we've provided a list of colors and their contrast value that you make want to consider for use in UI elements, such as buttons. You can, for instance, add these colors to your customization file using the [WorkflowColors Custom Variable](#).

The colors presented below are grouped by shade, with the appropriate hexadecimal HTML colors and level of WCAG compliance, i.e., AA or AAA. Most of the colors are AAA-compliant. The color names we've provided are arbitrary, and do not necessarily correspond to the formal HTML color names (though some do).

Please note that the HTML named colors **Red** and **Green**, are particularly difficult to use at the AAA compliance level. These mid-range colors do not particularly contrast well with either white or black text. You may wish to replace these colors with the colors we've designated as **Red (Contrast)** [B10000] and **Green (Contrast)** [006200] below, both of which are dark enough to be AAA-compliant when contrasted with white.

You can, of course, check contrasts for your own UI colors at [ContrastChecker.com](#), or many others available online.

### Compliance Levels

**AA:** Contrast ratio > 4.5:1

**AA\*:** Contrast ratio > 3:1 for 18pt or larger text.

**AAA:** Contrast ratio > 7:1

**AAA\*:** Contrast ratio > 4.5:1 for 18pt or larger text.

### Suggested Colors

Shade	Hexadecimal Color	Suggested Name	Contrast Ratio With Black/White Text	WGAC Compliance Level
Grayscale	000000	Black	21	AAA
	333333	Dark Gray	12.63	AAA
	555555	Gray	7.46	AAA
	CCCCCC	Light Gray	13.08	AAA
	F1F1F1	Smoke	18.59	AAA
	FFFFFF	White	21	AAA
Blue	000066	Dark Blue	17.62	AAA
	0000FF	Blue	8.59	AAA
	96B6F5	Light Blue	10.31	AAA
	E2E2FE	Pale Blue	16.54	AAA

Shade	Hexadecimal Color	Suggested Name	Contrast Ratio With Black/White Text	WGAC Compliance Level
Purple	2C0047	Dark Purple	17.53	AAA
	6300A6	Purple	10.09	AAA
	C972FF	Lavender	7.33	AAA
	ECCEFF	Pale Lavender	14.84	AAA
Red	5C0000	Dark Red	14.43	AAA
	B10000	Red (Contrast)	7.08	AAA
	FF0000	Red (HTML)	5.25	AA, AAA*
	FF0000	Red (HTML)	4	AA*
	FF7F7F	Light Red	8.59	AAA
	FFC0C0	Pink	13.55	AAA
Orange	532D00	Brown	12.05	AAA
	FA8800	Orange	8.59	AAA
	FFBB69	Peach	12.56	AAA
	FFE3C0	Buff	17	AAA
Yellow	5E5400	Dark Yellow	7.65	AAA
	FFE600	Bold Yellow	16.57	AAA
	FFFF00	Yellow	19.56	AAA
	FBF3AA	Light Yellow	18.5	AAA
Green	003300	Dark Green	14.25	AAA
	006200	Green (Contrast)	7.64	AAA
	008000	Green (HTML)	4.09	AA*
	008000	Green (HTML)	5.14	AA, AAA*
	7CFF7C	Light Green	16.45	AAA
	C9FFC9	Mint	18.63	AAA
Teal	002D40	Dark Teal	14.49	AAA
	035D79	Teal	7.38	AAA
	2AEEFF	Light Teal	14.77	AAA
	C7FBFF	Mist	18.67	AAA

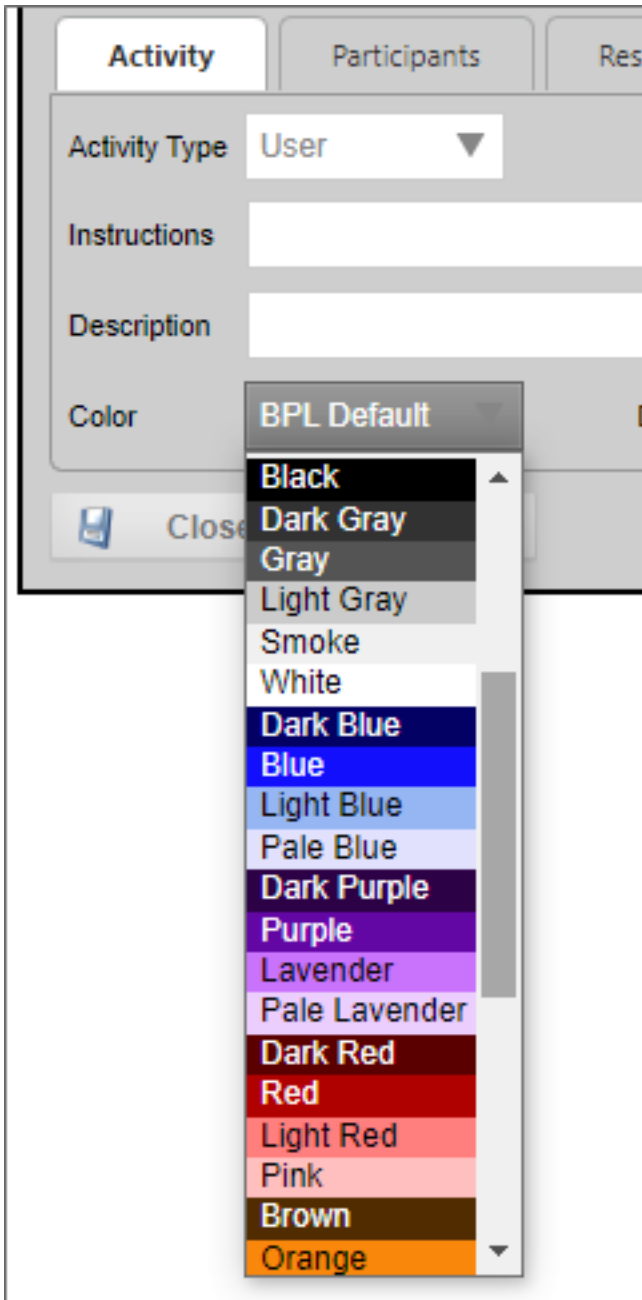
Should you wish to add the color set above to your installation, you can copy the code below, as is, into the `PreSetSystemVars()` section of your Process Director Installation's Custom Variables file.

```

bp.Vars.WorkflowColors = new List<ColorValue>();
bp.Vars.WorkflowColors.Add(new ColorValue("Default", "#054FB9", "#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Black", "#000000", "#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Dark Gray", "#333333", "#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Gray", "#555555", "#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Light Gray", "#CCCCCC", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Smoke", "#F1F1F1", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("White", "#FFFFFF", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Dark Blue", "#000066", "#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Blue", "#0000FF", "#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Light Blue", "#96B6F5", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Pale Blue", "#E2E2FE", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Dark Purple", "#2C0047", "#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Purple", "#6300A6", "#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Lavender", "#C972FF", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Pale Lavender", "#ECCEFF",
"#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Dark Red", "#5C0000", "#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Red", "#B10000", "#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Light Red", "#FF7F7F", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Pink", "#FFC0C0", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Brown", "#532D00", "#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Orange", "#FA8800", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Peach", "#FFBB69", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Buff", "#FFE3C0", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Dark Yellow", "#5E5400", "#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Bold Yellow", "#FFE600", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Yellow", "#FFFF00", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Light Yellow", "#FBF3AA", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Dark Green", "#003300", "#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Green", "#006200", "#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Light Green", "#7CFF7C", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Mint", "#C9FFC9", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Dark Teal", "#002D40", "#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Teal", "#035D79", "#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Light Teal", "#2AEFFF", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Mist", "#C7FBFF", "#000000"));

```

The code above will **replace** the default colors that are installed with Process Director with the new color set shown above. You can also simply add these colors to the existing default colors by deleting the first two lines of code in the example above. Once applied, this color set will appear in the UI's color chooser dropdowns like the example below.



### ***Accessibility Principles***

**Principle 1 - Perceivable:** This principle relates to making information and user interface components presentable to users in ways they can perceive.

**Principle 2 - Operable:** This principle relates to how user interface components and navigation operate.

**Principle 3 - Understandable:** This principle relates to making information and the operation of the user interface understandable to users.

**Principle 4 - Robust:** This principle relates to making content robust enough to be interpreted by a wide variety of user agents, including assistive technologies.

## Accessibility Principle 2 – Operable

This principle relates to how user interface components and navigation operate.

### 2.1.1 Keyboard

**Level A - All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes, except where the underlying function requires input that depends on the path of the user's movement and not just the endpoints.**

Process Director uses HTML controls and links for all form controls. All user-accessible objects are navigable via keyboard, e.g., users can use the tab key to navigate from one control to the next. The path of the user's movement is controlled by the location of the control on the form. Navigation operates from left to right and from top to bottom for browsers that have European language settings, and from right to left and top to bottom for other browser language settings. Additionally, the use of the Set Focus Custom Task enables you to specify a custom tab order, if you desire, though this is generally not necessary in the context of a form, which generally has a logical structure to begin with, and corresponds with the natural reading order of the specified culture.

### 2.1.2 No Keyboard Trap

**Level A - If keyboard focus can be moved to a component of the page using a keyboard interface, then focus can be moved away from that component using only a keyboard interface, and, if it requires more than unmodified arrow or tab keys or other standard exit methods, the user is advised of the method for moving focus away.**

**Note 1: Since any content that doesn't meet this success criterion can interfere with a user's ability to use the whole page, all content on the Web page (whether it is used to meet other success criteria or not) must meet this success criterion.**

Though all controls on a form are keyboard navigable automatically through a set tab order in Process Director, it is possible for you to create a keyboard trap when using the Set Focus Custom Task. For instance, you can set the focus to field 3 when a customer completes field 1, then, in field 3, set the focus back to field one. This would create a never-ending tab cycle between field 1 and field 3, thus creating a keyboard trap, requiring the user to use the mouse to navigate to a different field. You should, as a best practice, extensively test any use of the Set Focus Custom Task to ensure you don't create a keyboard trap.

### 2.1.3 Keyboard (No Exception)

**Level AAA - All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes.**

Process Director's tab order is generally compliant with this specification, though, as mentioned in 1.2.1, above, you can, as a designer, break it.

## 2.1.4 Character Key Shortcuts

*Level A- If a keyboard shortcut is implemented in content using only letter (including upper- and lower-case letters), punctuation, number, or symbol characters, then at least one of the following is true:*

*Turn off: A mechanism is available to turn the shortcut off;*

*Remap: A mechanism is available to remap the shortcut to include one or more non-printable keyboard keys (e.g., Ctrl, Alt);*

*Active only on focus: The keyboard shortcut for a user interface component is only active when that component has focus.*

This is generally not relevant to Process Director.

## Guideline 2.2 – Enough Time

*Provide users enough time to read and use content.*

### 2.2.1 - Timing Adjustable

*Level A - For each time limit that is set by the content, at least one of the following is true:*

*Turn off: The user is allowed to turn off the time limit before encountering it; or*

*Adjust: The user is allowed to adjust the time limit before encountering it over a wide range that is at least ten times the length of the default setting; or*

*Extend: The user is warned before time expires and given at least 20 seconds to extend the time limit with a simple action (for example, "press the space bar"), and the user is allowed to extend the time limit at least ten times; or*

*Real-time Exception: The time limit is a required part of a real-time event (for example, an auction), and no alternative to the time limit is possible; or*

*Essential Exception: The time limit is essential and extending it would invalidate the Activity; or*

*20 Hour Exception: The time limit is longer than 20 hours.*

This is generally not relevant to Process Director forms, as they aren't content-timed. This may be relevant to Knowledge Views, however, which do have a [Automatically refresh results \(in seconds\)](#) property available on the [Configure](#) tab of the Knowledge View Definition. As a best practice, ensure that you don't impose automatic refreshes on Knowledge Views unless it is essential for real-time event viewing.



## 2.2.2 Pause, Stop, Hide

*Level A - For moving, blinking, scrolling, or auto-updating information, all of the following are true:*

*Moving, blinking, scrolling: For any moving, blinking or scrolling information that (1) starts automatically, (2) lasts more than five seconds, and (3) is presented in parallel with other content, there is a mechanism for the user to pause, stop, or hide it unless the movement, blinking, or scrolling is part of an Activity where it is essential; and*

*Auto-updating: For any auto-updating information that (1) starts automatically and (2) is presented in parallel with other content, there is a mechanism for the user to pause, stop, or hide it or to control the frequency of the update unless the auto-updating is part of an Activity where it is essential.*

This is generally not relevant to Process Director.

## 2.2.3 No Timing

*Level AAA - Timing isn't an essential part of the event or Activity presented by the content, except for non-interactive synchronized media and real-time events.*

Process Director is generally compliant with the specification, unless an automatic refresh is intentionally implemented by the designer, as in the case of Knowledge Views, as mentioned in 2.2.1, above.

## 2.2.4 Interruptions

*Level AAA - Interruptions can be postponed or suppressed by the user, except interruptions involving an emergency.*

Process Director is generally compliant with this specification, as users control the timing of data submission by manual action.

## 2.2.5 Re-authenticating

*Level AAA - When an authenticated session expires, the user can continue the Activity without loss of data after re-authenticating.*

Reauthentication isn't required in process Director by default, though designers can impose it. For example, in the **Advanced** tab of a User Timeline Activity, there is a **Re-authenticate users** property that can be set to require a user to reauthenticate *prior* to performing an Activity, but this reauthentication occurs before the data entry session is invoked. While it is possible to set up more onerous reauthentication requirements outside of Process Director, i.e., at the network security level, and Process Director will respect those requirements, Process Director doesn't, by default, require reauthentication or time users out.

## 2.2.6 Timeouts

*Level AAA - Users are warned of the duration of any activity that could cause data loss, unless the data is preserved for more than 20 hours when the user doesn't take any actions.*

As mentioned in 2.2.5, above, this isn't generally relevant to Process Director.



## Guideline 2.3 – Seizures and Physical Reactions

*Do not design content in a way that is known to cause seizures or physical reactions.*

### 2.3.1 Three Flashes or Below Threshold

*Level A - Web pages don't contain anything that flashes more than three times in any one second period, or the flash is below the general flash and red flash thresholds.*

As a best practice, designers should never include such elements in an object viewable by end users.

### 2.3.2 Three Flashes

*Level AAA - Web pages don't contain anything that flashes more than three times in any one second period.*

As a best practice, designers should never include such elements in an object viewable by end users.

### 2.3.3 Animation from Interactions

*Level AAA - Motion animation triggered by interaction can be disabled, unless the animation is essential to the functionality or the information being conveyed.*

As a best practice, designers should never include such elements in an object viewable by end users.

## Guideline 2.4 – Navigable

*Provide ways to help users navigate, find content, and determine where they are.*

### 2.4.1 Bypass Blocks

*Level A - A mechanism is available to bypass blocks of content that are repeated on multiple Web pages.*

While it's difficult to think of a use case in which this might be relevant, the [Section](#) and [Embedded Section](#) controls can be used to enable users to show or hide repeated blocks of content, as a best practice.

### 2.4.2 Page Titled

*Level A - Web pages have titles that describe topic or purpose.*

All user-viewable objects in Process Director have an instantiated name property that can be customized. This instance name serves as the page title when displayed to end users. As a best practice, designers should provide the appropriate instance names for every object. By default, when an object is created, a generic instance name is created for that object. For example, when creating a new form definition, the default [Instantiated Form Name](#) is "{FORM\_DEF\_NAME} Submitted On {CREATE\_DATE}". When a new instance of a form definition named "Vacation Request" is created on 1 January, 2020, the user will see the title of the page as "Vacation Request Submitted on 1/1/2020".

### 2.4.3 Focus Order

*Level A - If a Web page can be navigated sequentially and the navigation sequences affect meaning or operation, focusable components receive focus in an order that preserves meaning and operability.*

This specification is generally covered by the answers provided in 2.2.1-2.2.3 above, along with the same caveat that the use of the Set Focus Custom Task requires extensive testing to ensure that the focus is properly navigable by end users.

## 2.4.4 Link Purpose (In Context)

*Level A - The purpose of each link can be determined from the link text alone or from the link text together with its programmatically determined link context, except where the purpose of the link would be ambiguous to users in general.*

As a best practice, designers creating hyperlinks using the **Hotlink** control, or manually creating hyperlinks in the raw HTML of an **HTML** control, should comply with this requirement, so that end user know the context and purpose of the link before clicking it.

## 2.4.5 Multiple Ways

*Level AA - More than one way is available to locate a Web page within a set of Web pages except where the Web Page is the result of, or a step in, a process.*

In addition to the links provided to objects in the user's **Task List**, or in the **Forms I can Submit** Knowledge View, designer should, as a best practice, incorporate navigation buttons and/or dashboard/workspace links to all forms or other objects to which users have access. Workspace, dashboard, and Desktop interface links are fully configurable by designers.

## 2.4.6 Headings and Labels

*Level AA - Headings and labels describe topic or purpose.*

In addition to the automatic heading incorporated into Process Director's **Section** controls, The Online Form Designer also incorporates Header formatting for other content. As a best practice, designers should ensure that all heading and labels are appropriately descriptive.

## 2.4.7 Focus Visible

*Level AA - Any keyboard operable user interface has a mode of operation where the keyboard focus indicator is visible.*

Process Director is generally compliant with this specification, and highlights the field/control that has the focus when using keyboard navigation, i.e., tabbing between controls.

## Guideline 2.5 – Input Modalities

*Make it easier for users to operate functionality through various inputs beyond keyboard.*

### 2.5.1 Pointer Gestures

*Level A - All functionality that uses multipoint or path-based gestures for operation can be operated with a single pointer without a path-based gesture, unless a multipoint or path-based gesture is essential.*

This is generally not relevant to Process Director.

## 2.5.2 Pointer Cancellation

*Level A - For functionality that can be operated using a single pointer, at least one of the following is true:*

*No Down-Event: The down-event of the pointer isn't used to execute any part of the function;*  
*Abort or Undo: Completion of the function is on the up-event, and a mechanism is available to abort the function before completion or to undo the function after completion;*  
*Up Reversal: The up-event reverses any outcome of the preceding down-event;*  
*Essential: Completing the function on the down-event is essential.*

This is generally not relevant to Process Director.

## 2.5.3 - Label in Name

*Level A - For user interface components with labels that include text or images of text, the name contains the text that is presented visually.*

This is generally not relevant to Process Director, as all labels are text-based.

## 2.5.4 Motion Actuation

*Level A - Functionality that can be operated by device motion or user motion can also be operated by user interface components and responding to the motion can be disabled to prevent accidental actuation.*

This is generally not relevant to Process Director.

## 2.5.5 Target Size

*Level AA - The size of the target for pointer inputs is at least 44 by 44 CSS pixels except when:*  
*Equivalent: The target is available through an equivalent link or control on the same page that is at least 44 by 44 CSS pixels;*

*Inline: The target is in a sentence or block of text;*

*User Agent Control: The size of the target is determined by the user agent and isn't modified by the author;*

*Essential: A particular presentation of the target is essential to the information being conveyed.*

As a best practice, designers should ensure that user interface targets are sized appropriately.

## Other Accessibility Principles

**[Principle 1 - Perceivable](#):** This principle relates to making information and user interface components presentable to users in ways they can perceive.

**[Principle 3 - Understandable](#):** This principle relates to making information and the operation of the user interface understandable to users.

**[Principle 4 - Robust](#):** This principle relates to making content robust enough to be interpreted by a wide variety of user agents, including assistive technologies.

## Accessibility Principle 3 – Understandable

This principle relates to making information and the operation of the user interface understandable to users.

### Guideline 3.1 – Readable

*Make text content readable and understandable.*

#### 3.1.1 Language of Page

*Level A - The default human language of each Web page can be programmatically determined.*

Process Director's entire user interface is customizable for culture/language, based on the user's selected culture setting. All forms and other user-viewable objects are also fully customizable for culture as well. Please see the [Localization](#) topic of the Developers Guide.

#### 3.1.2 Language of Parts

*Level AA - The human language of each passage or phrase in the content can be programmatically determined except for proper names, technical terms, words of indeterminate language, and words or phrases that have become part of the vernacular of the immediately surrounding text.*

This is generally not relevant to Process Director, though designers can, through the use of the **HTML** control, provide the appropriate language indicators in the raw HTML via the "lang" attribute of an HTML text tag.

#### Code Sample

```
<blockquote lang="de">
  <p>
    Da dachte der Herr daran, ihn aus dem Futter zu schaffen,
    aber der Esel merkte, daß kein guter Wind wehte, lief fort
    und machte sich auf den Weg nach Bremen: dort, meinte er,
    könnte er ja Stadtmusikant werden.
  </p>
</blockquote>
```

#### 3.1.3 Unusual Words

*Level AAA - A mechanism is available for identifying specific definitions of words or phrases used in an unusual or restricted way, including idioms and jargon.*

This isn't generally relevant to Process Director, though designers can, through the use of the HTML control, use the "dfn" tag to specify a definition.

#### Code Sample

```
<p>The Web Content Accessibility Guidelines require that non-text content has a text alternative. <dfn>Non-text content</dfn> is content that isn't a sequence of characters that can be programmatically determined or where the sequence isn't expressing something in human language; this includes ASCII Art (which is a pattern of characters), emoticons, leetspeak (which is character substitution), and images representing text .</p>
```

## Guideline 3.2 – Predictable

*Make Web pages appear and operate in predictable ways.*

### 3.2.1 On Focus

*Level A - When any user interface component receives focus, it doesn't initiate a change of context.*

This is generally not relevant to Process Director. To the extent context is changed, it is changed as the result of a manual user input, not a result of simply gaining focus.

### 3.2.2 On Input

*Level A - Changing the setting of any user interface component doesn't automatically cause a change of context unless the user has been advised of the behavior before using the component.*

As a best practice, designers should provide users with explanatory text that a change to a user input will change context, so that users understand the result of any input action.

### 3.2.3 Consistent Navigation

*Level AA - Navigational mechanisms that are repeated on multiple Web pages within a set of Web pages occur in the same relative order each time they are repeated, unless a change is initiated by the user.*

As a best practice, designers should ensure that any navigation scheme they create is consistent, so that the user receives the same experience over time. This is important in form design, particularly in the use of result buttons for user action buttons displayed on a form when the user is in the context of a task. The order of positive and negative result buttons, their colors, and, to the extent possible, their text, should be the same in all forms. In other words, one form shouldn't present the options Approve, Deny, while a different form displays them as Deny, Approve. The user experience should be as consistent as possible.

### 3.2.4 Consistent Identification

*Level AA - Components that have the same functionality within a set of Web pages are identified consistently.*

As a best practice, similar to 3.2.3, above, component definitions and explanations should be consistent.

## Guideline 3.3 – Input Assistance

*Help users avoid and correct mistakes.*

### 3.3.1 Error Identification

*Level A - If an input error is automatically detected, the item that is in error is identified and the error is described to the user in text.*

In addition to the default test notifications for required fields, wrong data types, etc., Process Director enables designers to create their own custom validation conditions and associated text notifications. As a best practice, always ensure that your validation notifications are helpful and descriptive. By default, error notifications appear in text banners at the top and bottom of the form, though designers can change these locations through the use of the [Form Error/Info String](#) controls in the form.

### 3.3.2 Labels or Instructions

*Level A - Labels or instructions are provided when content requires user input.*

Process Director provides a number of methods for designers to accomplish this. In addition to the Label controls, input fields have an **Empty Text** property and **Tooltip** property that should be used to provide brief instructions about what information goes into the field. Additional instructions can also be provided via alert boxes using the Alert Custom Task.

### 3.3.3 Error Suggestion

*Level AA - If an input error is automatically detected and suggestions for correction are known, then the suggestions are provided to the user, unless it would jeopardize the security or purpose of the content.*

As a best practice, designers should include appropriate error correction suggestions in the error message they configure.

### 3.3.4 Error Prevention (Legal, Financial, Data)

*Level AA - For Web pages that cause legal commitments or financial transactions for the user to occur, that modify or delete user-controllable data in data storage systems, or that submit user test responses, at least one of the following is true:*

*Reversible: Submissions are reversible.*

*Checked: Data entered by the user is checked for input errors and the user is provided an opportunity to correct them.*

*Confirmed: A mechanism is available for reviewing, confirming, and correcting information before finalizing the submission.*

While the onus of incorporating this specification lies mainly on the process designer, Process Director provides a number of tools such as alert boxes in the form, confirmation requirements in the **Results** tab of User Timeline Activities, requiring text comments for a specified results, custom validation rules, and many more. Additionally, the Compliance component enables form field auditing when turned on via the **Audit Form field changes** property of the Form definition. Implementing this type of error prevention can be complex, so BP Logix is always ready to offer Direct Assistance for such implementations.

### 3.3.5 Help

*Level AAA - Context-sensitive help is available.*

Process Director provides a number of ways to provide context sensitive help, such as tooltips, empty text for form fields, and callable alert boxes. Designers should, as a best practice, incorporate context-sensitive help appropriately.

### Other Accessibility Principles

**Principle 1 - Perceivable:** This principle relates to making information and user interface components presentable to users in ways they can perceive.

**Principle 2 - Operable:** This principle relates to how user interface components and navigation operate.

**Principle 4 - Robust:** This principle relates to making content robust enough to be interpreted by a wide variety of user agents, including assistive technologies.



## Accessibility Principle 4 – Robust

This principle relates to making content robust enough to be interpreted by a wide variety of user agents, including assistive technologies.

### Guideline 4.1 – Compatible

*Maximize compatibility with current and future user agents, including assistive technologies.*

#### 4.1.1 Parsing

*Level A - In content implemented using markup languages, elements have complete start and end tags, elements are nested according to their specifications, elements don't contain duplicate attributes, and any IDs are unique, except where the specifications allow these features.*

This is a core feature of Process Director; however, designers must also ensure that the use of **HTML** controls uses raw HTML that is similarly parsable within the DOM.

#### 4.1.2 Name, Role, Value

*Level A - For all user interface components (including but not limited to: form elements, links and components generated by scripts), the name and role can be programmatically determined; states, properties, and values that can be set by the user can be programmatically set; and notification of changes to these items is available to user agents, including assistive technologies.*

This is a core feature of Process Director.

Tables manually inserted into a Form should have the Role attribute set appropriately, as outlined in the [Online Form Designer](#) topic.

#### 4.1.3 Status Messages

*Level AA - In content implemented using markup languages, status messages can be programmatically determined through role or properties such that they can be presented to the user by assistive technologies without receiving focus.*

This is a core feature of Process Director. Designer-configured status messages, such as validation errors, are always included in the existing, text-based, messaging system, which never received focus when error or other status messages are displayed.

### Other Accessibility Principles

[Principle 1 - Perceivable](#): This principle relates to making information and user interface components presentable to users in ways they can perceive.

[Principle 2 - Operable](#): This principle relates to how user interface components and navigation operate.

[Principle 3 - Understandable](#): This principle relates to making information and the operation of the user interface understandable to users.

## Managing Content

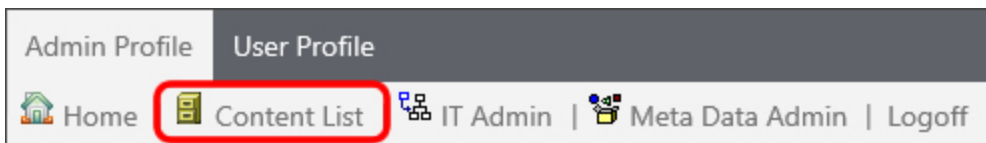
Process Director is a content management system that provides control, security and management of your digital data. This topic provides the essentials for managing objects in the database. All data and doc-

, which provides easier administration, higher security, distributed/remote database support, and a standard backup mechanism.

Process Director provides a database to securely store content objects. These objects include Folders, Documents, Files, Forms, etc. This chapter will provide an overview of folders, documents, and files. For information on Process Timelines, Forms, or Knowledge Views, refer to the respective chapters in this document.

All content is viewed through the use of Knowledge Views. A Knowledge View defines what data in the repository should be displayed. It is also used to view the [Content List](#).

To view content in the Process Director database, click on the [Content List](#) entry button in your profile.



## Content List #

The [Content List](#) is a hierarchical list of all Process Director objects. The Content List includes all of the objects created in Process Director, such as Forms, Process Timelines, Knowledge Views, etc. In addition, the Content List can include uploaded files such as Excel spreadsheets used to import data, Word documents used as templates for output documents, and any other files that become part of a process definition.

The [Content List](#) consists of a folder structure that is located on the left side of the screen, and a folder/file list that is displayed on the right side of the screen. Navigation through the Content List is done by clicking on a folder on either side of the screen.

Many users think of the [Content List](#) as a file system that users can create inside of Process Director, but this isn't really correct. The Content List organizes Process Director objects in a logical fashion for users who are browsing through the system, but the actual objects in Process Director aren't organized in the way that the Content List displays.

For instance, in a file system, the file name is the key element for identifying a file. Let's say you have a word document named "sample.docx", and you change the name to "NewSample.docx". If you go into Microsoft Word, and try to open the "sample.docx" file from the list of recent files, you'll receive a message telling you that the file can't be found. Changing the name of the file changes the identity of the file on your computer, making any references to the old file name invalid.



In Process Director, on the other hand, an object's name doesn't affect the identity of the object. Process Director tracks objects by a unique ID that you, as an end user will rarely, if ever, use. The object name is merely an attribute of the object that you can change at will without Process Director losing track of the object. Here's an example that demonstrates how object tracking works in Process Director.

---

<sup>1</sup>Process Director also provides file-based document storage as an alternative to database storage. Consult your system administrator regarding the setup at your site.





1. In this example, you have a Business Rule named "My Business Rule" and a Form named "My Form".

<input type="checkbox"/>	Name ▲
<input type="checkbox"/>	 My Business Rule
<input type="checkbox"/>	 My eForm

2. The Form references the Business Rule to determine when to enable the fields on the Form, as shown below.

Form Fields are Enabled by Default	<input type="checkbox"/>
Entire form is read-only when	<a href="#">(My Business Rule Is True)</a>
Options	

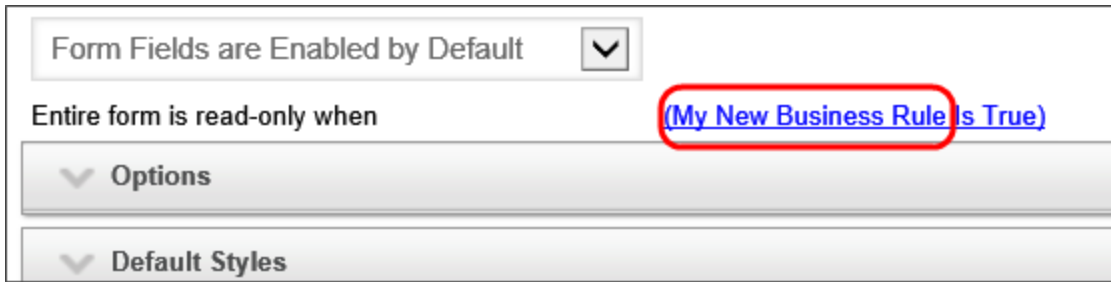
3. Change the name of the Business Rule to "My New Business Rule".

<input type="checkbox"/>	Name ▲
<input type="checkbox"/>	 My eForm
<input type="checkbox"/>	 My New Business Rule

4. Now, create a new Business Rule named "My Business Rule". This is the same name as the original Business Rule.

<input type="checkbox"/>	Name ▲
<input type="checkbox"/>	 My Business Rule
<input type="checkbox"/>	 My eForm
<input type="checkbox"/>	 My New Business Rule

5. Note that, even though you have a Business Rule with the name "My Business Rule", Process Director doesn't use it as the Business Rule in condition on the Form. Instead, Process Director has updated the Form definition to show the new name you gave to the original Business Rule.



Form Fields are Enabled by Default

Entire form is read-only when [\(My New Business Rule Is True\)](#)

Options

Default Styles

By the way, you can see the ID that Process Director uses to identify an object by opening the object definition. On a Form, for example, at the bottom of the Form definition screen, you'll see the property for the **Form URL**, written in a format similar to:

```
https://ServerName.com/form.aspx?pid=dc1c51ea-612e-48c0-8cc6-4ff9276cbfc9&formid=aa213c6f-a8aa-454f-a04d-30b56fd2e493
```

The "formid" URL parameter contains the ID that Process Director uses to uniquely identify the Form object, which, in this example, is

```
aa213c6f-a8aa-454f-a04d-30b56fd2e493
```

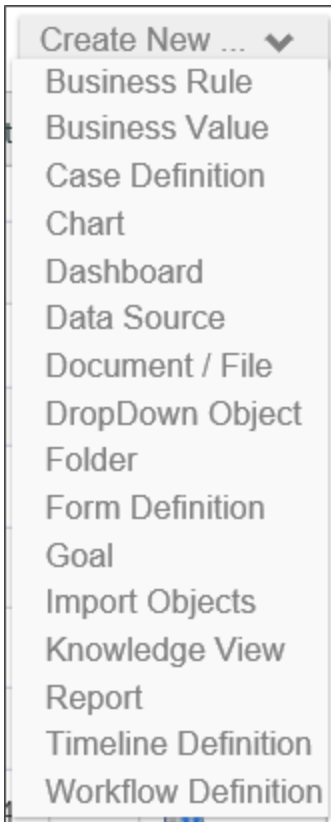
Changing the name of the Form, or any other property, doesn't change this ID. So, you can identify the Form with a name that is easily readable by humans, while Process Director identifies the Form with this ID, which is easily readable by computers.

## Creating Content List Objects

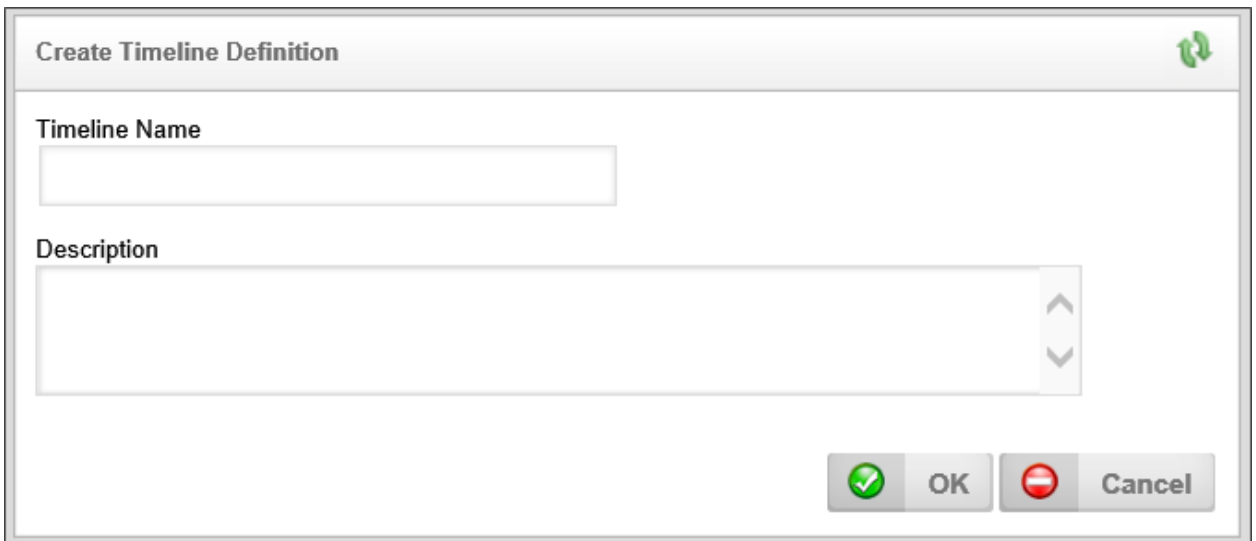
All objects are created either from the [Content List](#), and, for Process Director v6.0.300 and higher, from the [Home](#) page.

### *Creating from the Content List*








A **Create New...** dropdown menu is located at the top right each [Content List](#) page. This dropdown menu contains all of the process Director objects you can create. To create a new object, simply select the desired object from the dropdown menu.



When you select the desired object, Process Director will immediately display a **Create [Object]** screen, that will display the **Name** and **Description** properties for the object.



The **Name** property is mandatory, while the **Description** property is optional. Once you have given the object a **Name**, you can click the **OK** button to save and create the new object. If you have configured the **Description** property, the object's description will appear on the second line of the **Content List** where the object appears.

<input type="checkbox"/>	Name ▲
<input type="checkbox"/>	 <b>Advanced Form</b> Sample demo form for advanced design concepts.
<input type="checkbox"/>	 <b>Advanced Form TL</b> Timeline prompted by the Advanced Form.
<input type="checkbox"/>	 <b>Email Notification</b> Default email template for this application.
<input type="checkbox"/>	 <b>HR Requests</b> Consolidated KView for HR Requests
<input type="checkbox"/>	 <b>Paid Time Off Requests</b> Consolidated KView for HR Requests
<input type="checkbox"/>	 <b>Request Types</b> Dropdown to specify different HR action request types.
<input type="checkbox"/>	 <b>Review Types</b> Dropdown to specify different employee review types.

## Content List Object Types

The following [Content List](#) objects are available in Process Director.

CONTENT LIST OBJECT	DESCRIPTION
Business Rule	The definition for an object that encapsulates a key decision or value used in a business process.
Business Value	The Business Value is an object that provides data virtualization for any accessible external data.
Case Definition	The Case definition stores the properties/attributes for a Case Management application. The Properties can be shared across the multiple Forms and processes that might be part of a Case.
Chart	The Chart provides a graphical representation for any accessible data.
Dashboard	The Dashboard object provides a way to display many different Pro-

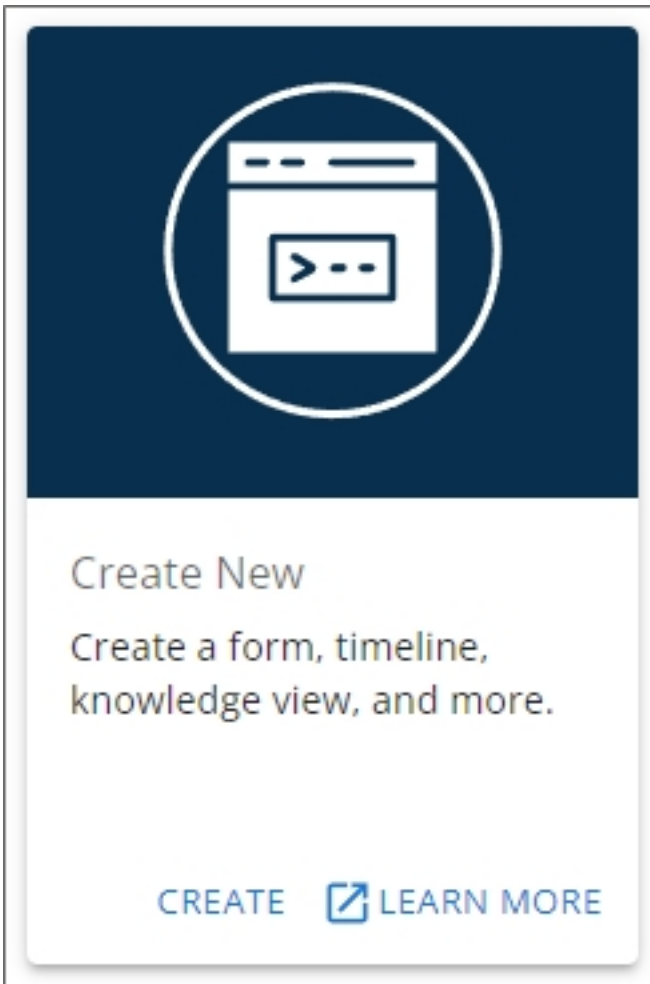
CONTENT LIST OBJECT	DESCRIPTION
	cess Director objects in a single interface.
Data Source	An object that accesses information found in a database.
Document/File	The <a href="#">Content List</a> placeholder for documents that you upload to process director for use as part of a process definition.
DropDown Object	The definition of labels and values that will be applied to a drop-down control on a Form.
Folder	The folder is the container for all other <a href="#">Content List</a> objects.
Form Definition	The definition of an electronic form. It stores the Form template, the fields that are created in the template, the user-defined events that are initiated while using the Form, and the validation rules for determining if the information provided in a Form is valid when submitted.
Goal	The Goal object creates a scheduled evaluation of conditions, and, based on the evaluation, sets a global system state and/or automatically starts a process.
Import Objects	Implements a process to import an XML file containing an exported set of Process Director objects.
Knowledge View	The definition of a view of the process and/or Form data, containing user-defined fields for display in a tabular format.
Machine Learning Definition	An object that enables you to use Process Director's Artificial Intelligence capabilities to review a dataset, and make predictions based on the state of that dataset.
Process Timeline	A process model that implements the patented BP Logix method of implementing a time-based method of modeling processes.
Report	The definition of a report object that uses the advanced reporting tool. The advanced reporting tool is available to users of cloud-based installations, and on-premise installations that explicitly include the Advanced Reporting module.
Stream Action	This object will read a recordset stream from a Datasource, and enable you to submit a form instance for each record in the stream.
Workflow Definition	A process model that implements the traditional, flowchart-based method of modeling processes.

 This model is the legacy process model for early versions of Process Director, and has been largely replaced by the patented Process Timeline object. New process features since

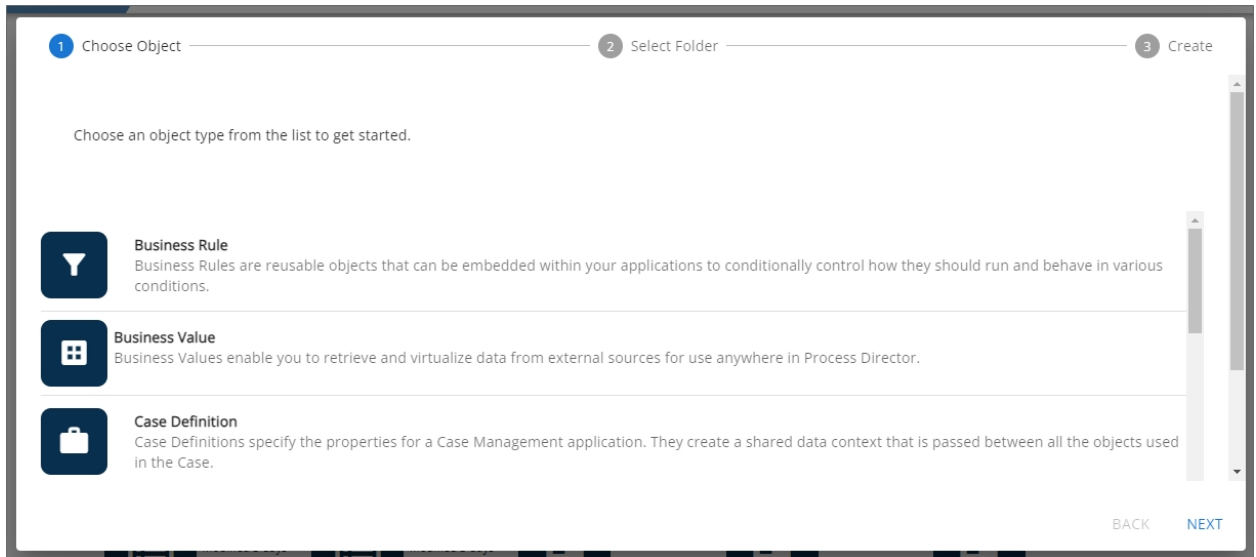
CONTENT LIST OBJECT	DESCRIPTION
	Process Director v4.5 have been added only to the Process Timeline.

### Creating From the Home Page

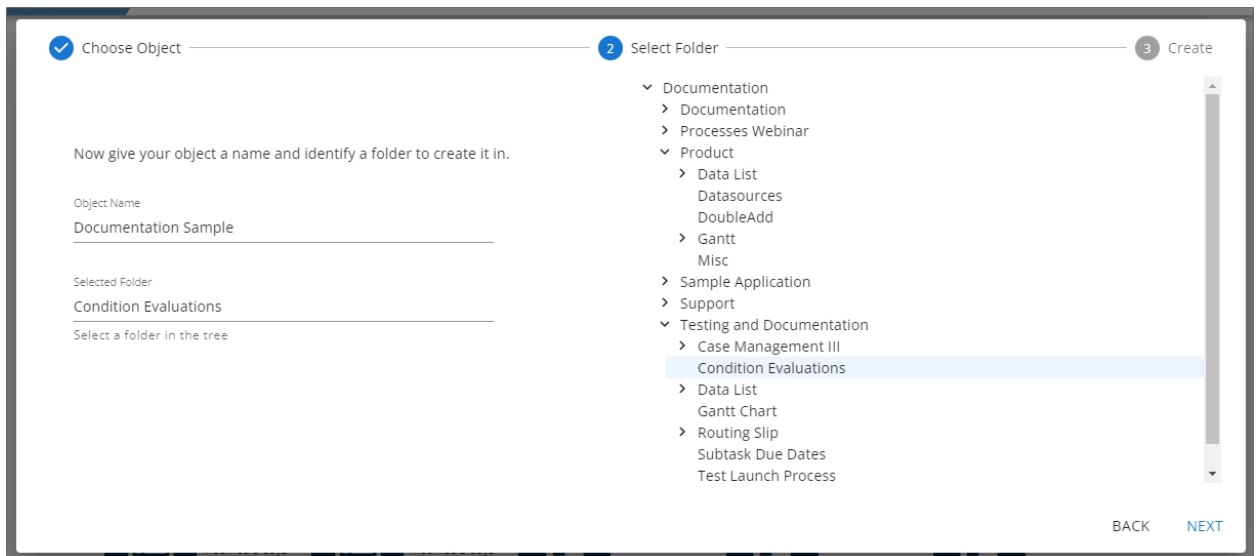
For Process Director v6.0.300 and higher, the [Home](#) page displays a panel labeled "Create New", which enables you to create a new [Content List](#) object. A [Learn More](#) link will, when clicked, open this documentation topic in a new window.



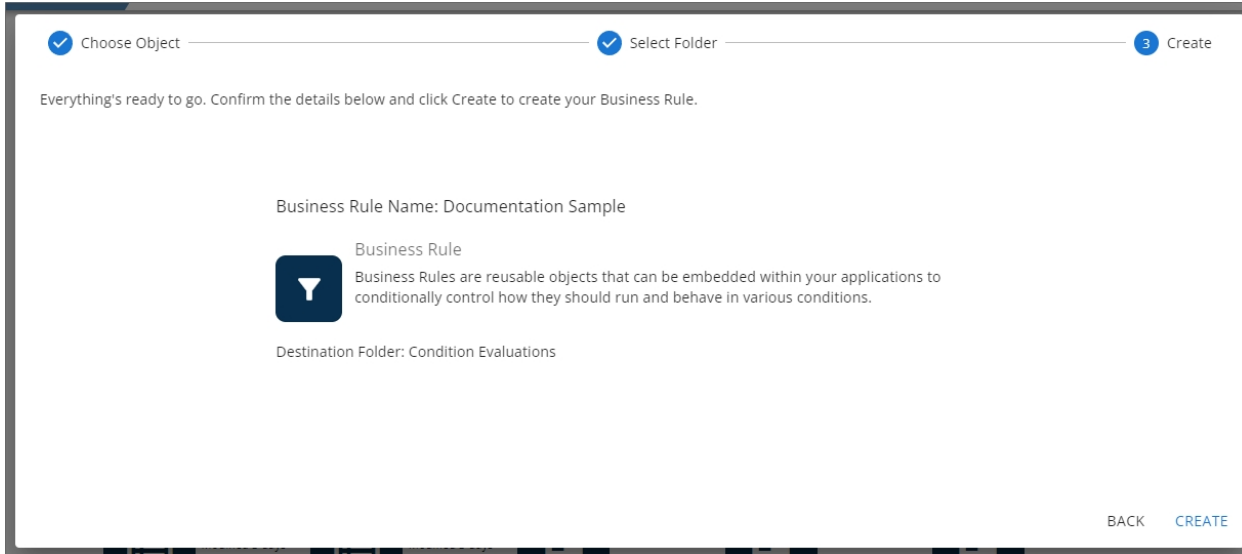
The [Create](#) link will, when clicked, open a wizard that will guide you through the process of creating the new object you desire.



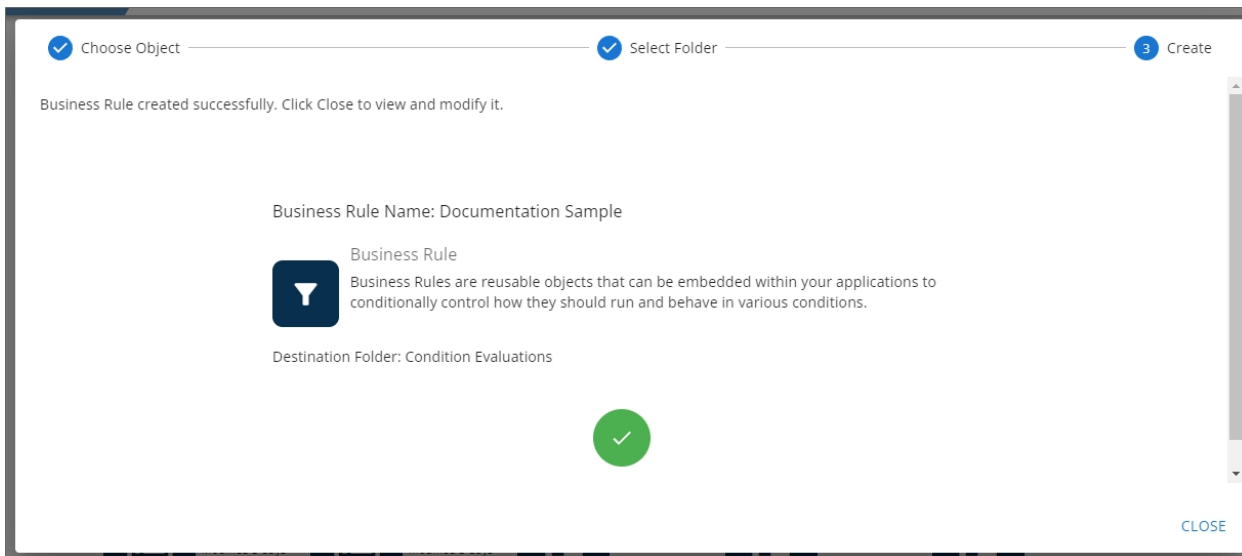
The first pane of the wizard is the **Choose Object** pane, which displays a list of all Process Director objects you might wish to create. Simply scroll to the object you wish to create, and click it to select it. Once you've done so, click the **Next** button at the bottom right corner of the wizard to advance to the **Select Folder** pane.



On the left side of the **Select Folder** pane, enter the name for the new object in the **Object Name** property. Once you've done so, you can navigate through the folder list displayed on the right side of the pane to find and select the folder into which you want to add the new object by clicking it (The folder must already exist in the **Content List**). When you do so, the folder name will appear automatically in the **Selected Folder** property, as shown above. When you're done, click the **Next** button again.

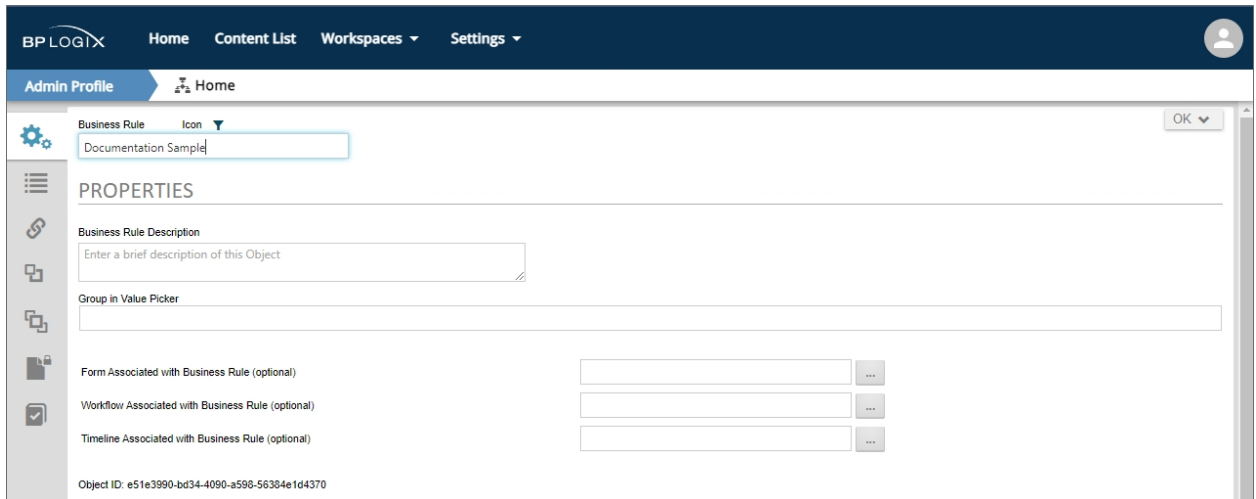


The **Create** pane enables you to review the choices you've made in the previous two wizard panes. You can, at any time, click the **Back** button to return to a previous pane to edit your choices. Once you finished reviewing your choices, you can click the **Create** button to create the new object. When you click the create button, a "working" animated icon will briefly appear, followed by a success icon when the object has been created.



Click the **Close** button to exit the wizard. Process Director will automatically navigate you to the new object definition and open it for editing, so that you can complete its configuration.





## Creating a New Application #

Most Process Director implementations consist of standalone applications. Each application should reside in a single [Content List](#) folder that contains all of the objects that you create as part of the application. Placing your application inside a single folder makes importing and exporting your application between the development and production environment much easier.

As a minimum, an working application requires at least two objects, a Form and Process Timeline that, when linked, provide the data used by the application (the Form), and the process the application will use to perform the activities required to operate (the Process Timeline).

The Form and Process Timeline must be associated with each other to ensure that 1) the Form, when submitted, starts the Process Timeline properly, and 2) the Process Timeline can reference or evaluate any required Form field values correctly. Together, the linked Form and Process Timeline provide the core of your application, or the "application stub". The application may require the use of other objects, such as Business Rules, Knowledge Views, etc., all of which might reside in the application folder, but the Form and Process Timeline application stub will always be the central operating elements of an application.

As explained in the [Folder Structure section of the Importing/Exporting Content topic](#), more complex applications might implement more complex methods of organization, but the general rule of thumb is that most application objects should be included in the application's single parent folder.

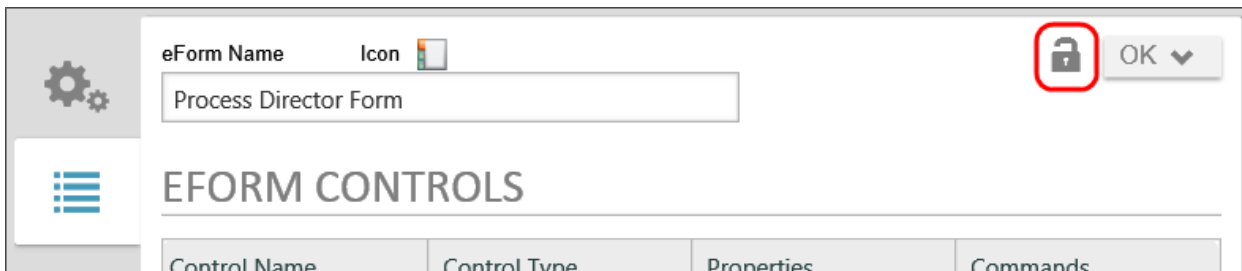
## Object Locking

Process Director v3.49 and above implements a new object locking system for object definitions (e.g., Business Rules, Knowledge Views, or Process Timeline definitions) to ensure that multiple users can't edit the same object definition at the same time, and thus overwrite each others' changes. Process Director implements object locking by default, but object locking can be turned off by setting the [ObjectLockingEnable](#) custom variable to "false" in the Custom Vars file.

When object locking is enabled, users who edit an object definition can click the object's **Lock Object** button to ensure that any other user who opens the definition will see the object only in read only mode. The locking user will be the only user with editing capability while the object is locked.



Once an object is locked, the lock will remain in effect until the locking user manually clicks the object's **Release Lock** button to release the lock. Even if the locking user saves or closes the form, or logs out of Process Director, the object will remain locked until the locking user manually releases the lock.



Object locking can also be required by setting the [ObjectLockingForce](#) custom variable to "true" in the Custom Vars file.



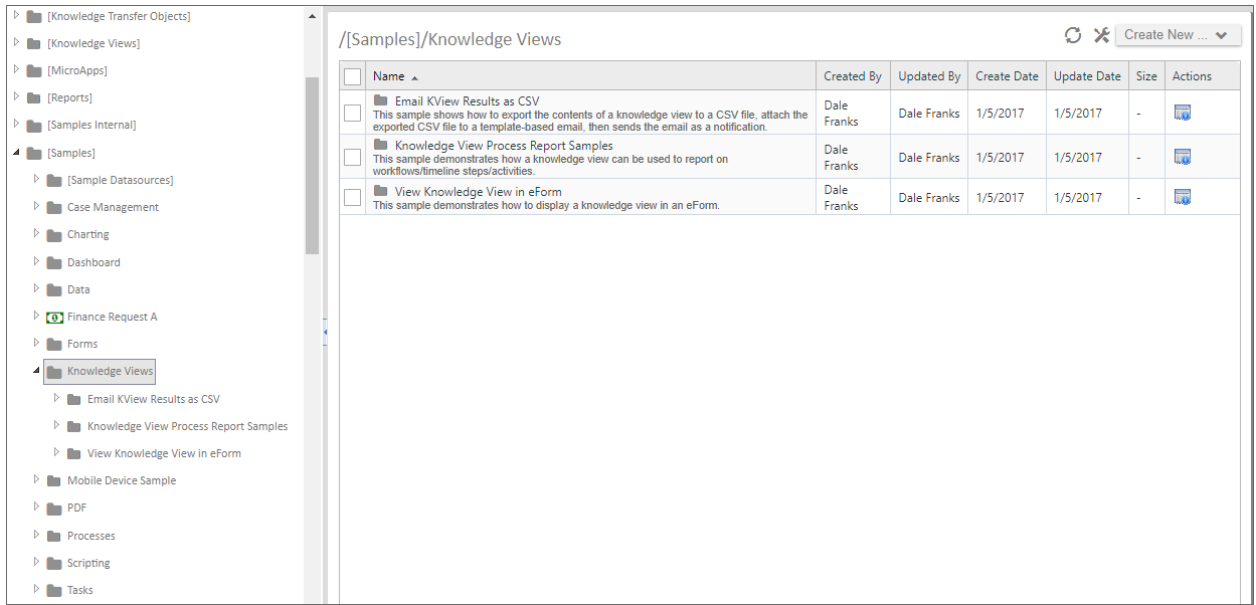
BP Logix recommends requiring object locking.

When object locking is required, object definitions are opened in "read-only" mode for all users. Any user who wishes to edit the object definition can only do so by checking out the object. When the locking user clicks the object's **Lock Object button**, the definition will display in edit mode for the locking user, and remain in read-only mode for all other users. Again, the lock will remain in effect until the locking user manually clicks the object's **Release Lock button** to release the lock. Once the locking user clicks the **Release Lock button**, the object definition, when opened, will display in read-only mode for all users.

If you lock or release a lock on a folder, the operation will be applied to all of the contents of the folder, assuming that you have the right to lock or unlock the child objects. Process Director will silently skip objects the current user doesn't have the right to unlock. Locking, similarly, will preserve any existing locks in the folder that are held by someone other than the current user (that is, they'll still be held by the original lock holder). Folder locking will lock all objects recursively in the folder tree, not just the direct children of the folder. The button text for the lock/unlock button will reflect this difference to locking applied to folders.

## Organizing Content List Objects

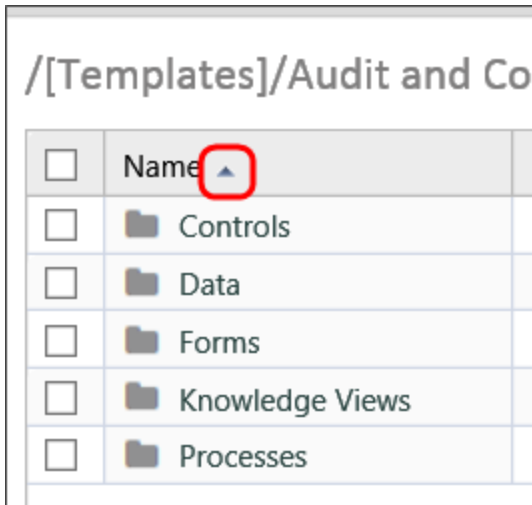
In general, [Content List](#) objects are sorted into folders, much like files in your computer's file system. There is no hard and fast method of organizing the folders, except that folders and subfolders should be organized logically. For instance, one possible method of organizing folders is shown below.



In this particular organization scheme, you can see that the [\[Samples\]](#) folder on the left side of the screen contains subfolders for different Process Director Objects. In the [Knowledge Views](#) folder, you can see the folders are further subdivided into three sample application folders. The [Knowledge Views](#) folder is selected, so the [Content List](#) on the right side of the example shows all three of the objects in the selected folder. These three folders—though not shown on this example—might further be subdivided into folders named [Forms](#), [Processes](#), [Knowledge Views](#), etc., into which the appropriate Process Director objects would be placed.

You do not, of course, have to copy this particular method of organization, but you should, as a best practice, define some standard method of organizing your [Content List](#) so that all implementers are familiar with how an application should be organized. Another general rule of organization—and implementation in general—is that you should use the fewest number of objects necessary to model and automate your processes.

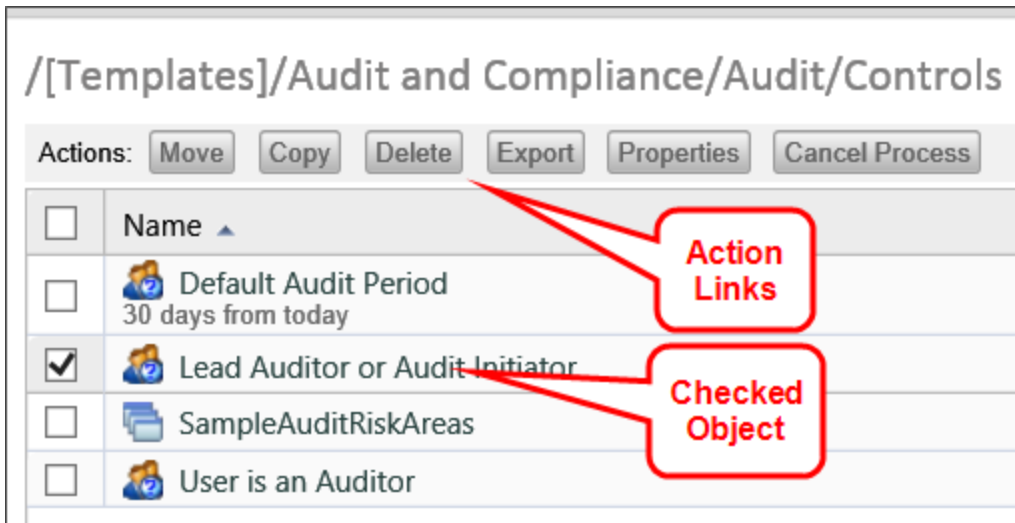
The [Content List](#) of the right side of the screen is sortable by column. To sort the items displayed in the Content List, click on the column name. To change the sort from ascending to descending, click on the column name a second time. A small arrow icon will appear next to the column name to show whether you are sorting the column in ascending or descending order.



Each time you click on the column header, the sorting function will occur automatically.

## Managing Objects #

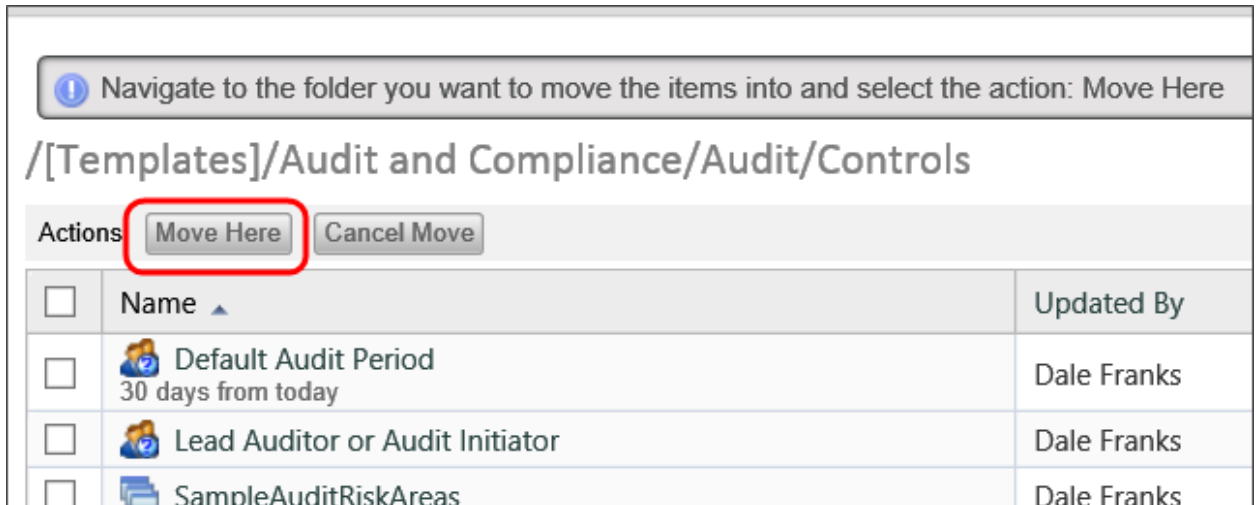
On the left side of each object in the [Content List](#) is a check box that can be checked to select one or more objects. When you check one or more of these check boxes, a series of Action Links will appear at the top of the Content List that enable you to copy, move, etc. the checked objects.



The following action links are available.

### Move

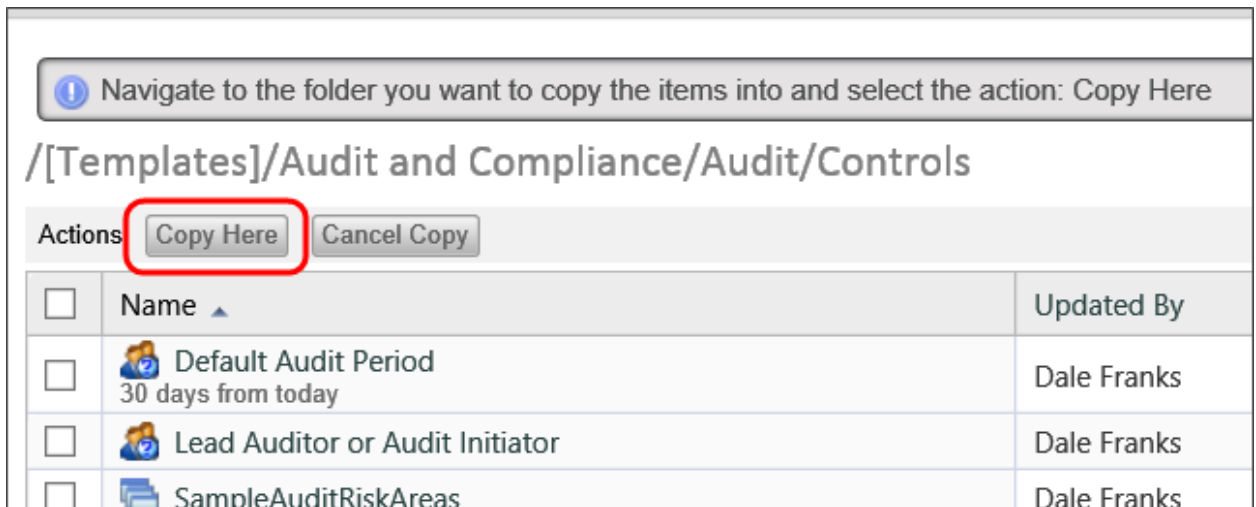
Clicking the **Move** button will enable you to move the object, along with its complete configuration, to another location in the [Content List](#). When you click the **Move** button, a new set of action buttons will appear.



Simply navigate to the folder into which you'd like to place the object and click the **Move Here** button to place the moved object into the desired location. Process Director will move the object from its original location to the new location you have selected. You can cancel the move at any time by clicking the **Cancel Move** button.

### Copy

Clicking the **Copy** button will copy the basic object definition but *not* any of the configured Custom Task event mappings. Just as with the **Move** button, clicking the **Copy** button will cause process Director to present you with a new set of action buttons.



Simply navigate to the folder into which you'd like to place the object and click the **Copy Here** button to place the copied object into the desired location. Process Director will copy the object from its original location and place the copy in the new location you have selected, while leaving the original copy in its place. You can cancel the copy at any time by clicking the **Cancel Copy** button.

### Delete

When you click the **Delete** button, a confirmation dialog box will be displayed to enable you to confirm that you wish to delete the object. Clicking the **OK** button on the confirmation dialog box will immediately delete the object.



When you delete a definition object, like a Form or Process Timeline definition, all of the instances of that object will also be completely removed from the system. There is no undo action for deletions!

## Export

Clicking the **Export** button will start the process of exporting the content object. If you have selected a folder to export, all of the folder's contents will be exported, and the folder/subfolder hierarchy will be exported exactly as it appears in the [Content List](#). See the [Importing/Exporting Objects](#) topic for more information about this process.

## Properties

Clicking this button will open the properties screen for the selected object. This option will only appear if a single item is selected in the [Content List](#).

## Run

Clicking this button will run the object, i.e., will open a Form for editing, or start a process. This option will only appear if a single item is selected in the [Content List](#), and will start a specific action, depending on the object type, as shown below.

OBJECT TYPE	ACTION
Form Definition	Open a new instance of that form.
Process Definition	Launch a new instance of that process
Knowledge View, Dashboard, etc.	Opens the object in display mode.

## Cancel Process

Clicking this button will cancel a running process.

## Creating Folders #

Folders provide a hierarchical structure to content in the database. The system supports folders and sub-folders. Each folder can contain any number of objects or sub-folders. A folder can have any name and may contain an optional description. Folders provide the organizational foundation for your data. They can be used to group related items according to Process Timelines, or structured according to user roles.

To create a new folder, use the **Create New** dropdown in the [Content List](#) and choose the **Folder** menu item. A user must have Modify permission to the parent folder to create a folder. If a user doesn't have the appropriate permission, the dropdown won't appear.

Documentation						
Name ▲	Updated By	Create Date	Update Date	Size	Actions	
Namespace	Dale Franks	10/30/2015	10/30/2015	-		
Reports	Dale Franks	12/5/2014	2/3/2015	-		


## Managing Documents #

Process Director securely stores your documents and files. As an electronic document management system (EDMS), it can store any type of document or file in their native formats. Any file type can be uploaded and managed by the server, including file types such as .PDF, .TXT, .GIF, .JPG, .DOC, .DOCX, .XLS, .XLSX, .PPT, .PPTX, .BMP, .PNG, .ICO, .EPS, .AI, and .TIFF to name a few. Process Director can convert documents or files to a PDF or HTML format; while still maintaining the native file format in the database. Process Director doesn't provide the document authoring tools; instead it utilizes the native authoring tools you are currently using to create and view your files. The type of document can be viewed by clicking on the item in the [Content List](#). The native viewer for the specific document type will be used.

## Uploading Documents #

Documents are added to Process Director database by uploading them from your computer. To add a new document, navigate to the appropriate folder on Process Director where the document should be added. Select the **Create New** dropdown and choose the **Document/File** entry. A user must have Modify permission to the parent folder to add the new document. If a user doesn't have the appropriate permission, the **Create New** dropdown item won't appear.

## Uploading With the BP Logix Plug-in [Legacy Information]

 The BP Logix Plugin uses Microsoft's ActiveX technology with Internet Explorer. ActiveX technology has been deprecated by Microsoft, and Microsoft is phasing out support for Internet Explorer, with Windows 10 targeting 15 June, 2022 as the phase-out date for that operating system. Additionally, ActiveX can't be used with any 64-bit version of Microsoft Office, or recent versions of Office 365. Information about the BP Logix plug-in, therefore, is largely provided for operators of legacy systems that don't use Microsoft products released after 2019, or any 64-bit product.

The BP Logix Plug-in makes the process of uploading and/or creating documents easier. When uploading a document with the BP Logix Plug-in installed, a button labeled **Check Out and Edit** will display which will open the editor associated with that file.

## EDIT

This eForm was created using the BP Logix Plugin inside MS Word.

To edit the eForm definition, you must first check it out. This will prevent other users from editing the eForm definition. Users **will** be able to Run and Open eForm instances while editing the definition.



Check Out and Edit

This action will check out the eForm, then automatically download and open the eForm on your PC.



Check Out

This action will check out the eForm. You can then download the file to any location on your PC.

### Uploading Without the Plug-in

The BP Logix Plug-in isn't required to upload documents. You'll notice on the following screen that the **Check Out and Edit** button isn't displayed. This will happen when adding a new document without the BP Logix plug-in.

## EDIT

To edit the document, you must first check it out. This will prevent other users from editing the document.



Check Out

This action will check out the file. You can then download the file to any location on your PC.

This browser does not support the BP Logix plugin. The plugin will enable one-click editing within 32-bit Internet Explorer.

### Searching For Content #

Process Director supports the ability for users to search through the entire database for any object type using a Knowledge View. The list of objects returned that match the search request will be limited to only those for which the user has View permission.

Searches may be case-sensitive. This is determined by the back-end database system you use. If you are using SQL Server, the default is case-insensitive. If you are using Oracle, the default is case-sensitive.

There is an optional search feature called FTS (full-text search). This feature enables users to search the actual contents of a document for the existence of a word. This support requires SQL Server with FTS enabled. For information on how to enable the full-text indexing on SQL Server, please refer to the [Full-Text Search topic](#) in the Administrator's Guide.

### Controlling Who Can Create Content #

Process Director enables you to control a user's permission to create content in a folder. Administrators can use permissions to determine what type of objects or content can be created by users. You may only want certain types of users to be able to create objects like Process Timeline and Form definitions. See the [Permissions](#) topic for detailed information on how to set permissions.

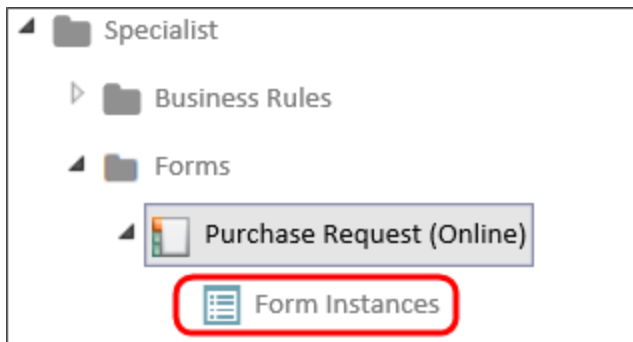


## Object Instances

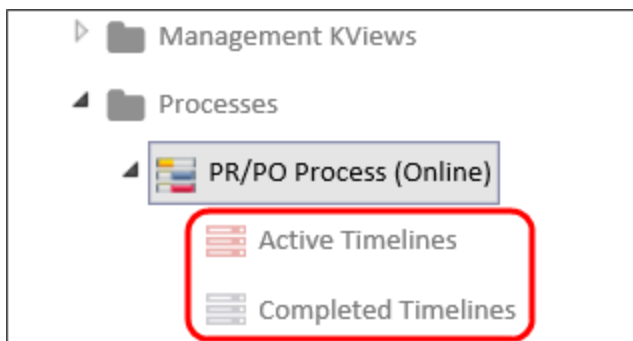
So, far, we've talked about [Content List](#) objects in terms of object definitions, but now we need to discuss object instances. The purpose of an object definition is to provide a template for how a form looks, and the data it will contain, or what order of activities a Process Timeline will follow. When a user opens a form for submission, or when a process starts, Process Director creates a copy of the object that will be used for that specific form submission, or that specific iteration of the process. Each of these copies are called an **instance** of the form or process.

Let's look at a form, for example. The form definition defines what fields will appear on the form, and how the form will look. In other words, the definition specifies the basic structure of all of the form that users will submit. Each individual instance of that form contains all the properties specified for the form, and will also contain all of the data entered into that form for this specific submission. So, each instance of the form contains the data for that specific form submission. Similarly, each instance of a process Timeline contains the basic structure specified in the definition, as well as all of the task start dates, due dates, etc. that occur during that specific run of the process.

All Process Director objects that support instances will display those instances in the [Content List](#), as child objects of the object definition. For instance, a Form definition will have a node that appears in the Content List when you open the definition, named Form Instances.



Similarly, A Process Timeline will have two nodes that appear in the [Content List](#), [Active Timelines](#) (which show all of the Timeline instances that are currently running), and [Completed Timelines](#) (which show all of the Timeline instances that have finished).



Object instances have their own properties, which are different than the properties of the object definition. Additionally, each type of instance has a set of properties unique to that instance type, e.g., the

properties of a form instance are different than the properties of a Timeline instance. We'll describe the Properties of the Form and Timeline instance below.

The Object Instances page will display a list of all applicable instances in a Knowledge View. Where appropriate, the Knowledge View will contain a Name search filter and **Search** button to narrow the list of instances to those whose names match the search filter text.







## Form Instances #

When you click on the [Form Instances](#) node of the [Content List](#), a list of Form instances will appear.

### Form Instances

Actions: [Fill Out This Form](#) [Form Properties](#)

Name Contains  [Search](#)







<input type="checkbox"/>	Name	Size
<input type="checkbox"/>	 Purchase Request (Online) Submitted On 9/15/2017 11:21 AM	54 fields
<input type="checkbox"/>	 Purchase Request (Online) Submitted On 8/11/2017 1:33 PM	54 fields
<input type="checkbox"/>	 Purchase Request (Online) Submitted On 6/21/2017 12:54 PM	60 fields
<input type="checkbox"/>	 Purchase Request (Online) Submitted On 5/5/2017 4:29 PM	54 fields
<input type="checkbox"/>	 Purchase Request (Online) Submitted On 1/17/2017 11:26 AM	78 fields
<input type="checkbox"/>	 Purchase Request (Online) Submitted On 1/17/2017 7:59 AM	54 fields

You can use the action buttons above the list of instances to open the form to fill out data by clicking the **Fill Out This Form Button**, and open the Form definition by clicking the **Form Properties** button. When you select a check box next to one of the form instances, the action buttons will change.

### Form Instances

Actions: [Delete](#) [Export](#) [Properties](#)

Name Contains  [Search](#)

<input type="checkbox"/>	Name	Size
<input checked="" type="checkbox"/>	 Purchase Request (Online) Submitted On 9/15/2017 11:21 AM	54 fields
<input type="checkbox"/>	 Purchase Request (Online) Submitted On 8/11/2017 1:33 PM	54 fields
<input type="checkbox"/>	 Purchase Request (Online) Submitted On 6/21/2017 12:54 PM	60 fields
<input type="checkbox"/>	 Purchase Request (Online) Submitted On 5/5/2017 4:29 PM	54 fields
<input type="checkbox"/>	 Purchase Request (Online) Submitted On 1/17/2017 11:26 AM	78 fields
<input type="checkbox"/>	 Purchase Request (Online) Submitted On 1/17/2017 7:59 AM	54 fields

You can delete the instance by clicking on the **Delete** button. You can export the Form instance by clicking on the **Export** button. Exporting form instances is rarely done, and the use case for doing so is extremely limited. It's not a feature that you'd usually use, but there are some cases where a form can be

used to provide configuration data for a process, and you may want to export the instance containing the configuration data from the Development environment to Production. Other than that, though, BP Logix doesn't recommend exporting Form instances.

Finally, you can view the properties of the Form instance by clicking the **Properties** button. When you open the properties screen for the Form instance, you'll see the tabbed interface common to Process Director objects. The following tabs are available for the Form instance.

### Properties Tab

Form Instance Name Icon

Auto Claim # 18-000012

## PROPERTIES

**Description**

Enter a brief description of this Object

**Form Properties**

Submitted By: Dale Franks  
 Last Updated By: Dale Franks  
 Instance Created: 10/16/2018 11:48 AM  
 Instance Updated: 1/24/2019 12:26 PM  
 Version: 5  
 Object ID: c1ce07d2-7b4f-4d25-a2a0-b56fcd9d0134  
 Form URL: http://localhost/form.aspx?forminstid=c1ce07d2-7b4f-4d25-a2a0-b56fcd9d0134  
 Case Instance Name: Claim 18-000012 for Policy #A458632542 Submitted on 10/16/2018 5:00 AM  
 Case Instance ID: [e1192164-37e7-4fe0-ba1e-a8545dc10f4d](#) Remove Object and Attachments From Case  
 Currently Editing:

View Form View Form Definition

Control Name	Control Type
Address	TextBox
City	TextBox
ClaimCause	Radio
ClaimNarrative	TextBox
ClaimNumber	TextBox

The **Properties** tab displays the Form Instance name and Description, along with a **Form Properties** section that has some basic information about the form instance, when it was created, it's ID, etc. If the instance is associated with a Case definition, the object and all of its children can be removed from the Case instance by clicking the **Remove Object and Attachments from Case** button. Below that is a **View Form** button to enable you to view the instance in it's current state, and a **View Form Definition** button to open the form definition.

Below that is a list of all the fields on the form, and the current values stored in each field.

### Show Parents Tab

The **Show Parents** tab displays all of the parent process objects, such as running Process Timelines. You can check one of the processes, then click the **Properties** button to open the **Properties** screen of the process instance.

### Audit Viewer Tab

If the Form definition is configured to audit field changes, the **Audit Viewer** tab will display all of the field changes that have been made to the form fields, showing the old value, the new value, and the change date and time.

Form Instance Name    Icon

Purchase Request (Online) Submitted On 9/15/2017 1

**AUDIT VIEWER**

Instance Created: 9/15/2017 11:21 AM    Instance Updated: 12/5/2017 4:51 PM    Version 6

Control Name	Previous Value	New Value	Updated
ExistingVendor	Colgate-Palmolive Company	Birds Eye Co. Inc.	12/5/2017 4:51 PM
ShipVia	UPS Ground	USPS Priority Mail	12/5/2017 4:51 PM
SubmitterPhone		760-555-1234	12/5/2017 4:51 PM
Terms	Net 30	Net 45	12/5/2017 4:51 PM
VendorAddress1	909 River Rd	1400 Hague Road	12/5/2017 4:51 PM
VendorAddress2		Suite 185	12/5/2017 4:51 PM
VendorCity	Piscataway	Indianapolis	12/5/2017 4:51 PM
VendorContact	Ross Cannon	Carmen Esposito	12/5/2017 4:51 PM
VendorName	Colgate-Palmolive Company	Birds Eye Co. Inc.	12/5/2017 4:51 PM
VendorState	NJ	IN	12/5/2017 4:51 PM
VendorZip	08854-5503	46250-0414	12/5/2017 4:51 PM

The remaining tabs of the Form instance are the common tabs used for all objects, such as **Meta Data**, **Permissions**, etc.




### Timeline Instance #

As mentioned previously, in the **Content list**, you can view either the running Timeline instances by clicking the **Active Timelines** node on the treeview, or you can view Timelines that have finished by clicking the **Completed Timelines** node. In the example below, you can see a list of completed Timeline instances.

**Completed Timeline Instances**

Actions: [Run This Timeline](#) [Timeline Properties](#)

Name Contains  [Search](#)

<input type="checkbox"/>	Name
<input type="checkbox"/>	 PR/PO Process (Online) (Purchase Request (Online) Submitted On 9/15/2017 11:21 AM, Question for PR #PR17-0019 from Dale Franks to Dale Franks, Question for PR #PR17-0019 from Dale Franks to Dale Franks, Question for PR #PR17-0019 from Dale Franks to Dale F
<input type="checkbox"/>	 PR/PO Process (Online) (Purchase Order #PO17-0008, created on 1/17/2017 11:26 AM, Purchase Order #PO17-0008, created on 1/17/2017 11:26 AM.pdf, Purchase Request (Online) Submitted On 1/17/2017 11:26 AM)
<input type="checkbox"/>	 PR/PO Process (Online) (Purchase Request (Online) Submitted On 1/17/2017 7:59 AM)




Just as with the Form instances, you can run a new instance of the Timeline by clicking the **Run This Timeline** button, or you can open the Timeline definition by clicking the **Timeline Properties** button.

If you select a Timeline instance by checking the box next to the instance name, you see a **Delete** button to delete the instance, and a **Properties** button to open the instance properties. A running Process Timeline will have an additional **Cancel** button that, when clicked, will cancel the running Timeline.

**Completed Timeline Instances**


Actions: [Delete](#) [Properties](#)

Name Contains  [Search](#)

<input type="checkbox"/>	Name
<input type="checkbox"/>	 PR/PO Process (Online) (Purchase Request (Online) Submitted On 9/15/2017 11:21 AM, Question for PR #PR17-0019 from Dale Franks to Dale Franks, Question for PR #PR17-0019 from Dale Franks to Dale Franks, Question for PR #PR17-0019 from Dale Franks to Dale F
<input checked="" type="checkbox"/>	 PR/PO Process (Online) (Purchase Order #PO17-0008, created on 1/17/2017 11:26 AM, Purchase Order #PO17-0008, created on 1/17/2017 11:26 AM.pdf, Purchase Request (Online) Submitted On 1/17/2017 11:26 AM)
<input type="checkbox"/>	 PR/PO Process (Online) (Purchase Request (Online) Submitted On 1/17/2017 7:59 AM)

Opening the properties of the Timeline instance will display the regular tabbed interface for the instance properties.

## Timeline Status Tab

Timeline Instance Name  Auto Claim (Auto Claim # 18-000012)

### TIMELINE STATUS

Description  
Enter a brief description of this Object

[View Timeline Form](#)


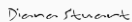
Timeline

Timeline Status Active Timeline Started 10/16/2018 Timeline Ended -

Case Instance Name Claim 18-000012 for Policy #A458632542 Submitted on 10/16/2018 5:00 AM Case Instance ID [e1192164-37e7-4fe0-ba1e-a8545dc10f4d](#) [Remove Object and Attachments From Case](#)

Routing Slip Options

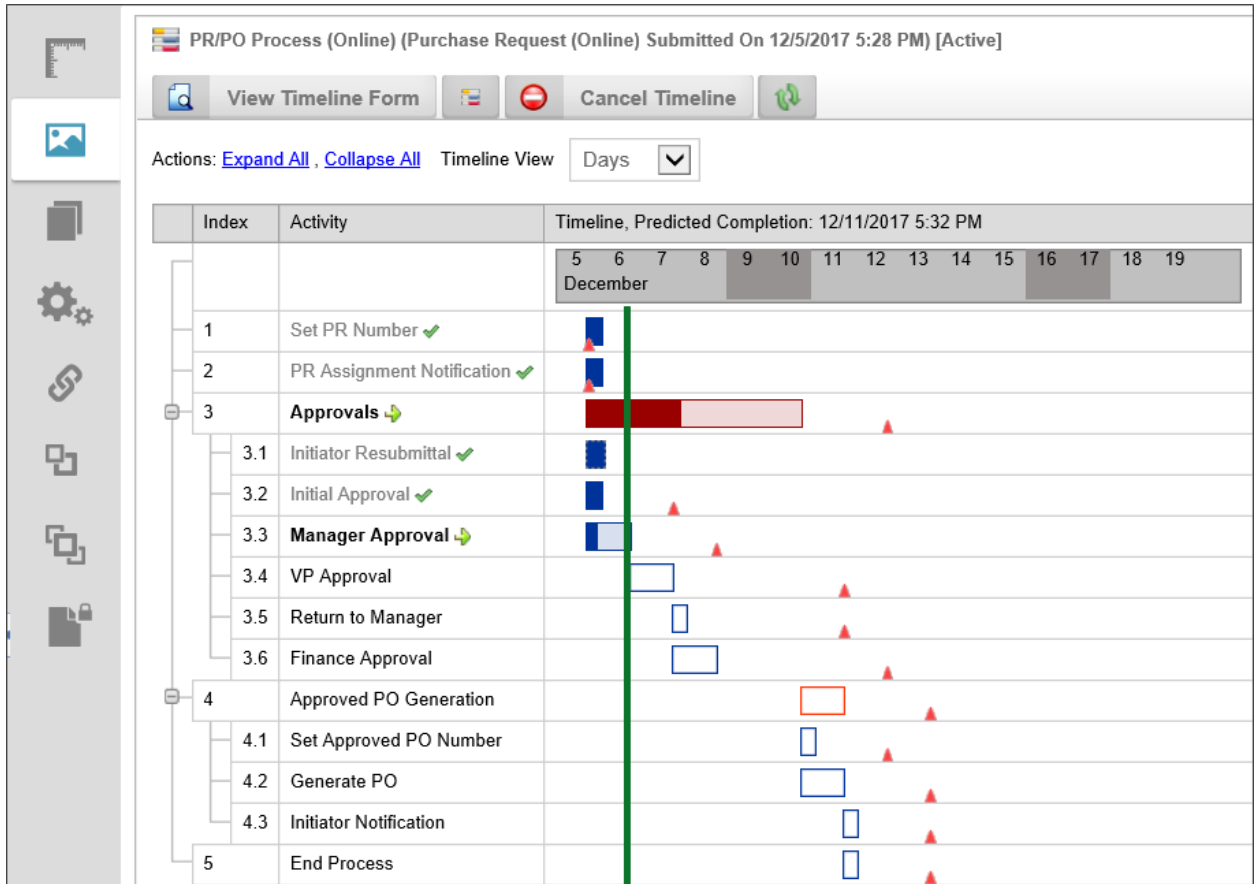
Routing Slip

Participants	Signature	Completed	Status
<b>Initiator</b>			
Dale Franks		10/16/2018	Completed
<b>Claim Review</b> 10/16/2018 11:48 AM			
Diana Stuart Impersonated By: Dale Franks		1/22/2019	Completed

You can view the Form associated with the Timeline instance by clicking on the [View Timeline Form](#) button. If the instance is associated with a Case definition, the object and all of its children can be removed from the Case instance by clicking the [Remove Object and Attachments from Case](#) button.

The remainder of this tab displays the status of the Timeline instance, along with a [Routing Slip](#) that shows all of the activities that have occurred during the lifetime of the instance.

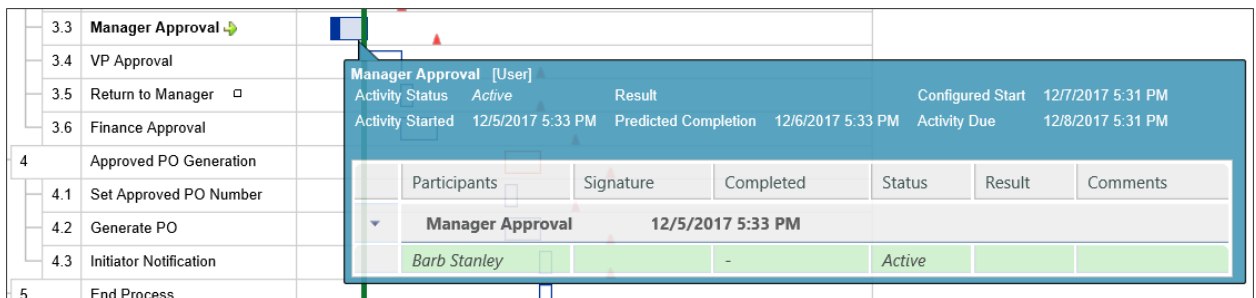
## Administration Tab



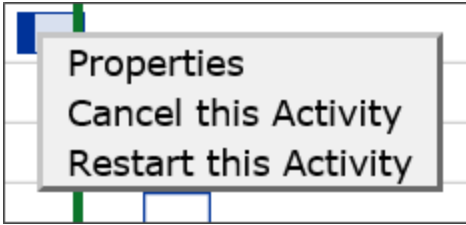
At the top of the **Administration** tab, there is a series of buttons.

- The **View Timeline Form** button will open the associated form.
- The **View Timeline Definition** button will open the parent Timeline definition.
- The **Cancel Timeline** button will cancel the running Process Timeline.
- The **Refresh** button will reload the Gantt Chart for the Process Timeline to show its current status.

Each of the Timeline Activities will be displayed in a Gantt chart. Completed activities will be displayed with a green check mark next to the Activity name, while the currently running activities will show a green arrow next to the Activity name. Due dates for each of the activities will be shown by red triangles, which, when moused over, will display the due date information. Each of the bars denoting the Timeline activities will also, when moused over, show the relevant information for the Activity.



Right-clicking one of the Activity bars will open a pop-up menu that will enable you to view the properties of the Activity, cancel it, or restart it.



The remainder of the tabs for the Timeline instance are the common tabs available to all objects.

For more information about managing Timeline instances, please see the [Managing Process Timelines](#) topic.

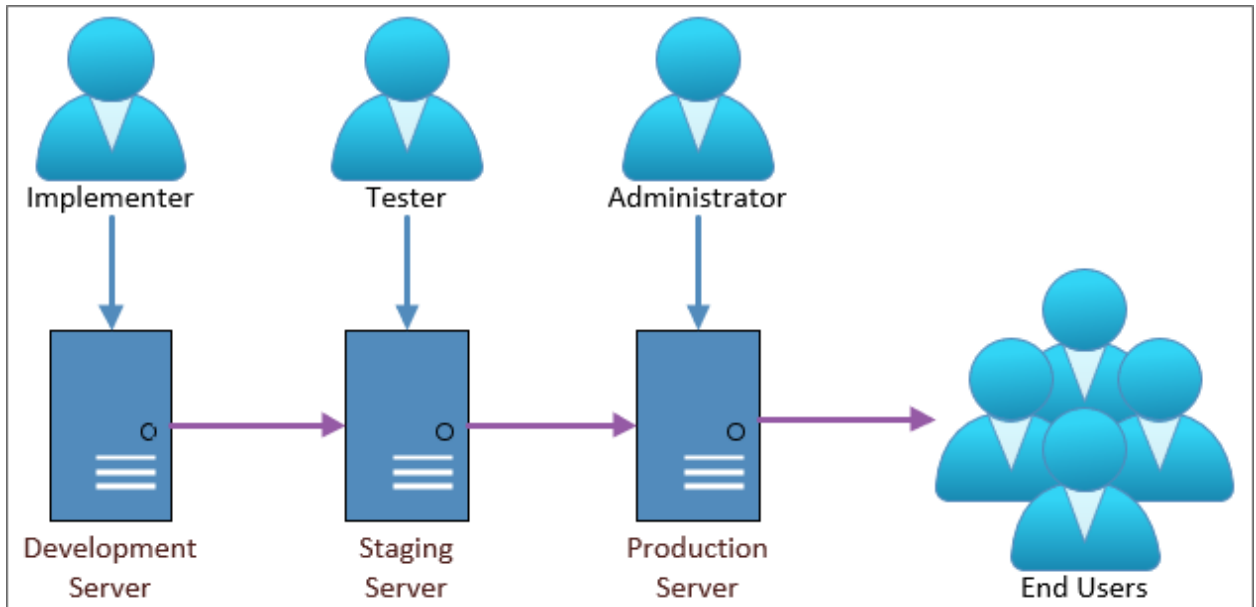
## Importing/Exporting Content

When importing and exporting content, there are a number of vital considerations that need to be kept in mind. While import/export is a routine procedure, it must be done correctly to avoid serious problems. As such, it is important to discuss some best practices—not only in the import and export process itself, but in some basic practices for configuring Process Director objects.

In an ideal environment, there would be three different Process Director installations:

- A **Development** system that is relatively open, and can be used for testing and development in a fairly unrestricted atmosphere.
- A **Staging** system that exactly mirrors the configuration, users, permissions, and [Content List](#) structure of the production server. This server should replicate the production environment so that QA testing can be completed in a copy of the production environment before an implementation is moved to production. Fresh replications of the Production system should be made on the Staging server before any pre-release testing is performed.
- A **Production** system where all of the real-world operations are conducted.





## Content List Folders #

In Process Director, Content List Folders have two purposes.

First, they make it easier to establish permissions that allow different classes of users to access different [Content List](#) objects. The best practice is to set permissions only on folders, and not on individual Forms, Knowledge Views, Business Rules, etc. Permissions are inherited from parent objects, and when new child objects are created, they automatically have the same set of permissions as the folder in which they are contained. (If you change permissions on the folder, however, the permissions on objects within the folder won't be changed unless you click one of the ["Replicate Current Permissions..."](#) buttons.)

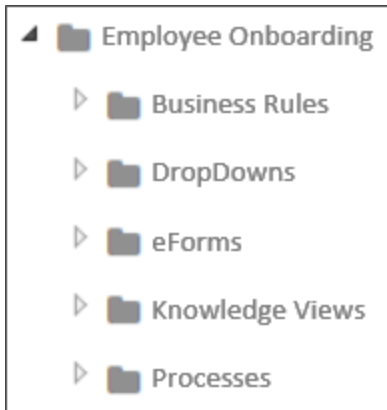
Different folders might have different permissions to ensure that administrators, process managers, and users are only granted access to objects appropriate to their roles. For example, you might have certain Knowledge Views that only managers can access, while other Knowledge Views can be accessed by any user. In this case, you could create a [Manager KViews](#) and a [User KViews](#) folder. By setting the permissions of the [Manager KViews](#) folder to allow only Managers to view it, any Knowledge View that you place in the folder will inherit those permissions, and won't be accessible by other users.

It is important to remember, however, that permissions don't export. The first time you create a folder, manually or via the import process, you must explicitly set its permissions. Once you've done so, however, any new objects created or imported into the folder will automatically inherit the folder's permissions. (Remember to replicate permissions to child objects if the folder already contains objects.)

To use our example of the [Manager KViews](#) folder above, if you create new objects in the [Manager KViews](#) folder on Staging, then, when you reimport the folder to Production, all of the new objects will immediately be restricted to the permissions you set on the [Manager KViews](#) folder in the Production system. Because permissions don't export, you can set much less restrictive permissions on the [Manager KViews](#) folder in the Development environment to allow non-managers to test the system with non-sensitive data. When you re-import the folder to Staging and Production, the full set of access restrictions will be automatically applied to all the objects in the folder on the destination systems.

## Folder Structure #

Structuring your folders properly makes it easier to deploy individual projects or implementations. As mentioned in the [Creating Applications topic](#), the best practice for structuring folders is to organize them by application, and include subfolders for various [Content List](#) object types contained in the application. For instance, look at the sample structure below.



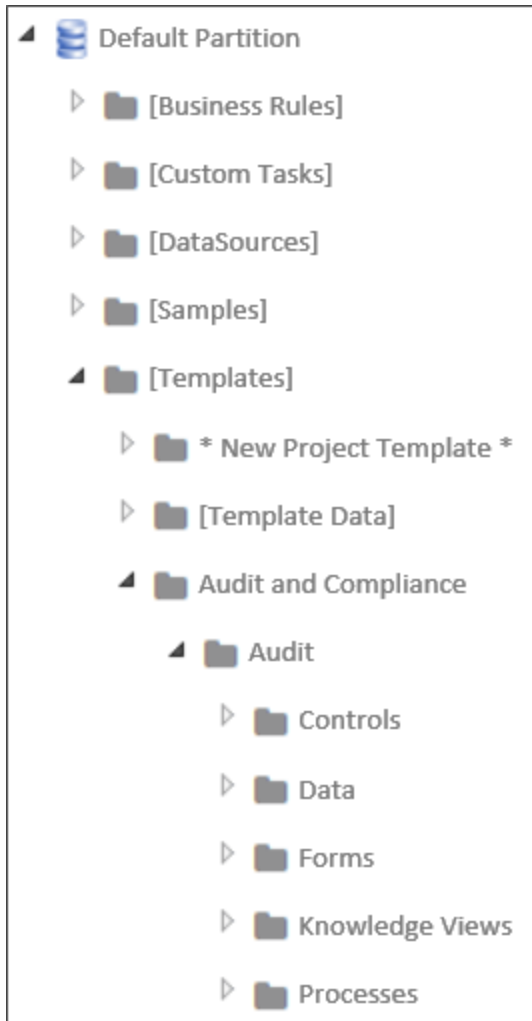
The image above is an example of structuring an implementation project logically. The top-level folder provides the name of the application you're modeling, in this case, a New Employee Onboarding process. All of the individual Forms, Knowledge Views, etc. are organized into the lower-level folders. With this type of organization, any changes you make to the application are contained inside this folder structure, which allows you to simply export the New Employee Onboarding folder, and import it into production, with all of its [Content List](#) objects intact, and without having to worry about interfering with any other application.

There are, of course, some exceptions to application-based organization. For instance, shared objects, like Datasources, Goals, and Business Values should probably each be grouped together in their own folder at the root level of the partition. You may have noticed that the Custom Tasks are organized in this fashion. You really shouldn't export/import Datasources between servers, to avoid mixing connections to test data with production data. The Data Sources should have exactly the same name, but the best practice is to create your Production and Staging Datasources manually. The important thing to remember is that you need to keep the folder structure exactly the same on your staging and production server, so that shared objects have the same relative paths on both servers.

Also, remember that any Datasources that are pointing to an external production system may need to be changed on the test server to point to an external test system. The same may be needed for some Custom Tasks that don't use a Datasource but connect to the external application directly, such as the Web Service CT. For the most part, this is important for external systems that are being updated from Process Director. If Process Director is just reading from the external system this may be less of an issue, but something you should keep in mind.

You may also want to create a root level Business Rules folder for those Business Rules that can be used in multiple projects. Business rules that are specific to a single project can be contained in the project's folder structure. Similarly, some common sub-processes might be organized into a root-level folder as well.

With the above in mind, a sample partition might look something like this:



Notice that the root folder contains folders for all of the shared objects, as well as the high-level organization for different types of applications the organization has implemented. At the second level, the HR Process folder contains the individual applications for various Human Resource operations. Finally, at the third level, the New Employee Onboarding folder contains folders for all of the [Content List](#) objects that are needed to run that specific application. BP Logix recommends that you implement a similar type of folder structure to make importing/exporting projects (not to mention simply finding things) easier.

## Users and Groups <#>

Users and User Groups aren't imported or exported in Process Director when exporting applications (though they can, of course, be [imported and exported separately by system administrators](#), using the procedure described below. So, be sure that any test users or groups to which a project refers in the Development environment also exist in Staging and Production. If you are syncing users in all three systems from Active Directory, this won't usually be a problem. But there may be cases where you've manually created users in the Development system, and in such cases, the users will have to be manually created on, or exported to, the Staging and Production systems prior to the application.

So, make sure that all users match in your test and production environments. If they do not, and you attempt to import a project with a reference to a user that doesn't exist on the target system, an import error will occur. (Keep in mind, though, that explicit references to individual users within an application's Process Timeline activities aren't a good idea to begin with.)

Speaking of import errors, if an import throws an error or warning...**stop**. Fix the problem and try again. There are several errors that are common when importing, such as the user mismatch we just discussed. Another very common error is forgetting to add a Datasource for a dropdown that uses the Fill Dropdown from DB Custom Task. Commonly, the problem arises when you create a new Datasource on the development server, but neglect to create a matching Datasource on the production system before performing the import. The bottom line is that if you see an import error, your project didn't import properly, so you need to fix the problem and try to import it again.

Another important caution to remember is to avoid manually renaming any [Content List](#) object on the target system (such as the Production system) that may ever be updated via import. Doing so can result in no end of problems the next time you perform an import. If you do need to rename an object, you must do that manually on both the destination and source servers prior to doing the import. The development/staging/production objects must all have the same names.

To understand why, we need to delve down a bit into how Process Director tracks objects, and how the import process works. Process Director is database-driven, meaning that every single object in Process Director is stored in a database. The database uniquely identifies every object by assigning it a special identifier known as a [Globally Unique Identifier](#) (GUID). A [GUID](#)<sup>1</sup> is a 128 bit, 36 character value, and is a series of [hexadecimal](#)<sup>2</sup> numbers that are created using algorithms that ensure uniqueness. The GUID is generally represented in the format:

```
aa213c6f-a8aa-454f-a04d-30b56fd2e493
```

GUIDs don't mean anything to us humans, of course, so we always provide logical names, like "Manager KViews", that we humans can understand when we create objects. But, in almost every case, Process Director never uses the name. It always uses the GUID. Two important exceptions to GUID-based identification are import/export, and when referencing objects and controls via the system variable syntax, e.g. `{RULE:RuleName}`, `{:FieldName}`.

But, ***GUIDs can't be exported or imported because they are unique.*** An object on the Staging system will have a completely different GUID than the parallel object on the Production system. Since that is so, the import process has to try and match the logical name that we have given the object when performing an import.

So, to return to our [Manager KViews](#) folder, if we change the name of the folder on the Production system to [Mgr KViews](#), then the next time we import from Staging, problems will arise. The import process will be

---

<sup>1</sup>GUIDs generated from random numbers are so large that the probability of the same number being generated randomly twice is negligible. Assuming uniform probability for simplicity, the probability of one duplicate would be about 50% if every person on earth owned 600 million GUIDs. (Wikipedia)

<sup>2</sup>In mathematics and computing, hexadecimal (also base 16, or hex) is a positional numeral system with a radix, or base, of 16. It uses sixteen distinct symbols, most often the symbols 0–9 to represent values zero to nine, and A, B, C, D, E, F (or alternatively a, b, c, d, e, f) to represent values ten to fifteen. (Wikipedia)

looking to match the [Manager KViews](#) folder on Staging with the same folder on Production. Because we've changed the name on production, however, the import can't match the text "Mgr KView" to "Manager KView". It will, therefore, ignore the renamed [Mgr KViews](#) folder on the Production system, and create a new [Manager KViews](#) folder on import. Similar problems arise if you rename the folder on the Staging system before importing it to Production.

The new [Manager KViews](#) folder won't have any of the restrictive permissions you placed on the original folder. As far as the Production system is concerned, the newly-imported [Manager KViews](#) folder is a completely new folder, with the default permissions that will allow anyone to access it. You now have a security hole in your Production system, while the secured object has been orphaned, with all of the imported links now pointing to an unsecured folder.

So, it is vitally important to remember that you should only rename objects in the target environment when there is a real need, and if you do, you must manually rename it in the source environment.

Now, at this point, you might be thinking that you could simply delete the orphaned object. But wait, it gets worse.

If there are other objects on the production system that were not part of the import, and that were linked to the [Mgr KViews](#) folder when you renamed it, those links still exist. Those object links were not replaced with links to the [Manager KView](#) folder. So, now, your production system may have some objects linked to the [Mgr KViews](#) folder, while other objects are linked to the unsecured [Manager KViews](#) folder.

If you simply try to delete an orphaned object, the remaining links to orphaned object will now be broken, so you may break all of those related applications as well. But, even worse, when you delete an object, you also delete every instance of that object. That might not be a big deal if the object is just a Knowledge View, but if the object is a Process Timeline or Form, then every single instance of that object, and all of the data associated with those instances, will be deleted from the system. You have just deleted all of the historical data in your system for that process or Form. Also, you'll delete any link that any other object has to the object, so may destroy more than one application.

If you find yourself in a situation like the above, where you've renamed an object, performed an import, and now have two objects, you have two options. First, you can call BP Logix Technical Support for assistance, which is your best option. Second, you can try to fix the problem yourself by following the procedure below:

1. Back up your Process Director database for the Production system.
2. Identify the new objects that were just created on the Production system because they were renamed.
3. Remove the new objects from Production.
4. Rename the objects on Production so they match the Staging system.
5. Re-import the objects.

With the above in mind, you may conclude that, as long as you aren't importing or exporting, you can rename or delete objects that don't have instances, like KViews or Business Rules, without running into problems. But, that's not completely true either, because, in addition to importing/exporting, there is one other case where the name of the object, rather than the GUID, is vitally important. Sometimes, you refer to objects through System Variables. For instance, you might have a condition in a process or Form that relies on the result of a Business Rule where you've used a System Variable to return the result, e.g.,

{RULE:RuleName}. System Variables are name-based, not based on the GUID. That means that if you rename or delete an object that is named in a System Variable, you'll break the object that uses the System variable as a reference.

The general rule, therefore, is that deleting object definition in the [Content List](#) of a Production system is an extraordinarily bad idea. If you do so, and eliminate vital historical data, then you have to try to restore your database which is a complicated and risky course of action.

So, to avoid the massive headaches any deletion can cause, only Partition Administrators should be able to delete anything on Production. Moreover, no actual user should ever be given Partition Administrator privileges, in order to ensure that no user can accidentally delete anything.

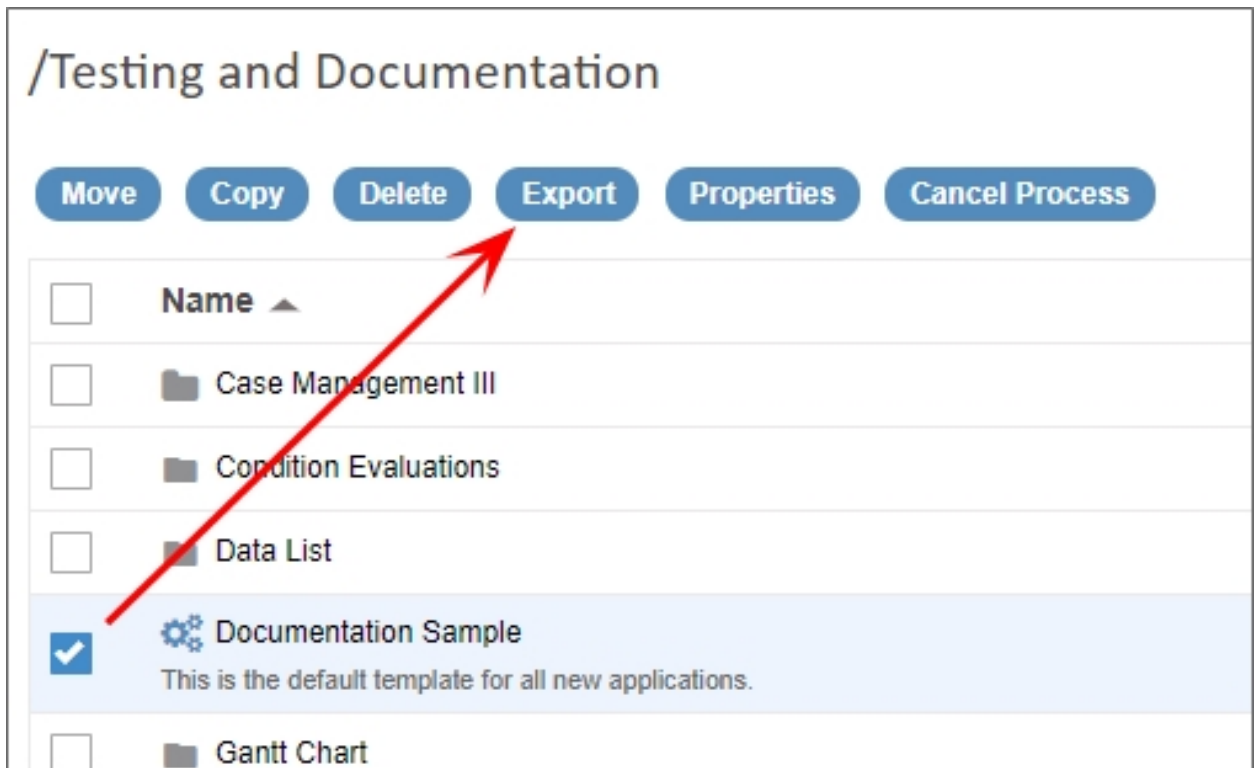
Instead, create a specific Partition Administrator account, and require administrative personnel to log out of their personal accounts and log into the Partition Administrator account before deleting anything. And even then, they should take extraordinary care before deleting an object. There are, in fact, a number of other security options you should consider as well, all of which can be found in the Securing Process Director section of the System Administrator's Guide.

## Exporting and Importing Objects <#>



### Exporting Content List Objects

Process Director allows groups of objects (folders, documents, process definitions, etc.) to be exported or imported as a single file. This is needed when copying a [Content List](#) object/folder/application from a test system to a production server or importing Custom Tasks in to your system. It can also be used to backup entire Process Timelines as you are working on them. To export objects, check off the box next to the names of the files you wish to export, and click the **Export** action button.



**i** Prior to Process Director v5.25, exports were done in the XML format by default, though v4.55 and higher accepted exports/imports in the ZIP file format, with the ZIP file containing a compressed version of the XML export file. Process Director v5.20 and higher enables exports in the compressed PDZ file format by default. Users have the choice of selecting the ZIP file format for export to v4.55-v5.20, and XML format for all earlier versions of the product. The PDZ file format isn't a special or proprietary format. It 's actually a regular ZIP file, with the file extension simply changed from ZIP to PDZ (Process Director Zip) to indicate that it's a ZIP file containing a Process Director XML export. Renaming a PDZ file extension to ZIP will result in a normal ZIP archive file.

**Export Objects**

**Export**  
Exporting Content Objects: Documentation Sample

Save as an XML file to my local drive    Create backup of folder in Content List

**Export File Name**  
Documentation Sample   Default (Process Director 5.20 and higher) ▼

**Export Package Description**

OK    Close

When you click the **Export** button, a dialog box will appear that enable you to make changes to the export file name or format. Clicking the **OK** button will begin the file export, which, for most browsers, will automatically export the file to the default Downloads folder on your local machine.

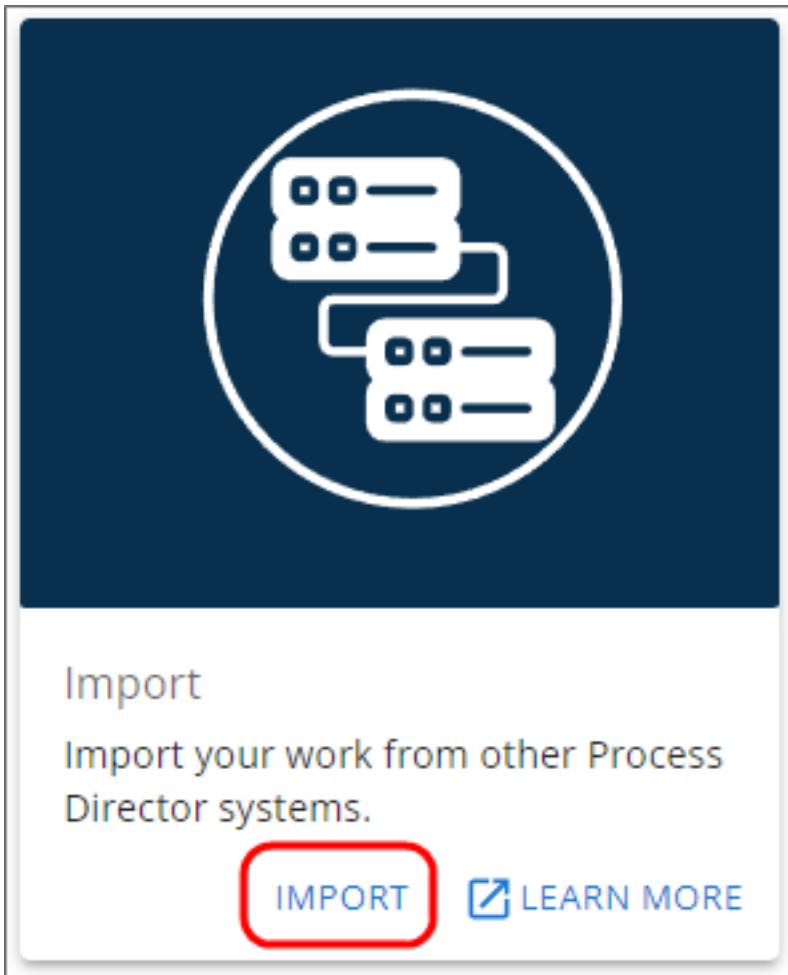
[Content List](#) items and Meta Data must be exported separately.



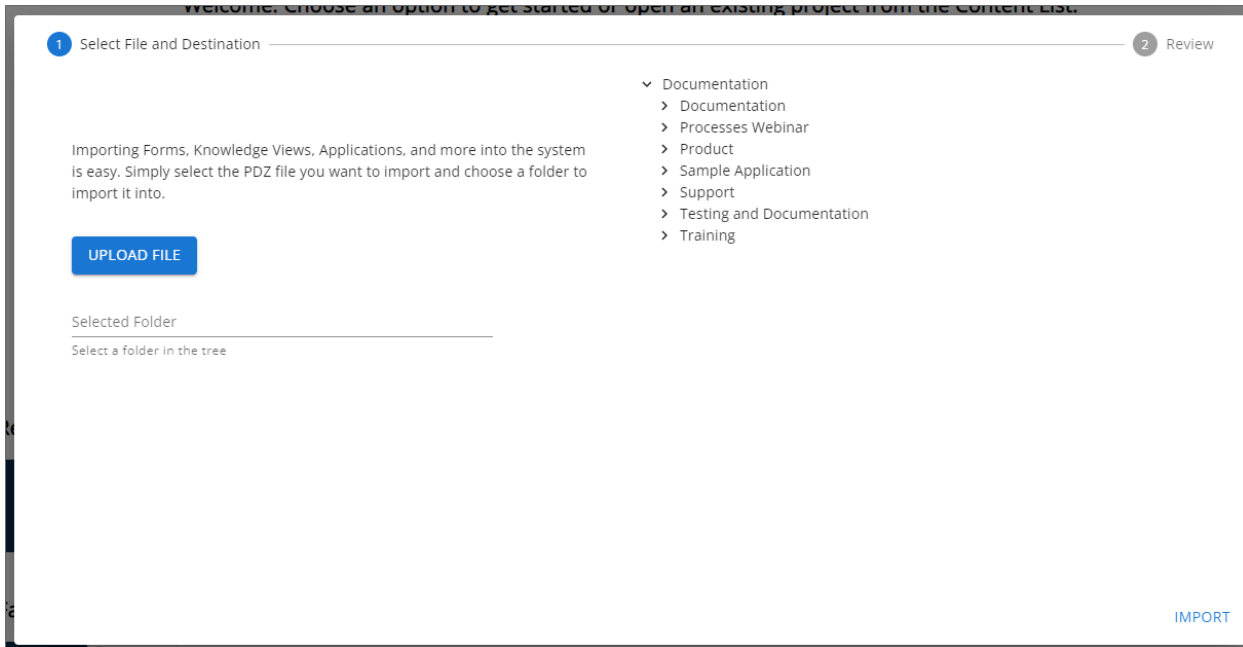
### Importing Content List Objects from the Home Page

For users of Process Director v 6.0.300 and higher, Process Director objects can be imported onto the server directly from the [Home](#) page. A large panel at the top of the home page, labeled **Import**, displays an **Import** button.



The image shows a user interface element for importing data. It features a dark blue square at the top containing a white circular icon with two server racks connected by lines. Below this is a white rectangular area with the text 'Import' in a large font, followed by 'Import your work from other Process Director systems.' in a smaller font. At the bottom of this area are two buttons: 'IMPORT' in a rounded rectangle with a red border, and 'LEARN MORE' with a blue link icon and text.

When clicked, the Import button will open an import wizard that will guide you through the process of importing the PDZ file containing the [Content List](#) objects.

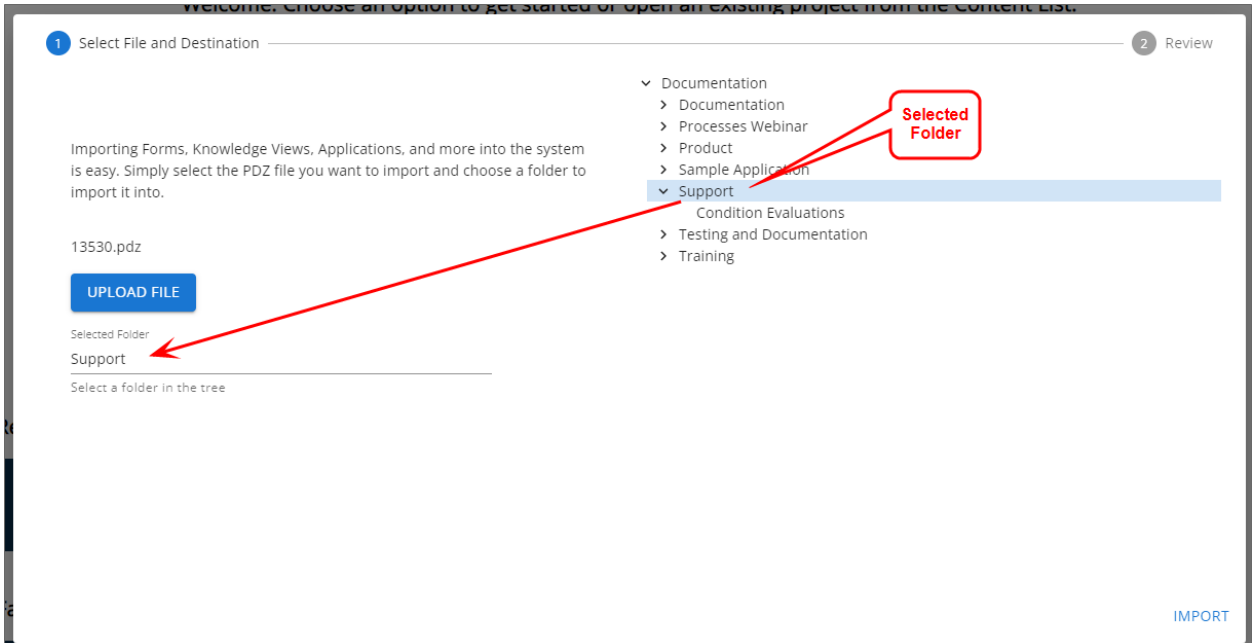


To begin the import click the **Upload File** button, which will open the Windows File Open dialog box. You can use this dialog to navigate to the location of your PDZ file, select it, then click the **Open** button to specify the file to import. Once the file is selected, it will be displayed in the wizard screen above the **Import File** Button.

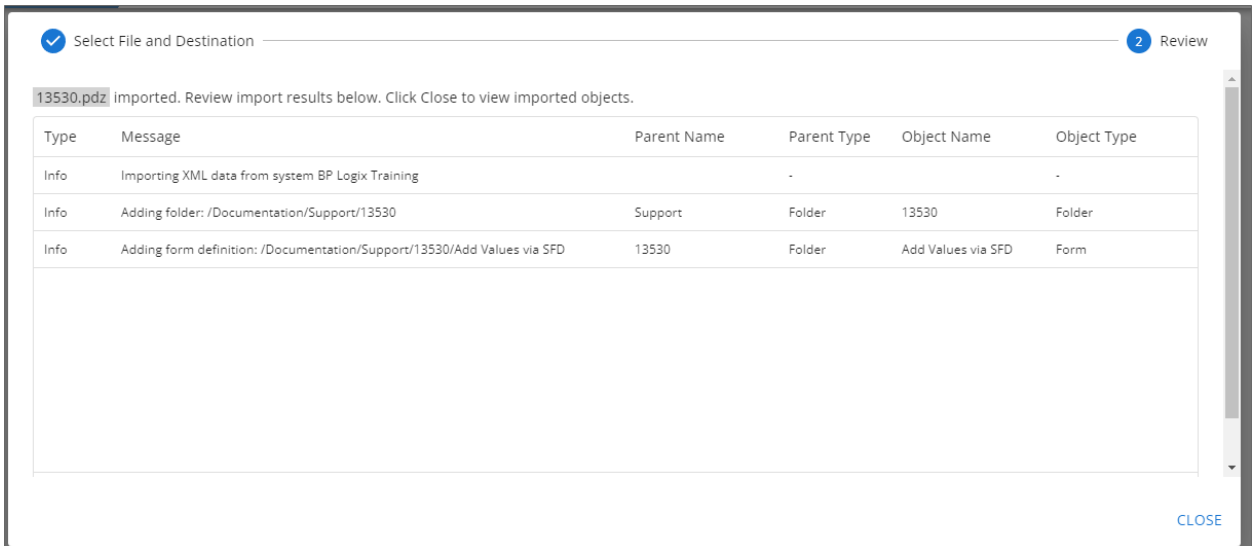


On the right side of the wizard, a list of folders will appear. Using this list navigate to the folder into which you wish to import the object. If the PDZ file contains a single object, the object will be imported into the selected location. For PDZ files that contain applications inside a single parent folder, the parent folder will be created as a subfolder of the selected **Content List** folder.

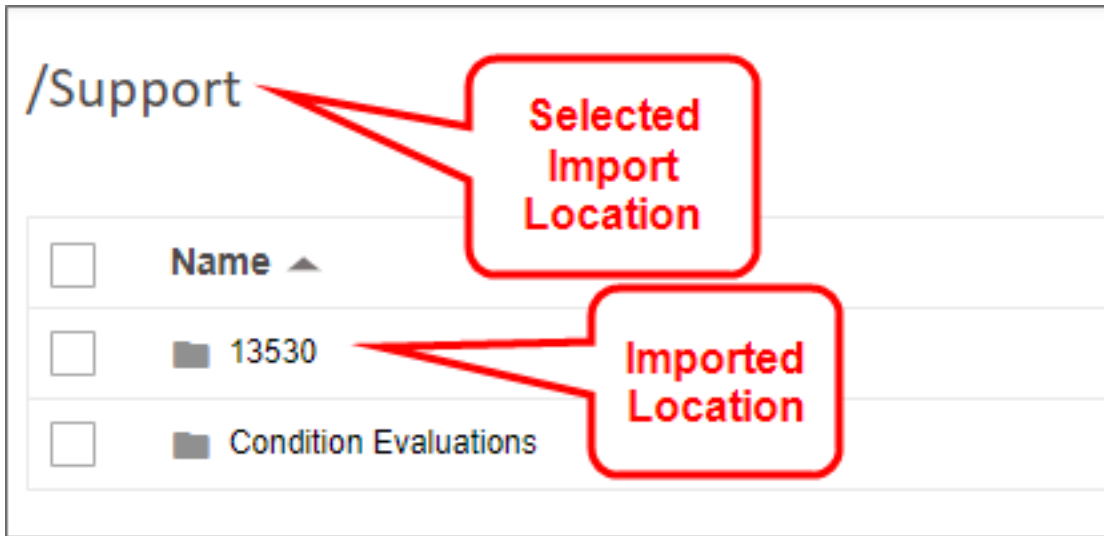
Once you select a folder, it will be highlighted on the right, and the name of the selected folder will be displayed in the **Selected Folder** property displayed on the left.



You can now click the **Import** button to import the PDZ file into the selected folder. A Progress icon will appear while the import process occurs. After a successful import, the Import Report will be displayed in the wizard to enable you to ensure the import ran correctly.

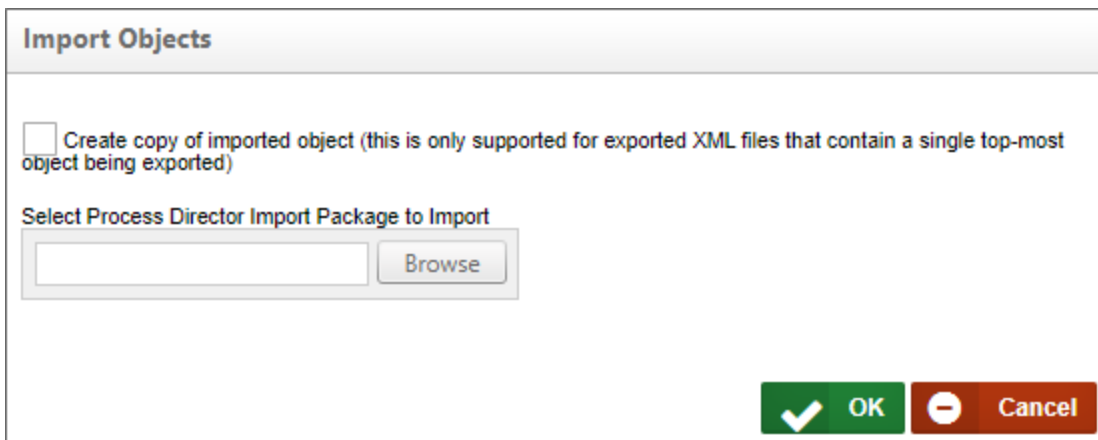


After reviewing the Import report, you can click the **Close** button to exit the wizard. Once the wizard exits, Process Director will automatically navigate you to the specified import location in the **Content List**.



### Importing Content List Objects from inside a Folder)

To import a file containing [Content List](#) objects, first navigate to the folder in the [Content List](#) where you want the imported item(s) to appear. Next, select [Import Objects](#) from the [Create New](#) dropdown located in the upper right corner of the screen. Selecting this option will open an import dialog box.



From this dialog, you can use the [Browse](#) button to browse to the location of the file, and select it. Once selected, clicking the [OK](#) button will begin the import process.

When the import process concludes, an import report will appear automatically.

Import Objects					
Import Results					
Type	Text	Parent Name	Parent Type	Name	Type
Info	Importing XML data from system BP Logix Training		-		-
Info	Replacing folder: /Default Partition/Dale/Training/System Variables	Training	Folder	System Variables	Folder
Info	Replacing form definition: /Default Partition/Dale/Training/System Variables/System Variables	System Variables	Folder	System Variables	Form
Info	Replacing Timeline: /Default Partition/Dale/Training/System Variables/System Variables	System Variables	Folder	System Variables	Timeline Definition
Info	Replacing form definition: /Default Partition/Dale/Training/System Variables/Email Template	System Variables	Folder	Email Template	Form

You should peruse this report to ensure there are no warnings or errors for the import. If there are, you should correct the errors, then re-import the objects until the errors and/or warnings are resolved.

## Other Functions #

### Copying Objects between Content List locations

To create a copy of a folder within the [Content List](#), first export the folder using the process described above. Doing so will create an XML file containing the exported information.

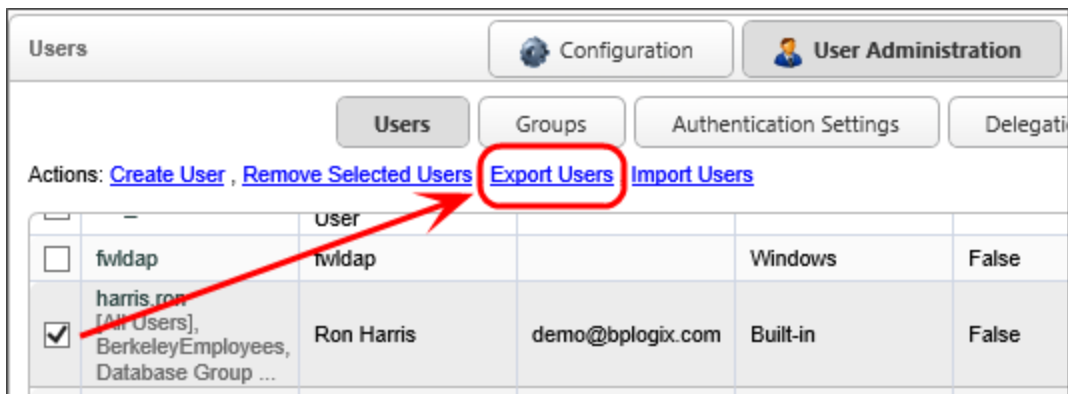
Next, import the XML file that you just created. When the import screen appears, check the box labeled "Create copy of imported object (this is only supported for exported XML files that contain a single top-most object being exported)". This will import the file as a copy of the existing item. If you don't select the checkbox, the import will overwrite the existing item.

Please note that this "create copy" feature will only work when importing an XML (or PDZ) file that contains either a single folder as a top-level object, or a single [Content List](#) object. If you wish to import multiple Content List objects, therefore, they must be exported in a folder, not as a collection of individual objects. The exported folder can then be re-imported as a copy.

As a quick example, if your original folder is called "vacation request", the XML file you create through export will be called "vacation\_request.pdz", and the new [Content List](#) folder you create through importing the XML file will be called "vacation request copy".

### Importing / Exporting Users Using Excel

Users in Process Director can be exported to and imported from a Microsoft Excel spreadsheet. To export Process Director's users to an Excel file, first ensure that you are logged in as an administrator. Navigate to [IT Admin](#) > [User Administration](#) > [Users](#), and click on the [Export Users](#) link. If you only wish to export specific users, you can select those users by clicking the checkboxes next to their names, and then clicking [Export Users](#).



The resulting excel file should contain information about the users on this Process Director installation, and should look like this:

	A	B	C	D	E	F	G	H	I
1	UserID	UserName	EmailAddress	Culture	TimeZone	Disabled	AuthType	LastLogin	Groups
2	Administrator		administrator@bplogix.com			False	BuiltIn	5/17/2013	admin, group2
3	user1		user1@bplogix.com			False	BuiltIn	2/1/2013	2
4	user2		user2@bplogix.com			False	BuiltIn	2/1/2013	2

You can also include other columns that contain additional information about the user, such as contact information and information about the user's role in the company.

J	K	L	M	N	O	P	Q	R	S	T	U
Phone	Description	Title	Office	Company	Department	CustomString	CustomNumber	CustomDate	LegalEntity	BusinessUnit	Manager

The Manager field can refer to the Manager's User ID, or Process Director's internal oUID for that user.

We can then edit much of the information on the Excel file. We can change the names of the users in the UserName column, or their email addresses in the EmailAddress column. The Culture column specifies which language and time format and number format the user will view, and the TimeZone column specifies the user's time zone, using [.NET's list of time zones](#). The column values for some common time zones are as follows:

- Pacific Standard Time (US and Canada West Coast, GMT-08:00)
- US Eastern Standard Time (United States East Coast, GMT-05:00)
- Greenwich Standard Time (UK, GMT)
- Central Europe Standard Time (Central Europe, GMT+01:00)

- Tokyo Standard Time (Japan, GMT+09:00)
- AUS Eastern Standard Time (Australian East Coast, GMT+10:00)

You also disable users and/or specify the date and time of their most recent log in. **AuthType** determines how the user authenticates, whether it's through Windows, Process Director, or another method. The Groups column is a comma-separated list of columns to which the group belongs. UserIDs identify the users: they must be unique, as each different UserID specifies a different user.

After creating or editing an Excel file, we can import it into Process Director at the same page from which we exported the Excel file. After navigating to [IT Admin](#) > [User Administration](#) > [Users](#), click on the **Import Users** link.

To import users, specify the name of the worksheet in the Excel file that contains information about the users. By default, the name of this worksheet is “Users”, and an Excel file created by Process Director will store the user data in a worksheet called “Users”.

Checking the “Create groups as needed during import” checkbox will automatically create groups for any group specified in the Excel file that doesn't currently exist in Process Director. If this checkbox is checked, and you add a user to a group that doesn't exist, that group will be created upon import.

Checking the “Remove users from groups not listed in import file” checkbox removes users from groups unless it is specified in the file that they belong to that group. So, if in Process Director I have a user, Administrator, belonging to groups “admin” and “group2”, and I import a file that just specifies that the Administrator user belongs to the group “admin”, then, when this checkbox is checked, the user Administrator will be removed from group2 on import. Nothing happens to the group itself, but any user whose membership to that group isn't specified in the Excel file will be removed from that group.

### Importing Custom Tasks

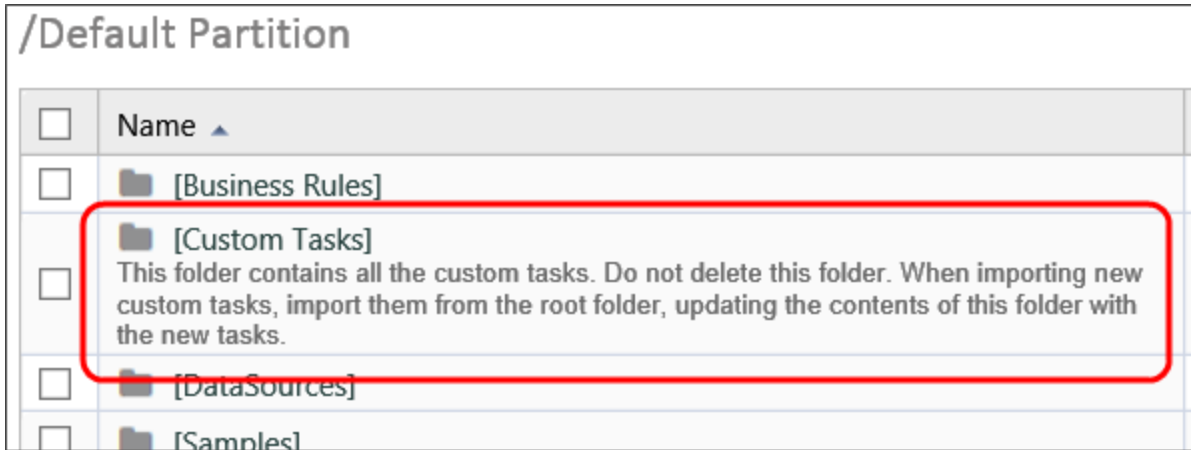
Custom Tasks may need to be periodically updated, as BP Logix updates or adds Custom Tasks on an ongoing basis. Custom Tasks are usually unrelated to any specific version of Process Director, though there are a few exceptions that require a specific product version to work properly. When upgrading the product to a new version, you may wish to upgrade the Custom Tasks after you complete the product upgrade.

To ensure proper installation and upgrade of the Custom Tasks review the following procedures:

Do not delete the current Custom Tasks unless you are sure you want to delete them. Deleting Custom Tasks will remove all Custom Task event mappings to any objects in the partition.

1. To import Custom Tasks, ensure that you aren't inside of the [\[Custom Tasks\]](#) folder, or any of its subfolders, unless you are importing an individual Custom Task to a specific location.

Navigate to the root of the location of the Partition, The [\[Custom Tasks\]](#) folder should always reside in the Partition root, as shown below.:



2. Continue to import the Custom Tasks by selecting **Import Objects** from the **Create New** dropdown. Once you've selected the PDZ file to import, click **OK**. This will import and overwrite the current [\[Custom Tasks\]](#) folder with the new/updated Custom Tasks.






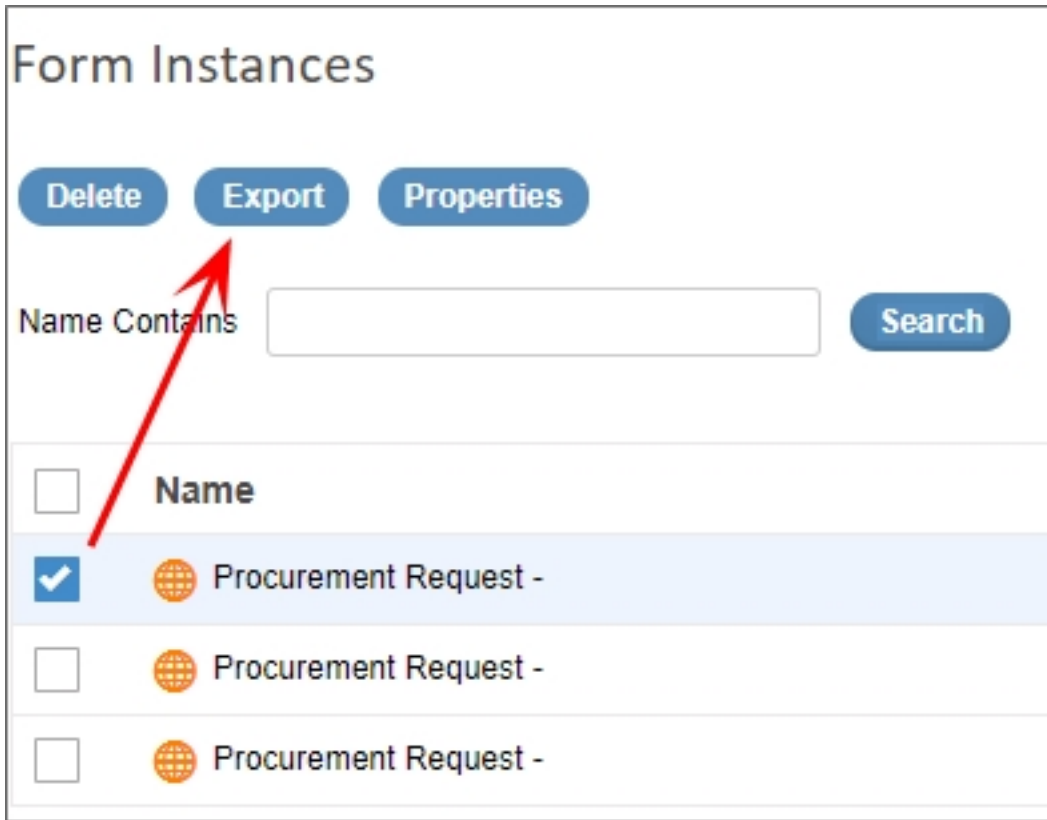
3. Ensure that there are no duplicate [\[Custom Tasks\]](#) folders. If not, then you've successfully updated your Custom Tasks.

### Exporting Form Instances

For users of Process Director v4.54 and higher, you can export form instances. This is a feature that enables a relatively rare use case where you need to create Form instances to store configuration data, and you want to export the form instance(s) containing the configuration data between systems.

 This feature is *not* designed to export regular form instances between systems, as it does *not* export the linkages to process instances or to any other objects. The export *only* exports the discrete form instances.

In the [Content List](#), when viewing form instances, clicking the check box next to the form instance will show the action buttons, which now include an [Export](#) button.



## Home Page Import Wizard #

For Process Director v6.0.300 and higher, a wizard located on the [Home](#) page can be used to import a PDZ file containing [Content List](#) objects.

 This is a documentation stub for an upcoming Process Director feature, and is subject to change without notice.

## Complex Application Imports #

When creating a new application, there may be other Process Director objects, like Workspaces and Meta Data categories, created as part of the application. Exporting other Process Director Objects, like Workspaces or Partition Meta Data, is done from the [IT Admin](#) area of the installation, on the appropriate page for administering those item types. For example, to export a workspace, you must go to the [Workspace](#) page of the [Configuration](#) section to perform the export.

The process for exporting these objects is very similar to the method for exporting [Content List](#) items. Each object type must be exported separately, and are exported into their own separate PDZ files.

These associated objects will need to be imported separately from the Application folder, and **the order in which these other objects are imported is critical** to avoid import errors. Process Director objects exist in a hierarchy, where lower-level objects can only import correctly if the associated higher-level objects already exist on the system. For instance, if a Form has a Meta Data Category assigned to the Form

Definition, you can't import the Form correctly unless that Meta Data category already exists on the target system.

While the order in which you import objects to a target system is important, the order in which you export the objects from the source system is not.

To perform complex import operations like these, the Process Director objects should be imported in the following order.

1. **Partition Meta Data:** Meta data can be applied to any Process Director object, which means it sits at the highest level of the Hierarchy.
2. **Users/Groups:** Any new users or groups that have been created on the source system, and that are used in the application, must be imported. Exports of Groups and Users are, unlike other objects, exported and imported as Excel files rather than PDZ files.
3. **Partition-Level Content List Objects:** If you have created additional Business Values, Dropdown Objects, or Business Rules that reside in the partition's "system" folders (e.g., [\[Business Values\]](#), [\[Business Rules\]](#), etc.), import them to the appropriate place in the [Content List](#) before importing the application. BP Logix does **not** recommend importing Datasource objects between systems. Instead, they should be manually created on the Production system, using the same name as they have on the Development/Staging systems. You should have a [\[Data Sources\]](#) folder at the partition root of all systems, which is where all Datasource objects should reside.
4. **Application Content List Objects:** Once the higher-level objects have been imported, you can now import the PDZ file containing the application folder and all its contents.
5. **Workspaces:** Workspaces are usually configured with specific groups or users assigned to them. Similarly, a Workspace may display an application Dashboard or Knowledge View, or have navigation buttons that reference application Forms. Importing Workspaces before importing the required objects will result in an import warning. The Workspace will be imported, but any missing users or groups or application objects will be removed from the Workspace.

Remember, failing to import objects in the proper order will result in import errors that will necessitate re-importing the objects.

## Template Library

For customers who have a Cloud or Subscription license for Process Director v5.31 and higher, the Template Library enables you to easily replicate an existing Template as a new application.

Many organizations have specific design guidelines that they would like to see replicated in any application. For instance, a specified logo, color, or font might be required to display on each form exposed to end users, along with some commonly required Form fields. Additionally, system administrators might want to provide one or more "shell" applications with an associated Form and Process Timeline already provided for each new application, simply to save on the time and tedium of creating the same objects from scratch. Templates enable you to achieve those objectives easily. Please see the [Creating an Application Template topic](#) for more information on how to create a template for this purpose.

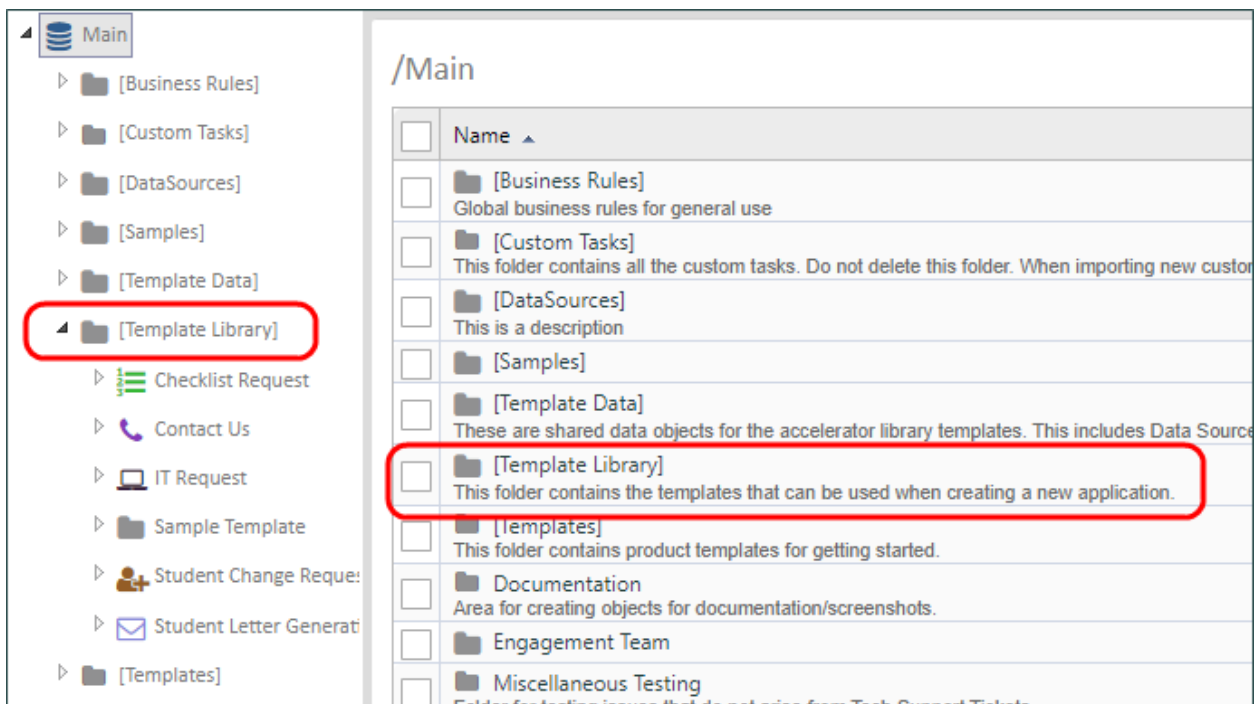
A **Template** is a pre-made application, containing all of the design features and functionality you desire, to use as the basis of a new object. The Template can be as simple as a single form with some font and

color styling, or as complicated as a fully functional application that you need to replicate. Any Process Director object, or collection of objects, can be used as a Template.

Once you have created a template, the Template Library feature enables you to select that template as the starting point for a new application by copying the existing template and placing the copy of it in any desired location on your Process Director installation. All of the existing configuration options of the template objects will be replicated in the new application.

## The [Template Library] Folder #

For the Template Library feature to work properly, you must have a [Template Library] folder at the root level of the partition.



If you don't have this folder installed by default, you can simply create it at the root level of the partition. Once you have done so, you now have a template container folder that can be used for that partition. If you have multiple partitions on your installation, each partition will require its own [Template Library] folder.

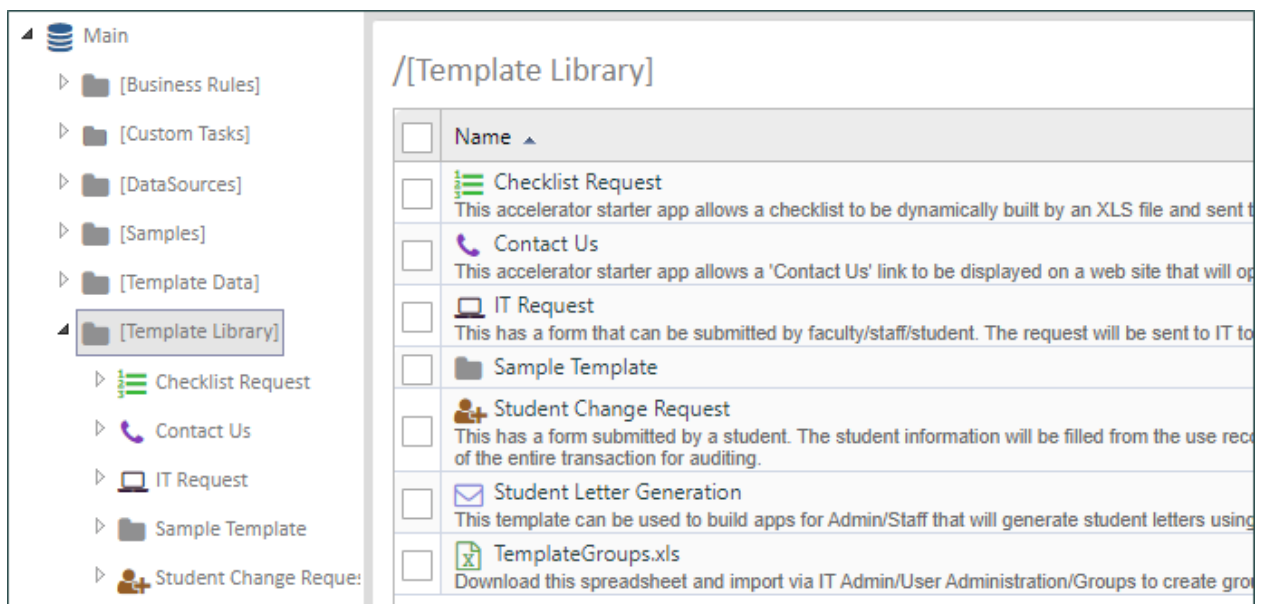
**i** For v6.1.100 and higher, installations without a [Template Library] Folder will display an error message to that effect on Staging and Development systems.

BP Logix provides some pre-built template applications that can be downloaded from the BP Logix Support site's [Download page](#), in the section labeled, "BP Logix Template Library". The download file is a zipped PDZ file that can be imported into the root of any partition. When importing, the PDZ will create the [Template Library] folder at the Partition root. This folder will contain several template applications from BP Logix that you can use as starter templates. The Import will also create a [Template Data] folder that includes some Dropdown Objects, Business Values, and an Internal User Database data source for use in

the pre-built template applications. This folder is required for the pre-built applications to work properly. There's also an Excel spreadsheet with administrative items like user Groups that must be edited and imported into your system prior to using some of the pre-made template applications. This Excel file is named TemplateGroups.xls.

Inside the [\[Template Library\]](#) folder, each template application must reside in its own subfolder. If you do not have any template applications inside this folder, you will not, obviously, be able to replicate the template to create a new application. So, in addition to having the [\[Template Library\]](#) folder, you must have one or more application templates that reside inside the [\[Template Library\]](#) folder. Installing the BP Logix templates will, as mentioned, provide several template application for use. If, on the other hand, you manually create a [\[Template Library\]](#) folder, you'll need to supply at least one template application in a subfolder before you'll be able to create a new application from a template.

**i** For Process Director v6.0.300 and higher, if you do not have a [\[Template Library\]](#) folder, or if the folder contains no application subfolders, the user interface will display a dialog box notifying you that no templates are available.



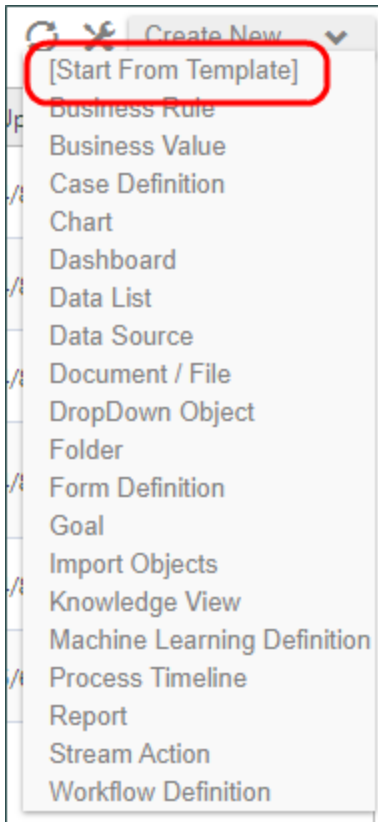
In the example above, there are seven different template application folders, each of which contain the appropriate Forms, Process Timelines, and other objects. When using the Template Library feature, all of the objects in the selected [\[Template Library\]](#) subfolder will be replicated into the location you desire.

## Creating an Application from a Template Library #

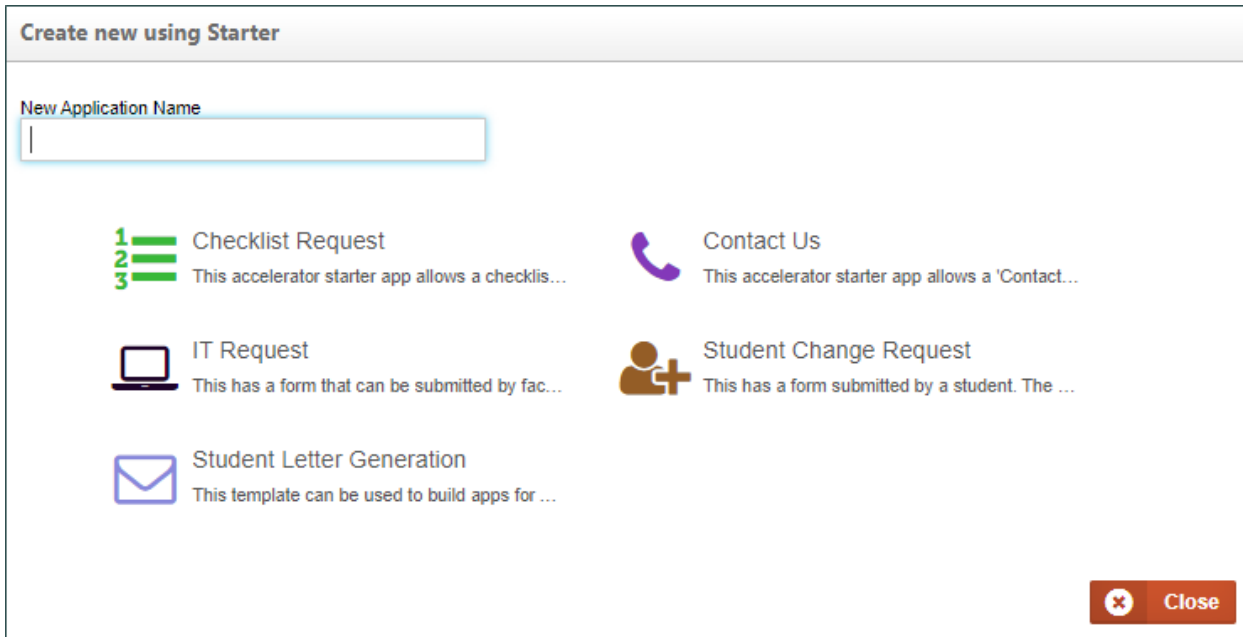
To create a new application from the Template Library, you can create it directly from a [Content List](#) folder, and, for Process Director v6.0.300 and higher, from the [Home](#) page.

### *Creating from the Content List*

First, navigate to the folder location in which you'd like to create the application. Then, select the [\[Start from Template\]](#) option from the [Create New...](#) dropdown menu.



The **Create new using Starter** dialog box will open to display the list of [\[Template Library\]](#) subfolders from which you can select the desired template. All of the existing Template templates will be shown as clickable buttons.

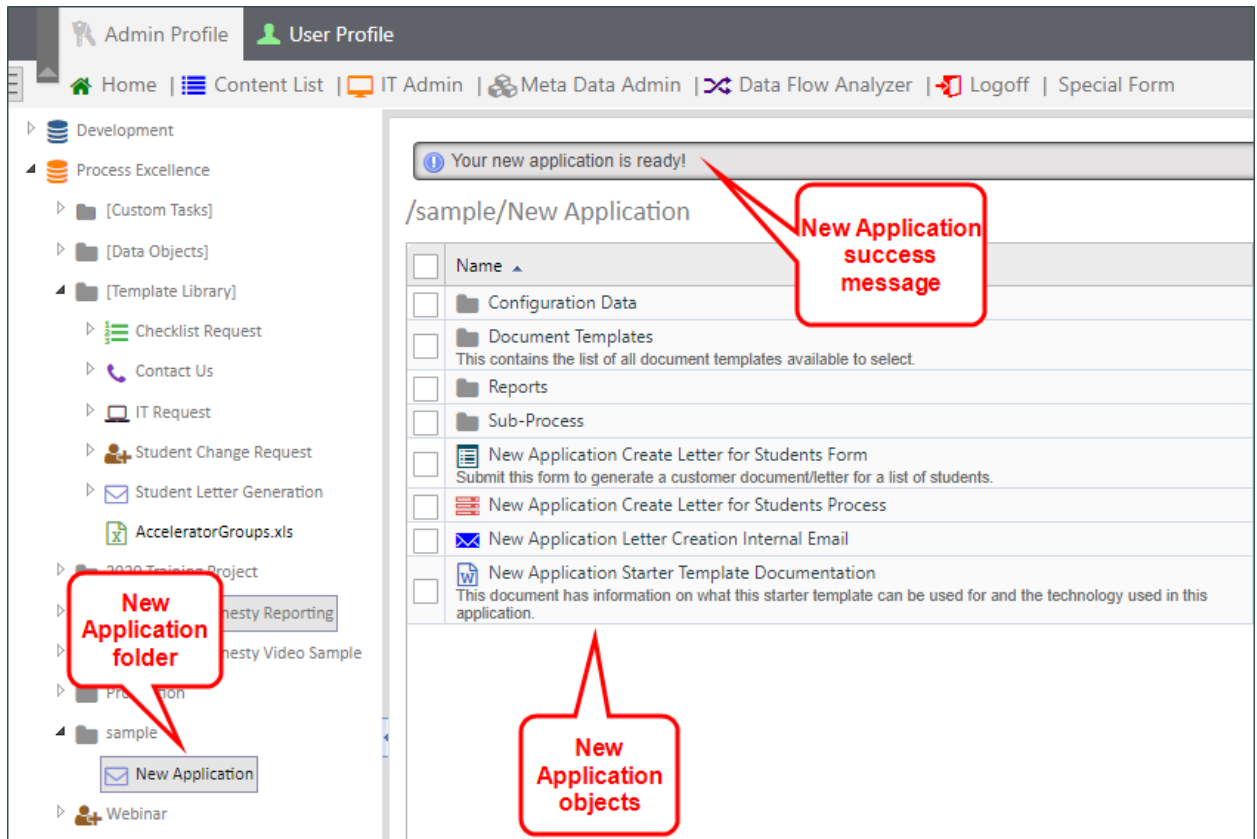


In the **New Application Name** text box, type the name you want for your new application. The name you type here will be used to name the folder that will contain your new application. This name will also be

appended to the names of all Process Director objects created in the new folder. Below the **New Application Name** text box, a list of all of the template folders contained in your [\[Template Library\]](#) folder will appear. In the case of this example, we have the Template folders we referred to above. Simply click on the desired template folder to select it. When you do so, a confirmation dialog box will appear, asking you to confirm this choice.

Once you confirm the choice in the dialog box, Process Director will automatically copy the template into a new folder in the location you chose.


In this example, we used "New Application" as the name for our sample application.



The New Application folder displays the name we provided in the **New Application Name** text box. Additionally, the names of the template objects will also display the same name, "New Application", appended to the original object name. Objects in subfolders will be similarly renamed. A success message will also appear, notifying you that the application is ready for editing. You can now configure the new objects as you desire.

### ***Create from the Home Page***

For Process Director v6.0.300 and higher, the [Home](#) page displays a panel labeled "Create from a Template", which enables you to create a new template-based application. A [Learn More](#) link will, when clicked, open this documentation topic in a new window.



Create from a Template




Reusable objects that control how timelines should run and behave.

[CREATE](#) [LEARN MORE](#)

The **Create** link will, when clicked, open a wizard that will guide you through the process of creating the new object you desire.

1 Choose Template — 2 Select Folder — 3 Create

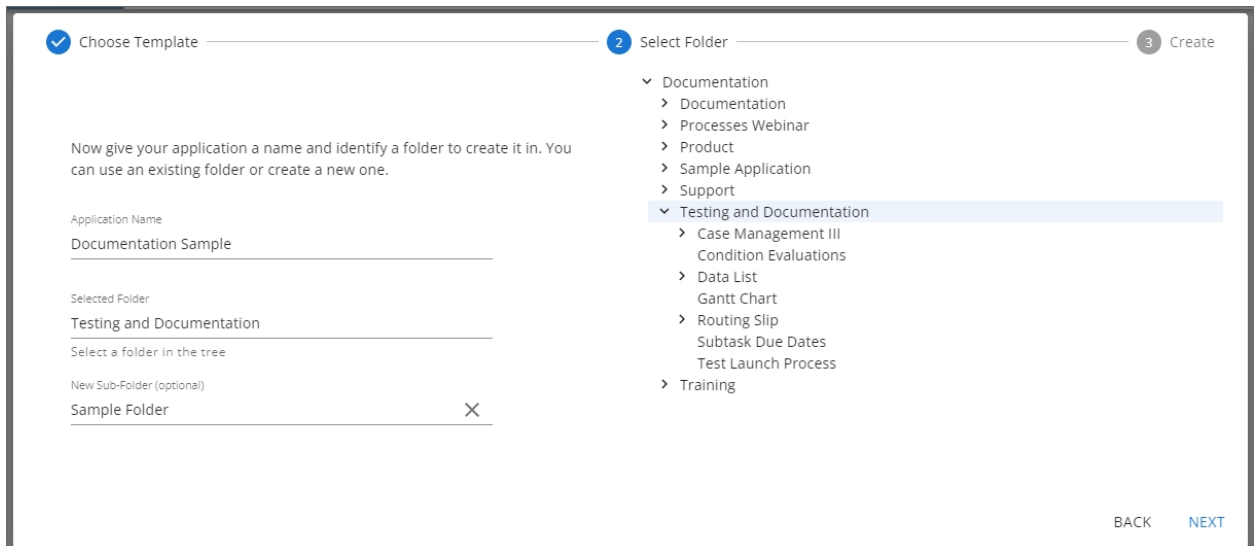
Jumpstart your application by choosing a template from the list on the right.

-  **Application**  
This is the default template for all new applications.
-  **Checklist Request**  
This template app allows a checklist to be dynamically built by an XLS file and sent to users to complete the items on the list.
-  **Contact Us**  
This template app allows a 'Contact Us' link to be displayed on a web site that will open a form allowing a user to submit a request to staff. This allows an anonymous user access (via permissions).

[BACK](#) [NEXT](#)



The first pane of the wizard is the **Choose Template** pane, which displays a list of all of the application templates that exist in the [\[Template Library\]](#) folder of your installation. Simply scroll to the template you wish to use, and click it to select it. For this example, we'll choose the **Application** template. Once you've selected a template, click the **Next** button at the bottom right corner of the wizard to advance to the **Select Folder** pane.



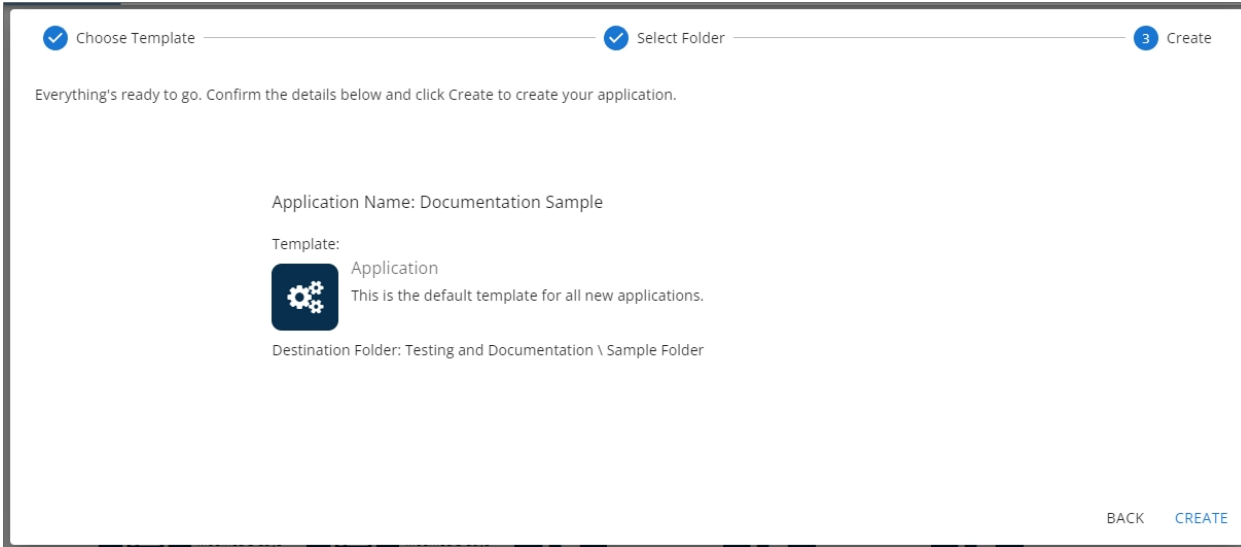
On the left side of the **Select Folder** pane, enter the name for the new application in the **Application Name** property. Once you've done so, you can navigate through the folder list displayed on the right side of the pane to find and select the folder into which you want to add the new application by clicking it (The folder must already exist in the [Content List](#)). When you do so, the folder name will appear automatically in the **Selected Folder** property, as shown above. If you'd like to place your application in a new folder that does not yet exist on the system, you can optionally enter the name of a new subfolder to create in the **New Sub-Folder** property. Keep in mind that the application itself will automatically be created in its own new folder, so if you add a **New Sub-Folder**, the application will still be created in a new folder inside the specified subfolder.

So, in the example show above, the new application, which will be named **Documentation Sample**, will be created with the following folder structure:

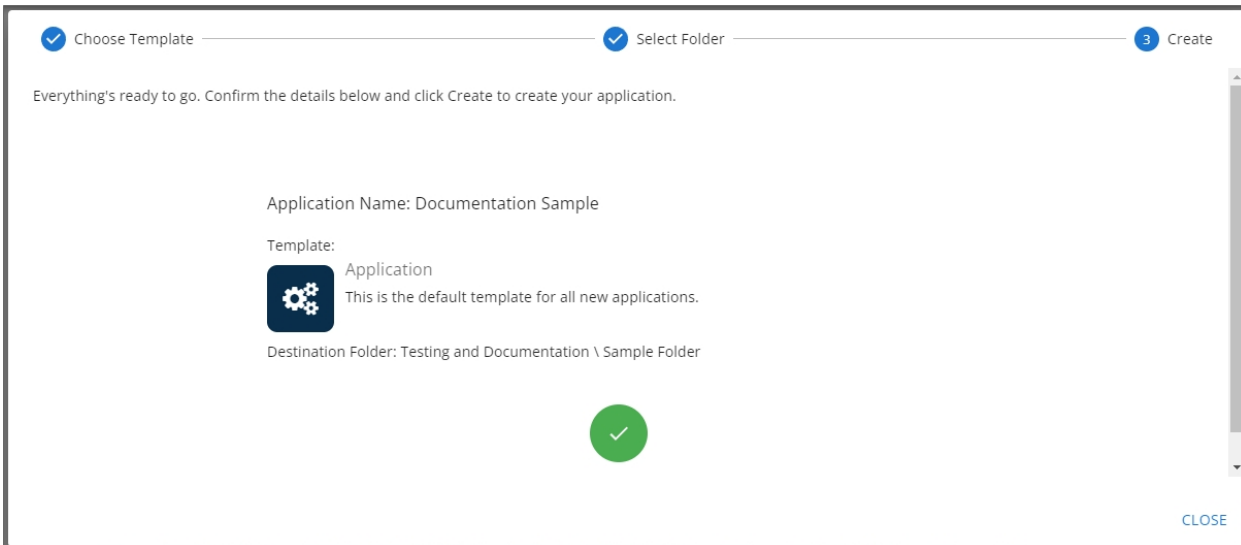
`Documentation/Testing and Documentation/Sample Folder/Documentation Sample`

All of the application objects will be located in the **Documentation Sample** folder upon creation.

When you're done configuring this pane, click the **Next** button again.



The **Create** pane enables you to review the choices you've made in the previous two wizard panes. You can, at any time, click the **Back** button to return to a previous pane to edit your choices. Once you finished reviewing your choices, you can click the **Create** button to create the new application. When you click the **Create** button, a "working" animated icon will briefly appear, followed by a success icon when the application has been created.



Click the **Close** button to exit the wizard. Process Director will automatically navigate you to the new application folder, which is **Documentation Sample** in this example, and open it for editing, so that you can complete its configuration.

The screenshot shows the BP Logix web interface. At the top, there is a dark blue navigation bar with the BP LOGIX logo and menu items: Home, Content List, Workspaces (with a dropdown arrow), and Settings (with a dropdown arrow). Below this is a lighter blue bar with 'Admin Profile' and a 'Home' button with a tree icon. The main content area displays a breadcrumb path: /Testing and Documentation/Sample Folder/Documentation Sample. Below the path is a table with two columns: 'Name' and 'Author'. Each row in the table has a checkbox on the left. The table contains the following items:

<input type="checkbox"/>	Name ▲	Author
<input type="checkbox"/>	Business Rules	Dale Franks
<input type="checkbox"/>	Knowledge Views	Dale Franks
<input type="checkbox"/>	Managers	Dale Franks
<input type="checkbox"/>	Documentation Sample Email Template	Dale Franks
<input type="checkbox"/>	Documentation Sample Form	Dale Franks
<input type="checkbox"/>	Documentation Sample Timeline	Dale Franks

## Importing Data

It's a very common requirement to use some internal organizational data in Process Director. For on-premise installations, this simply requires creating the appropriate Datasource that points to the data's repository, whether it's a CRM or ERP system, or some other SQL Server or Oracle database. For Cloud installations, however, this can be more difficult, as it can for organizations that don't wish to have even an on-premise installation of Process Director access the data directly.

Since Cloud installations are on the public Internet and most organization's have firewalls that prevent their organizational data from being accessed from outside the network boundary, access to the data is a bit more difficult. Moreover, most organizations are unwilling to open a hole in their firewall to enable a Cloud installation to access data from inside the boundary. So, for Cloud Customers, the data must be pushed to the Cloud Installation. For Process Director, the preferred method of sending data is via an Excel file, or series of files. Happily, importing the data is relatively straightforward.

First, create Excel exports of your organizational data. You can do this one time, or on a scheduled basis. When you do so, send the Excel files to a desired export folder. These files will need to be imported into Process Director, so be mindful of the technical requirements for [Excel Datasources](#).

Next, in your Process Director installation, create a destination folder that will store the Excel files in the [Content List](#). While you're at it, you should also create the Internal User Database Datasource that will be used to import the Excel files into tables in the Internal User Database.

Now, you can use the [bpImport utility](#) to transfer your Excel files from your file system, to the destination folder in your Process Director Installation. You can do this manually for the first import, but it's also

useful to set up the Window's Schedule Utility to import the files on a scheduled basis by calling the bplImport utility to send the files on the schedule you desire. The bplImport utility enables you to save the import configuration as a command that you can paste into the Windows Schedule Utility to run on schedule. For on-premise installations, the bplImport utility is available in your installation folder for Process Director. Otherwise, contact BP Logix, and we'll be happy to provide you with a copy. The bplImport utility can be run on any machine, so you don't need Process Director installed locally for it to work.

Once you've imported the files the first time, you'll need to configure the Excel files to import the data into the Internal User Database, using the Datasource you configured previously. Again, see the [Excel Datasources](#) topic for the details of how to configure this. If you're going to import the files on a scheduled basis, be sure to select the **Automatically import this Excel file after every check-in or import** property so that, every time the import runs, the data automatically gets re-imported.

Once you're done, you should have the following actions taking place on a manual or scheduled basis:

1. The data gets exported from the data repository to an Excel file or files, with the files stored in a specified file system folder.
2. The bplImport utility runs, to transfer your Excel files from the specified file location to the specified destination folder in the Process Director [Content List](#).
3. When properly configured, every time the old Excel files are replaced in the [Content List](#) by the newly imported files, the Excel data will be automatically imported into the Internal User Database.

Once the data is imported, it's immediately available for use in Process Director, via Datasources that use the Internal User database to access the data.

Obviously, you'll need to consider how fresh the data needs to be, in order to set up the appropriate schedule for the data import, as the data will only be as current as the most recent import.

## Direct URL Access to Objects

Most objects in Process Director have a unique URL that is specified in the **Properties** or **Options** tab of the object's definition. Workspaces and Documents, however, each share a generic URL, and specific Workspaces or documents are identified via a URL parameter, as described below.

## Workspaces

Hotlinks can be used to directly access a specific workspace on the system. To access a workspace directly use the following syntax:

```
https://servername.com/home.aspx?profile=profile_name
```

Where "servername.com" is the host where Process Director is installed and "profile\_name" is the name of the workspace. A user must be a member of the workspace for the page to be displayed.

## Documents

Hotlinks can be used to directly access and download documents on the system. You may access a single document, or multiple documents. A number of URL parameters can be used to define how you'd like to access the document(s).

OPTION	DESCRIPTION
Did	The Document ID of the document(s) you wish to download.
Path	The Partition path to the document.
Zipname	The name of the file to save as a zip file when downloading the document(s).
Inline	Setting this parameter to "Inline=1" will open the document in the same window (if it is web viewable).
Viewable	Setting this parameter to "Viewable=1" tells the system to open the web viewable version of the document, and is only used when Process Director is creating a web viewable rendition (e.g. PDF).
Height	The height in pixels to make the popup window to display the document.
Width	The width in pixels to make the popup window to display the document.

## Examples

### Download Via URL (Multiple Documents)

<https://servername.com/download.aspx?DL=DID1,DID2>

Where "servername.com" is the host where Process Director is installed and "DL" is the document ID of the document(s) you wish to download. In the example above, two documents would be downloaded by specifying two Document IDs as URL parameters. The specified documents would be zipped and downloaded in a single ZIP archive.

### Open Via Path (Single Document)

<https://servername.com/download.aspx?path=/Partition/Folder1/Folder2/document>

Where "servername.com" is the host where Process Director is installed and "path" is the Partition path to the document.

### Open Inline Option

You can optionally select to view the document inline in the browser, rather than downloading the document, by using the "inline" URL Parameter. This option is useful for files such as images, that can be viewed directly in the browser. For instance:

[https://servername.com/download.aspx?path=/Partition/Folder1/Folder2/image.png &inline=1](https://servername.com/download.aspx?path=/Partition/Folder1/Folder2/image.png&inline=1)

In this example, the image.png file would be downloaded and displayed in the browser, rather than downloaded to your local file system as a file.

This function is available to end users only through

- Administrators providing the URL to a user;
- Server-side scripting; or

- Client-side JavaScript to construct the URL with the appropriate document IDs, such as from a Knowledge View displayed on a form.

## Working with SharePoint



For SharePoint 365, also known as "Modern SharePoint", web parts pages have been modified by Microsoft to preclude the use of the SharePoint Picker.htm web part. The component, therefore, won't work for those SharePoint Installations.

Process Director has a number of Custom Tasks that provide interoperability between Process Director and SharePoint. These Custom Tasks allow you to do a number of things. You can pull a document from—or push a document into—a SharePoint document library. You can import and export data between Process Director Forms and SharePoint lists. You can also fill Process Director dropdown controls with data from SharePoint. All of these Custom Tasks can be used inside a Process Director Form or Process Timeline.

In this walk-through, we will step through the process of moving documents back and forth between Process Director and SharePoint using two of the Custom Tasks:

- Get Files from SharePoint.
- Push File to SharePoint.

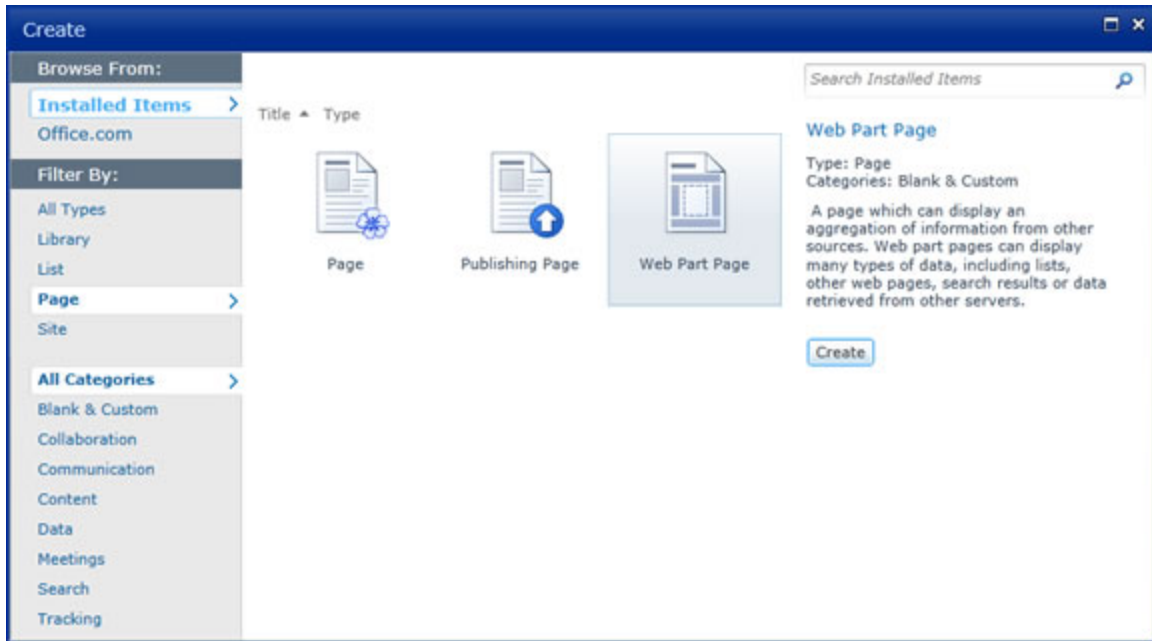
In this walk-through, we will create a simple round-trip document editing process that will allow you to retrieve a document from a SharePoint document library, edit it, then return the new version of the document to the SharePoint document library. All that's required to complete this process is a simple Process Director Form.

### Configuring SharePoint #

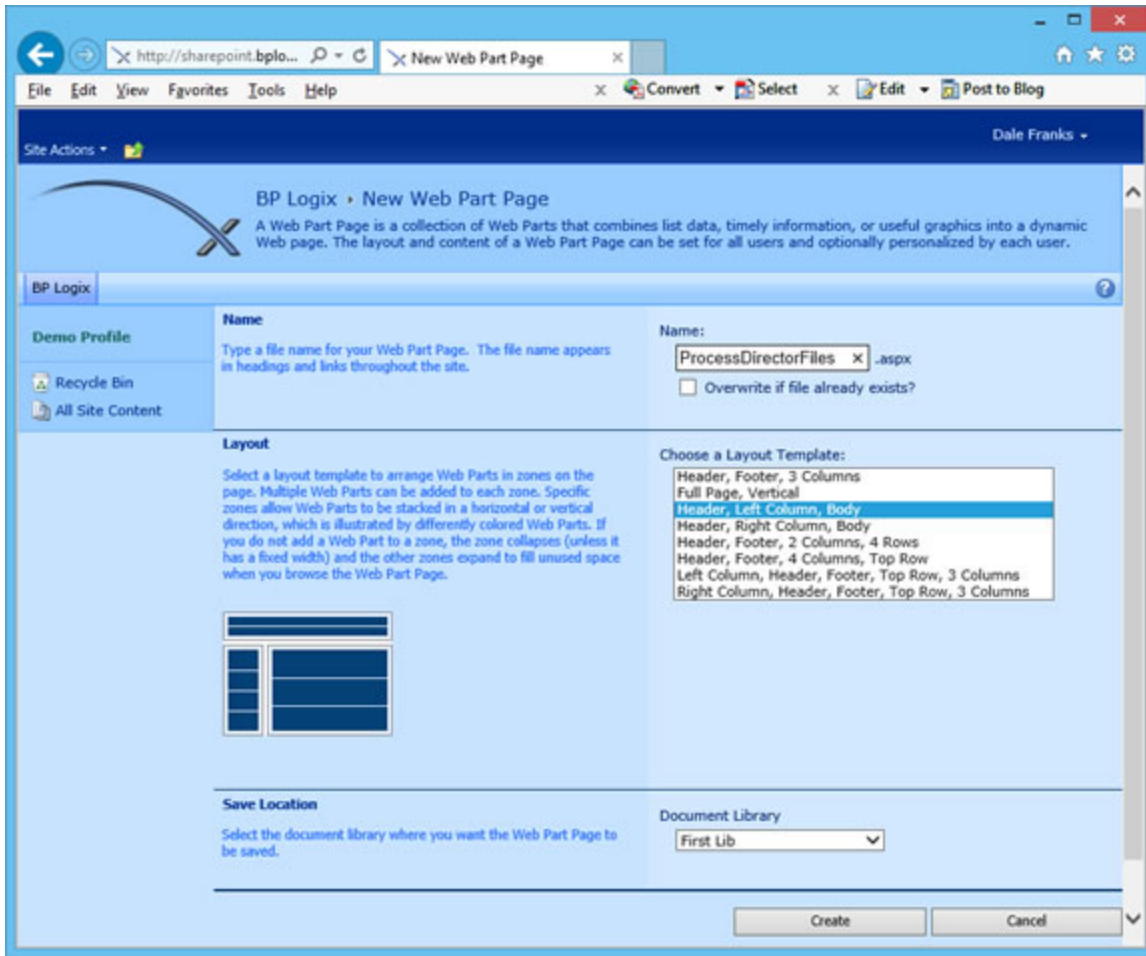
While Process Director has a number of Custom Tasks for working with SharePoint, getting and pushing documents between SharePoint and Process Director requires some work on SharePoint first. You'll need the "SharePoint Picker.htm" file uploaded to SharePoint and you'll need a web part page to house both it and the library with which you want to work. The "SharePoint Picker.htm" file can be obtained from the "/website/" directory under your Process Director installation directory.

One of the more convenient places to store the "SharePoint Picker.htm" file is the "Site Assets" library, as this is one of the standard SharePoint libraries, and its purpose is to house assets that may be required throughout the SharePoint instance. As such, it's a great place to store the file. So, once you have copied the "SharePoint Picker.htm" file, upload it to the Site Assets library.

Now, you'll need to build a SharePoint web parts page to which to attach the "SharePoint Picker.htm" file. To create this page in SharePoint, go to "Site Actions >> More Options". This will bring up the Create dialog box. From here select "Page" from the navigation menu on the left side of the form. This will display various types of SharePoint pages you can create. From the list of page type, click the "Web Part Page" icon, then click the "Create" button.

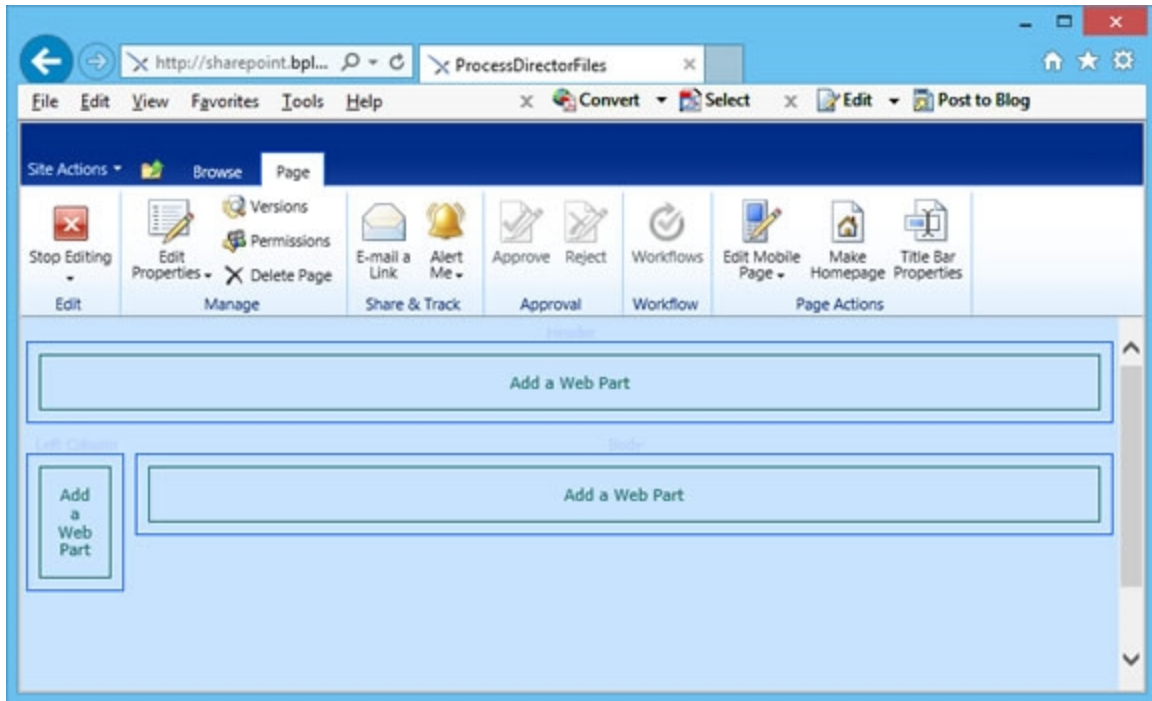


A configuration page will open. This page is where we will build our new web part page to house the “SharePoint Picker.htm” file and a document library to associate with it. Begin filling out the page by a file name for the new page in the “Name” text box. Then, select the “Header, Left Column, Body” style from the “layout” section. Finally select the document library where the page will be housed from the dropdown in the “Save Location” section. Once you’re finished, click the “Create Button. This will create your new, blank, web part editing page.



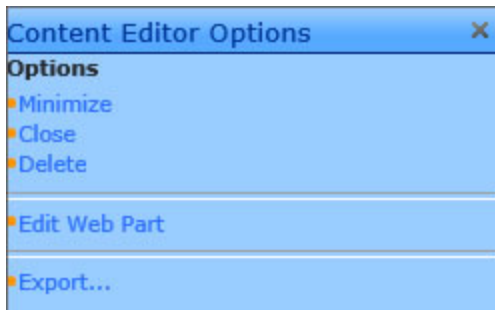
The editing page is where we will add the add web parts we need to configure a document library to work with Process Director. We'll need two web parts. The first will be the "SharePoint Picker.htm" file, and the second will be a document library. The document library will most likely be the same library where you saved the blank web part page in the previous step, but it may include a different library. For the purposes of this walk-through, we will use the same document library.



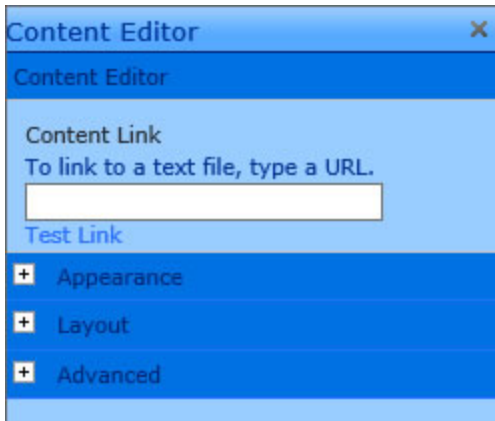


First, we need to add the web part that contains the “SharePoint Picker.htm” file. In the editing page, click on the top “Add a Web part” section. This will open the web part picker at the top of the page. Select “Media and Content” from the “Categories” section, then “Content Editor” from the “Web Parts” section. Click the “Add” button to create the web part. The web part will show up with only the “Content Editor” title showing.

We still need to point the content editor at the “SharePoint Picker.htm” file, so, if you look to the right side of the web part, you’ll notice a tiny, downward-pointing carat. Click it, and the Content Editor Options pane will appear.



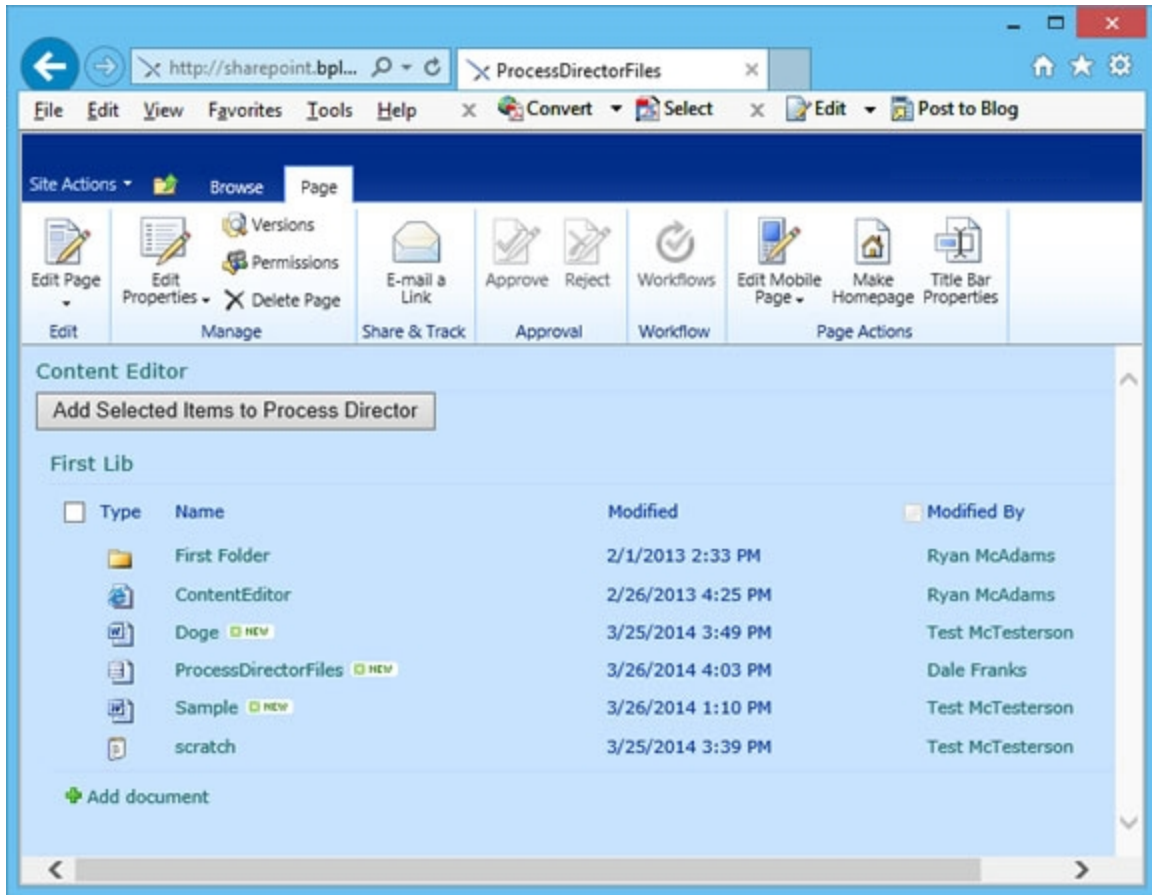
Click the “Edit Web part” link in this panel. The panel will change to show the editing pane.



In the “Content Link” text box, enter the path to the “SharePoint Picker.htm” file. Since we placed the “SharePoint Picker.htm” file in Site Assets for this walk-through, the path will be “/SiteAssets/SharePoint Picker.htm”. Click the “OK” button, and the “Add Selected items to Process Director” button will appear in the web part if you’ve entered the correct file path.

Now, add the web part to contain the document library by clicking the “Add a Web Part” box in the body section of the page. Once again, this will open the web part picker at the top of the page. Select “Lists and Libraries” from the “Categories” section, then select the document library you’d like to add from the “Web Parts” section. In the sample SharePoint setup we’re using for this walk-through, we’ve selected a library called “First Lib”. Click the “Add” button to create the web part. The web part will show up with only the “Content Editor” title showing.

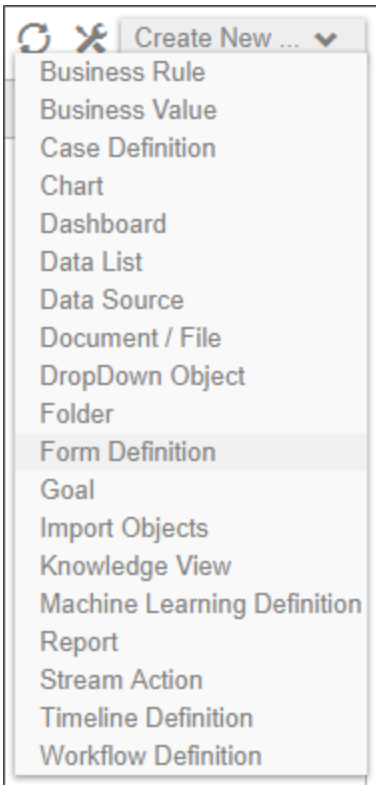
Now that we have a web parts page with the items we need, click the “Stop Editing” icon in the “Page” ribbon at the top of the SharePoint page. This will display the completed web parts page.



Make a note of the URL of the completed web part page. This is your Picker Page URL, and you'll need to use this URL to configure the Process Director Custom Task in the following sections.

## Building the Form #

The first thing we have to do is create a new, blank Form to handle our round-trip process. To create the Form, simply navigate to the Process Director folder where you'd like the Form to reside, Once you have selected the folder, create the Form by selecting "Form Definition" from the "Create New" dropdown located in the upper right corner of the Process Director window.

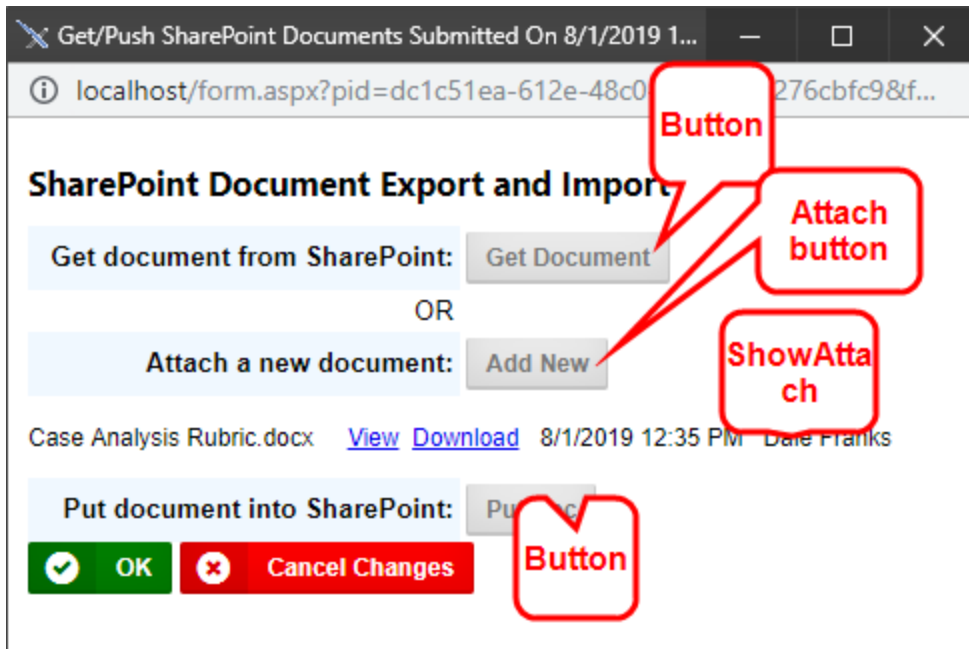


This will open the Form definition window. Leave the dropdown control that says “Use Online Form Designer” set to its default value. In the “Form Name” text box, enter the name for the new Form. You may also opt to enter a description of the Form’s purpose in the “Description” text box. BP Logix recommends as a best practice that you enter a description. This helps make it easier for others to see the Form’s purpose without having to open the form. For the purpose of this walk-through, we will name the form “Get/Push SharePoint Documents”, and fill out the description as shown below.

A screenshot of the 'Create Form' dialog box. The dialog has a title bar that says 'Create Form'. Below the title bar is a dropdown menu labeled 'Use Online Form Designer' with a downward arrow. Below that is a text box labeled 'Form Name' containing the text 'Get/Push SharePoint Documents'. Below the text box is a larger text area labeled 'Description' containing the text 'This form will get documents from a SharePoint library, allow document editing, then return the edited document to the SharePoint Document Library.' At the bottom of the dialog is a section labeled 'Options' with a downward arrow.

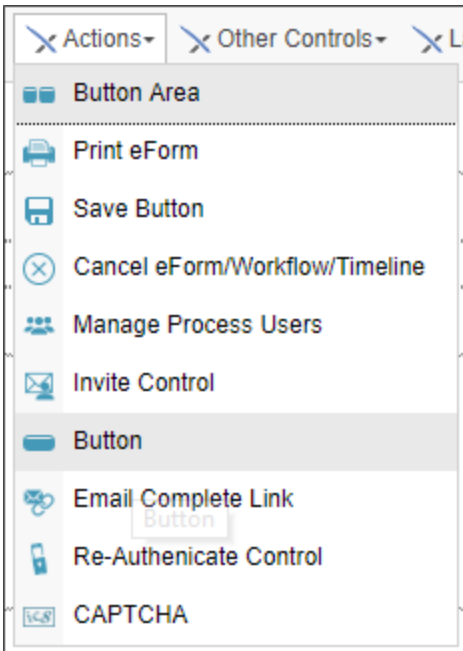
Once you have filled out the form definition fields, click the “OK” button to save your definition. You'll immediately be taken to the “Edit” tab of the Form’s definition to design the form using the Online Form

Designer. For this walk-through, we are going to build a form that looks something like the example below.

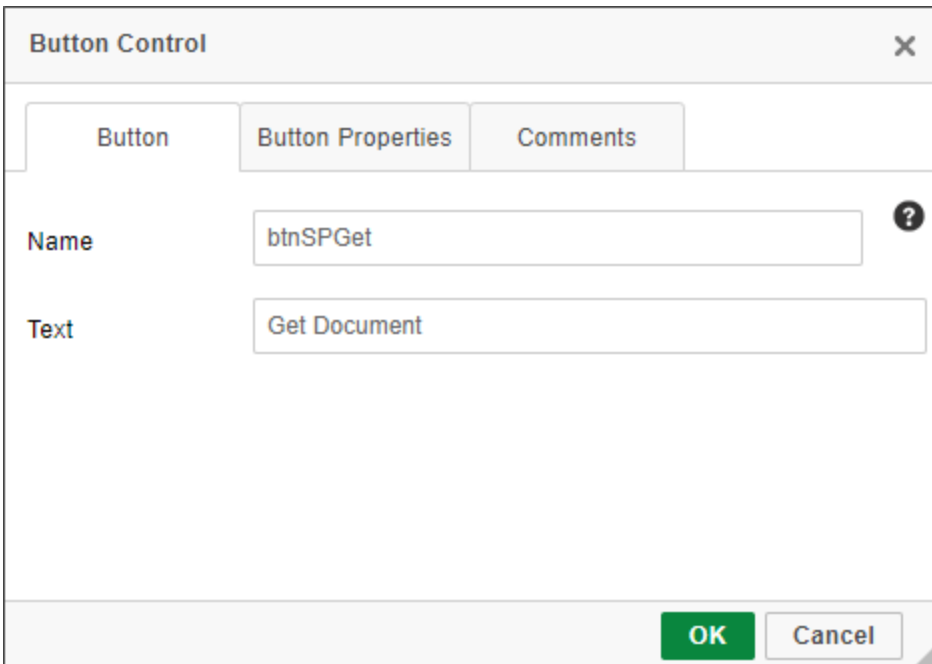


It's not important for the purposes of this walk-through that you take a lot of time to format the form template to look like the example above. What is important, though, is to see that to build the form, we are going to need to place four controls on the page: two button controls and two attachment controls. The two buttons at the bottom of the form that are labeled "OK" and "Cancel Changes" are placed on the finished form automatically by Process Director, so you can ignore those buttons for now. The rest of the template can be formatted however you'd like, just be sure to leave places to insert the four controls we will be adding.

Since we are going to need to add a couple of buttons to the form, we will need to select "Actions >> Button" from the menu to add each button.



Place the cursor on the page at the point you'd like the "Get Doc" button to appear, then select the button control from the menu. This will place a blank button on the Form template page at the location of your text cursor, and bring up the button's **Properties** dialog box.

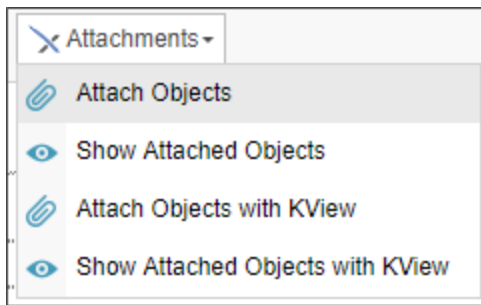


In the "Name" text box, type the name you want for this button. The name shouldn't contain any spaces, and it's best to use a name that logically identifies the button's function. In the "Text" text box, type the text that you'd like to appear as the button's label on the form. Once you're finished, clicking the **OK** button will close the dialog box and apply the settings to the button.

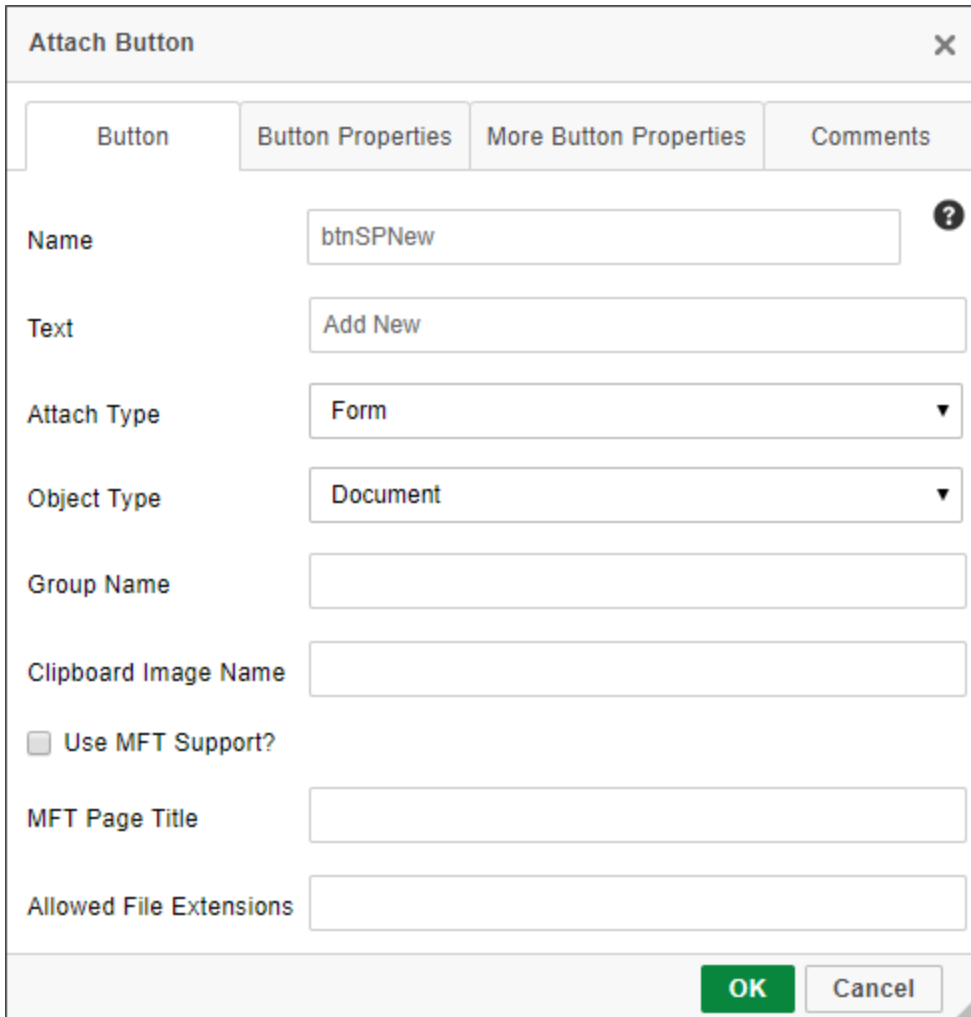
You'll need to repeat these steps once more to add the other button we will need. For the sample project we're using in this document, the following button properties are used:

BUTTON	NAME	TEXT
Get documents from SharePoint	btnSPGet	Get Doc
Put document into SharePoint	btnSPPut	Put Doc

Once the buttons have been placed on the template page, and the properties have been filled out, we must add the attachment controls. First, we will add the "Attach" button control that will allow us to attach a new document to the form, in case we'd like to upload a document into the SharePoint document library from our computer. Place your cursor at the desired place for the "Attach Objects" control, then select "Attachments >> Attach Objects".



Once the blank control has been added to the page, the [Properties](#) dialog box will open.



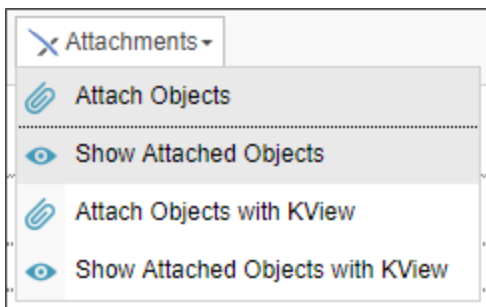
The "Attach Button" dialog box is shown with the "Button Properties" tab selected. It contains the following fields and controls:

- Name:** Text box containing "btnSPNew".
- Text:** Text box containing "Add New".
- Attach Type:** Dropdown menu set to "Form".
- Object Type:** Dropdown menu set to "Document".
- Group Name:** Empty text box.
- Clipboard Image Name:** Empty text box.
- Use MFT Support?:** Unchecked checkbox.
- MFT Page Title:** Empty text box.
- Allowed File Extensions:** Empty text box.

At the bottom right, there are "OK" and "Cancel" buttons.

On the properties dialog box, type in the appropriate values in the “Name” and “Text” text boxes, just as you did for the buttons. Set the “Attach type” to “Form”, and the “Object Type” to “Document”. When you are finished, click the **OK** button to close the dialog box and apply the properties.

Finally, we will need to add the control that displays the document attachments that will be associated with the form when we get them from SharePoint, or the new documents that we want to add to the SharePoint document library. Place your cursor in the spot where you’d like to display the document attachments control, then select “Attachments >> Show Attached Objects”. This will insert the “Show Attached Objects” control into the page.





Once, again, the control will be placed in the selected location, and the **Properties** dialog will appear.

Type in a logical name for the control in the “Name Dialog Box”. Since we won't need to use a Process Timeline for this sample, Change the “Attach Type” dropdown to “Form” in order to show objects attached to the Form, rather than to a process. Finally, check the “Show Edit” check box in the **Attach Properties** tab. This will show an edit link for the attached documents that will allow us to use Process Director’s check in/check out functionality to edit the attachments, which will provide us with the history and versioning functions for editing the attachment from Process Director. When you are finished, click the **OK** button to close the dialog box.

Our Form template should now be finished. Save and close the Form template, and go back to the form definition window.

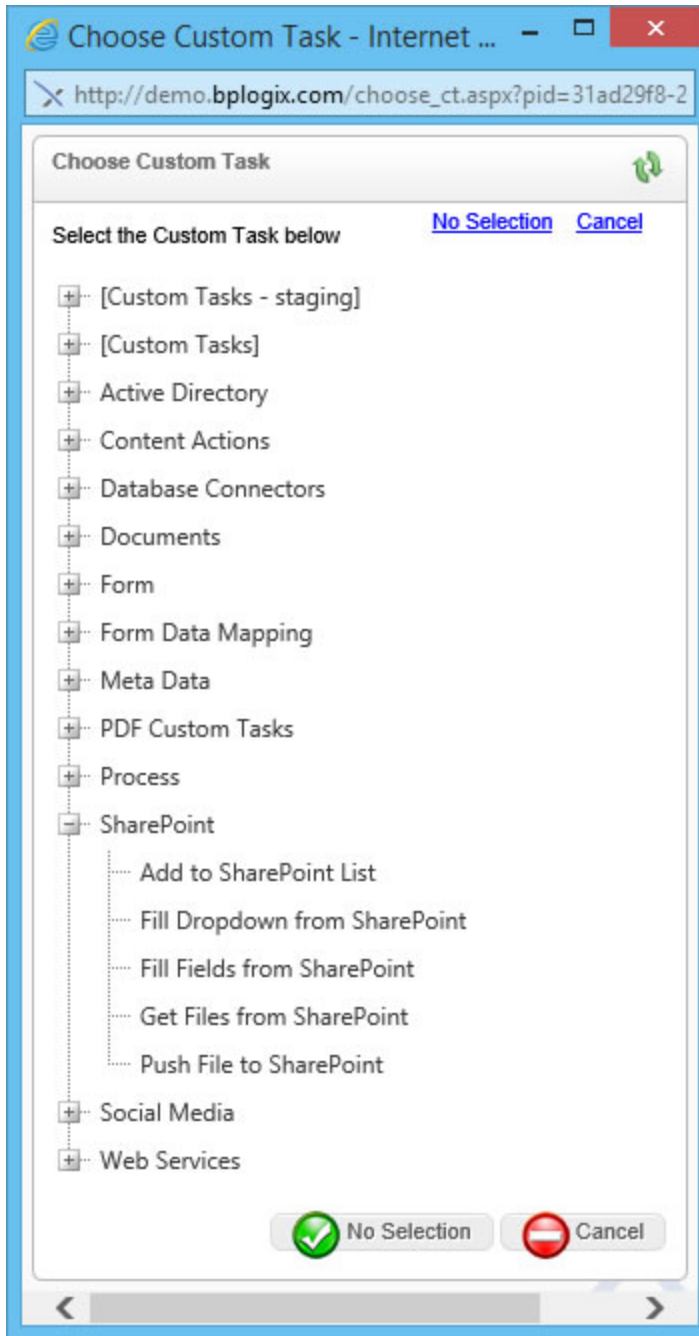
## Configuring Custom Tasks <#>

We'll need to set the SharePoint Custom Tasks for the form’s two buttons, so, from the Properties screen, click the **Custom Task Event Mapping** tab. From this tab, we will add the two SharePoint Custom Tasks that will be called by the buttons we placed on the Form template. When we are finished, the Custom Task setup in this tab will look much like the image below.

Event Name	Custom Form Task	Options	Condition
btnSPGet	Get Files from SharePoint	Configure	<a href="#">Click to create condition ...</a>
btnSPPut	Push File to SharePoint	Configure	<a href="#">Click to create condition ...</a>

At the moment, however, neither of the two Custom Tasks shown above appear on your screen, so let's add them.

First, click on the Build button at the top of the tab panel, which is labeled "...", and is placed to the immediate left of the "Add Custom Task" button. Clicking on the Build button will open the Choose Custom Tasks dialog box. This window displays all of the Custom Tasks that are available to you. For the first Custom Task, select "SharePoint", then select "Get Files from SharePoint".



Once you've made the selection of the Custom Task, the dialog box will close, and the text box to the left of the build button will now have the words "Get Files from SharePoint" displayed. You can now click the "Add Custom Task" button. This will add the first Custom Task to the [Task List](#).

To configure this task, we first have to select the button that will run the task. From the dropdown control in the "Event Name" column of the Custom Task list, select "btnSPGet". Next, click the button labeled "Configure" that is located in the "Options" column of the Custom Task list. This will open the configuration dialog box for the Custom Task.

### Get Files from SharePoint

Enables a picker for SharePoint files that will attach them to an object and optionally set Form data from Metadata fields.

[Click here for documentation](#)

**Site Details**

SharePoint Picker Page

SharePoint Connection [Select SharePoint Connection]

**References**    **Advanced**

Reference Type:  Process     Form

**Mappings**

Here, you may indicate data transfer between Form data and a SharePoint Item's Metadata. To specify a Metadata field, enter its name in the text box.

Add Mapping

Custom Task Version: 2014.11.04

In the “Site Details” section’s “SharePoint Picker Page” text box, type in the Picker Page URL. This is the URL of the Picker Page you created in the first section of this walk-through. This is the SharePoint page that will display the document library from which you’ll select documents to attach to the form. Next, select the SharePoint instance you want to use from the “SharePoint Connection” dropdown. Finally, in the “References” section, select the “Form” option button in the “Reference Type(s)” option list. Click “OK” to close the dialog box and apply your settings. The first Custom Task is now configured.

To add the next Custom Task, click the Build button again. This time, select the “Push File to SharePoint” Custom Task from the Choose Custom Tasks dialog box. This will close the dialog box and add the “Push File to SharePoint” into the text box next to the Build Button. Click the “Add Custom Task” button to add another row to the Custom Task list.

On the second row of the Custom Task list, select “btnSPPut” from the dropdown control in the “Event Name” column. Then click the Configure button to open the dialog box for the Push File to SharePoint Custom Task.


### Push File to SharePoint

Allows the transfer of Documents/Attachments to a SharePoint Folder or List.

[Click here for documentation](#)

**Site Details**

SharePoint Connection


 **Connect**

**References**    **Advanced**

Reference Type(s):  Workflow References     Timeline References     Form References

**Query Details**    **Advanced**

SharePoint List   Create Folder if it doesn't already exist

 **Verify SharePoint Data**

**Mappings**

Here, you may indicate data transfer between Form data and a SharePoint Item's Metadata. To specify a Metadata field, enter its name in the text box. **Add Mapping**

Custom Task Version: 2019.02.15

In the “Site Details” section’s “SharePoint Connection” dropdown, select the SharePoint instance where the document will be pushed. Then, select the “Form References” reference type from the references section. Next chose the list you want to use from the “SharePoint List” dropdown. You can also type in the path to the list if you’d like. This also allows you to create a new path by typing the path into the text box, then checking the “Create Folder if it doesn’t already exist” check box. Process Director will create the path location in SharePoint for you automatically. Now, click the “Advanced” tab for the Query Details section. Set the “Action if File exists” dropdown to “Check in new version”, and the “Handling Documents with SharePoint links” dropdown to “Update SharePoint Document, if available”. Once you have done this, click the “OK” button to close the dialog box and save your configuration settings for the Custom Task.

You should now be back at the Properties Pane for the Form Definition. Click the “Update” button to save the configuration changes you’ve made. Once you have done so, you’re now ready to test your form.

## Running the Form #

Run the form by clicking the “Run” button located at the top right of the Form definition page. This will open your newly-completed Form.

### SharePoint Document Export and Import

Get document from SharePoint:

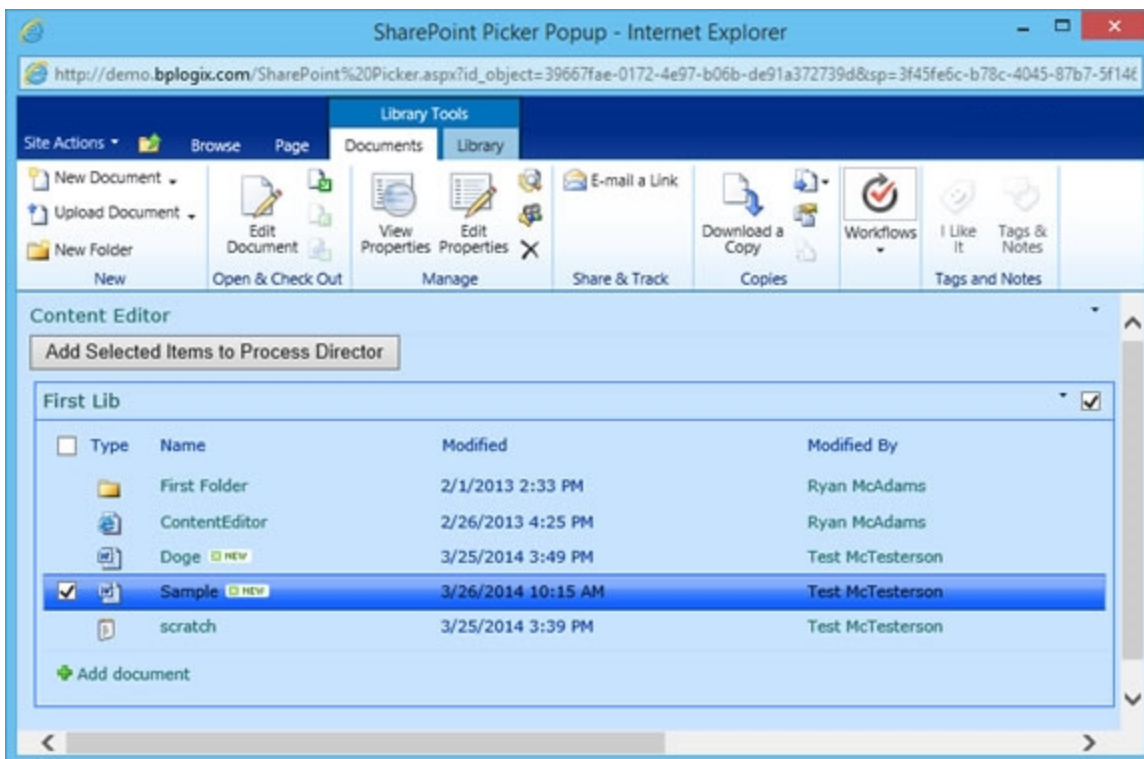
OR

Attach a new document:

Put document into SharePoint:

## Doing a Quick Round-Trip of a Document

Notice that the “Show Attached Objects” control is hidden. It won’t display until you’ve actually attached a document to the form, so let’s attach one by clicking the button labeled “Get Doc”. This will open the SharePoint Picker Page you specified in the configuration for the Custom Task.



From the picker page, select a document to import into Process Director by selecting the check box on the left side of the document’s row in the document list. See the image above for an example of a selected document. Once you’ve selected the document you want, click the button labeled “Add Selected Items to Process Director”. This will transfer the document to Process Director, and the “Show Attached Objects” control will appear, showing the document you’ve attached.

### SharePoint Document Export and Import

Get document from SharePoint:

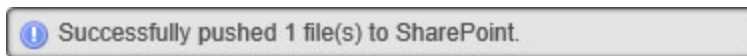
OR

Attach a new document:

Sample.docx [View](#) [Download](#) 8/1/2019 1:04 PM Dale Franks

Put document into SharePoint:

At this point, you can send the document right back to SharePoint by simply clicking the “Put Doc”. If the push is successful, a success notice will be displayed at the top and bottom of the form.



Of you were to go into SharePoint and view the documents history, you'd see that Process Director not only pushed the document back to SharePoint, but the document has also been versioned in SharePoint as well.

Click “OK” to close and save the Form.

### Adding a New Document

You can also add a brand new document to SharePoint by opening a new instance of the Form, then clicking the “Add New” button. This will open the standard File dialog box from which you can browse through your file system to select the document you want to push to SharePoint. Once you have selected the document, it will appear in the “Show Attached Objects” control.

Once again, you can click the “Put Doc” button to push the new document to SharePoint.

### Editing a Document from the Form

Once you have a document attached to the Form, you can edit it from Process Director, using Process Director’s built-in checkout and versioning system. There is an “Edit” link for each document in the “Show Attached Objects” control. If you click the “Edit” link Process Director will open the Edit Document dialog box.

Document Name Icon

---

### EDIT

To edit the document, you must first check it out. This will prevent other users from editing the document.

Check Out and Edit

This action will check out the file, then automatically download and open the document on your PC.

Check Out

This action will check out the file. You can then download the file to any location on your PC.

This process is identical to the process for checking out and editing the Form template we created earlier in this document. You can check the document out, edit it, and check it back in. Once the document is checked back in, the edited document is now the version attached to the Form.

Once again, you can push the edited document back to SharePoint by clicking the “Put Doc” button on the Form.

## Process Director Custom Utilities

For Process Director v5.43 and higher, a set of custom utilities is available from the Downloads section of the [BP Logix Support site](#) as a downloadable ZIP file that contains an import of all the custom utilities. These custom utilities include an assortment of processes and reports intended to assist Process Director process and system administrators. When importing the custom utilities, extract the PDZ file and import it into the root folder of your Process Director Installation.

The initial import will create a [Custom Utilities] folder in the root of the partition, which contains the new utilities. Subsequent imports will overwrite the [Custom Utilities] folder with updated versions.

Each subfolder in the [Custom Utilities] folder contains a different utility with its own instructional PDF. The following custom utilities are included:

- **E-mail Templates Link Report:** Enables you to select a Process Timeline to see which e-mail templates it or its activities uses, or to select an Email Template to see which Process Timeline and activities use it.
- **Find Tables by Column Name:** Searches your Process Director database by a full or partial column name and displays a report of any tables or SQL views containing a column with that name.
- **Form Instance Counts Report:** Generates a report on the number of form instances submitted, with counts grouped by form definition and month. This report can be filtered by month, year, and user.
- **Import Users:** This customizable process enables you to import users into Process Director using a CSV file generated by an external system. It includes a sample file and instructions on how to customize the process. This utility will add and update users as required based on the CSV content.
- **Permissions Report:** This report displays the current permissions set on user-facing objects in Process Director. It can be filtered by object type and permission type.
- **Process in Error:** This configurable utility runs daily and sends an e-mail notification to the configured user(s) when any process is in error. This utility reduces the chances of errors being overlooked by process or system administrators.
- **Processes by Group Report:** Generates a report for a selected User Group of any processes where that group is explicitly assigned to a timeline task.
- **SQL Views and Business Values Report:** Generates a searchable report of all SQL Views in the system, and the Business Value queries that utilize them. Selecting an existing SQL View enables you to see its fields and associated Business Values.
- **Task Duration Report:** Generates a chart of the average duration to complete tasks, as well as a report that displays details on individual user tasks. This report is searchable by Process Timeline, Activity name, and date range to see how long user tasks are taking to complete tasks.
- **User Task Completion Report:** Enables you to select a user and date range to see a report of which tasks the user has completed during the selected date range. The report includes the task result and **Routing Slip** comments.



## Using the bpUtil Utility

Process Director provides a utility, bpUtil, for passing commands to Process Director, usually on a scheduled basis. For customers, nearly all use cases only use a single command, `SU`, which calls a URL. There are several reasons why you might wish to do this, such as scheduling an Active Directory synchronization, or scheduling a Knowledge View to export an Excel or CSV file, or to run a process for each row of the Knowledge view.

You should use bpUtil to run all scheduled commands on Process Director. ***You should not call a Process Director URL directly from the Windows Scheduler Utility.*** The Windows Scheduler will not release the page after running, and keep it in memory, which can cause performance issues. bpUtil, on the other hand, will call the page and release it immediately after it runs. This utility was specifically constructed for this reason.

So, instead of calling the page directly through the Windows Scheduler, you would use Windows Scheduler to call bpUtil, passing the `SU` command and the fully qualified URL as a parameter. For example to schedule an Active Directory synchronization, you would place the following command into the Windows Scheduler:

```
"LocalPath\bputil.exe" SU "http://localhost/WD/admin/ad_sync.aspx?ads=ProfileName"
```

...Where `LocalPath` is the drive path on the local computer where bpUtil is located, and `ProfileName` is the name of the actual AD Synchronization Profile to run when the page is called.

This command calls bpUtil from the Windows Scheduler. bpUtil, in turn, will call the page to run the appropriate synchronization profile, then release the page from memory when the synchronization begins.

The appropriate Windows Scheduler command used to call bpUtil is provided in the documentation for each Process Director object to which it is relevant.

Like the `bpImport` or `bpEmailImport` utilities, bpUtil can be installed on any computer. Unlike the other utilities, bpUtil does not have a graphical user interface. It is generally only addressed via Windows Scheduler commands.

## Using the bpImport Utility

Process Director provides a utility for importing or exporting documents automatically between Process Director and file system folders, named "BPImport". The utility doesn't import/export any [Content List](#) items other than documents, so Forms, Knowledge Views, etc., aren't affected by the bpImport utility.

BPImport makes it easy to import documents from a file system when moving from a file based system to Process Director as your document management system. This utility can be run interactively with a dialog prompting for input fields, or scheduled to perform an automatic import on a scheduled basis. The import or export can be run from the machine on which Process Director is installed, or from a remote computer. You must enter credentials for a user that has full permissions to the folder in which to import/export.

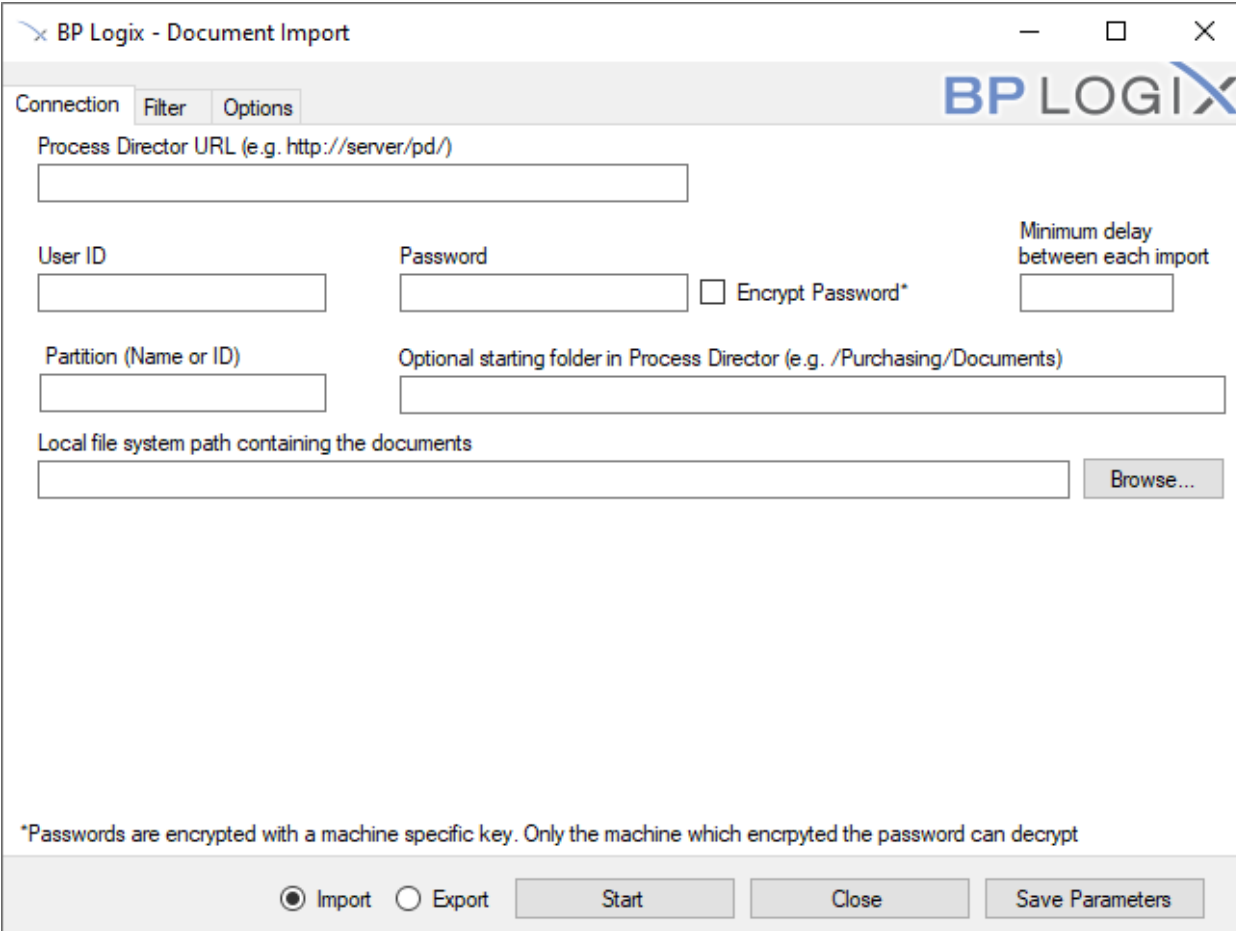
The executable file, `bpImport.exe`, is located in the `c:\Program Files\BP Logix\Process Director\` directory by default. You can copy this utility to any computer that has Internet Explorer 6 or higher installed. Simply execute `bpImport.exe` to launch the dialog, and press "Start" to begin the import/export process. The status of the import is displayed on the dialog.

Cloud installation customers can download the utility from the Downloads section of the [BP Logix support site](#).

The bpImport utility has a tabbed interface, and the fields in the interface are generally the same whether you wish to import or export documents.

## Connection Tab #

The connection tab creates the connection between Process Director and your file system.



The screenshot shows a window titled "BP Logix - Document Import" with a tabbed interface. The "Connection" tab is active, showing fields for "Process Director URL (e.g. http://server/pd/)", "User ID", "Password", "Encrypt Password\*" (checkbox), "Minimum delay between each import", "Partition (Name or ID)", "Optional starting folder in Process Director (e.g. /Purchasing/Documents)", and "Local file system path containing the documents" with a "Browse..." button. At the bottom, there are radio buttons for "Import" (selected) and "Export", and buttons for "Start", "Close", and "Save Parameters". A footer note states: "\*Passwords are encrypted with a machine specific key. Only the machine which encrypted the password can decrypt".

### Process Director URL

The complete URL to the server running Process Director.

### User ID

The User ID to use for the import routine. This user must be a system administrator. This User ID is designated as the document owner for the new documents.

### Password

The password to use for the import routine. This user must be a system administrator.

### Encrypt Password

For bplImport v2.21 and higher, this property enables you to encrypt the password in the command line command. The encryption uses a machine-specific key, so the password can only be decrypted on the machine on which the encryption was performed. So, a bplImport command that uses an encrypted password must be run on the machine on which it was created. If the command runs on a different machine, the password can't be decrypted when passing the command to the Process Director server.

### **Minimum delay between each import**

Specifies an optional amount of time in milliseconds to delay between each file imported/exported. The default is no delay. This is only needed if you want to give the server time to respond to other requests when many documents are being imported at once.

### **Partition (Name or ID)**

Enter the Partition name or ID to use when importing.

### **Optional starting folder in Process Director**

Specifies the starting folder in the Process Director database. If this field isn't specified, new documents will be stored in the root folder.

### **Local file system path containing the documents for import (Shown when importing)**

Specifies the starting location to the documents to be imported. This can specify a network path, or a file system path. The utility will import all files in the folder, and will traverse through all subdirectories as well. The relative folder structure will be maintained in Process Director.

### **Local file system path to which the documents will be exported (Shown when exporting)**

The file path on the local computer into which all of the exported files will be placed.

### **Import/Export**

Radio buttons to select whether you wish to import documents into Process Director, or export documents from Process Director.

### **Filter Tab <#>**

The Filter tab enables you to specify the criteria for the documents you'd like to import/export.

The screenshot shows a window titled "BP Logix - Document Import". It has three tabs: "Connection", "Filter", and "Options". The "Options" tab is selected. The window contains the following fields and controls:

- Matching criteria of documents to import (e.g., .pdf, .doc) (optional):** An empty text input field.
- Maximum # of documents to import (optional):** An empty text input field.
- Maximum Size (may use prefixes for large values, e.g. "20 k", "1.5 M") (optional):** An empty text input field followed by "B (bytes)".
- Minimum File Age (in seconds) (optional):** An empty text input field followed by "s".
- Sort Files By:** Three radio buttons: "Name" (selected), "Time Created", and "Time Last Modified".
- Sort In:** Two radio buttons: "Ascending Order" (selected) and "Descending Order".
- Bottom controls:** Radio buttons for "Import" (selected) and "Export", and buttons for "Start", "Close", and "Save Parameters".

### Matching criteria of documents to import/export

Specifies a filter that is matched against the file name before it is imported/exported. This can be used, for example, to only import documents that have a .pdf extension. If this field isn't specified, all documents will be imported.

### Maximum # of documents to import/export

Specify an optional maximum number of documents to import/export.

### Maximum Size

Specify an optional maximum number of bytes for a document. Documents larger than this number of bytes on the local file system will be skipped. (Note: a kilobyte is 1,024 bytes, a megabyte is 1,048,576 bytes, and a gigabyte is 1,073,741,924 bytes.)

### Minimum File Age

Enables you to specify the number of seconds since the file was created. Documents newer than the number of seconds specified will be excluded from processing.

### Sort Files By

This option enables you to determine the order in which files are imported/exported. You can select from one of the following options:

- Name
- Time Created
- Time Last modified

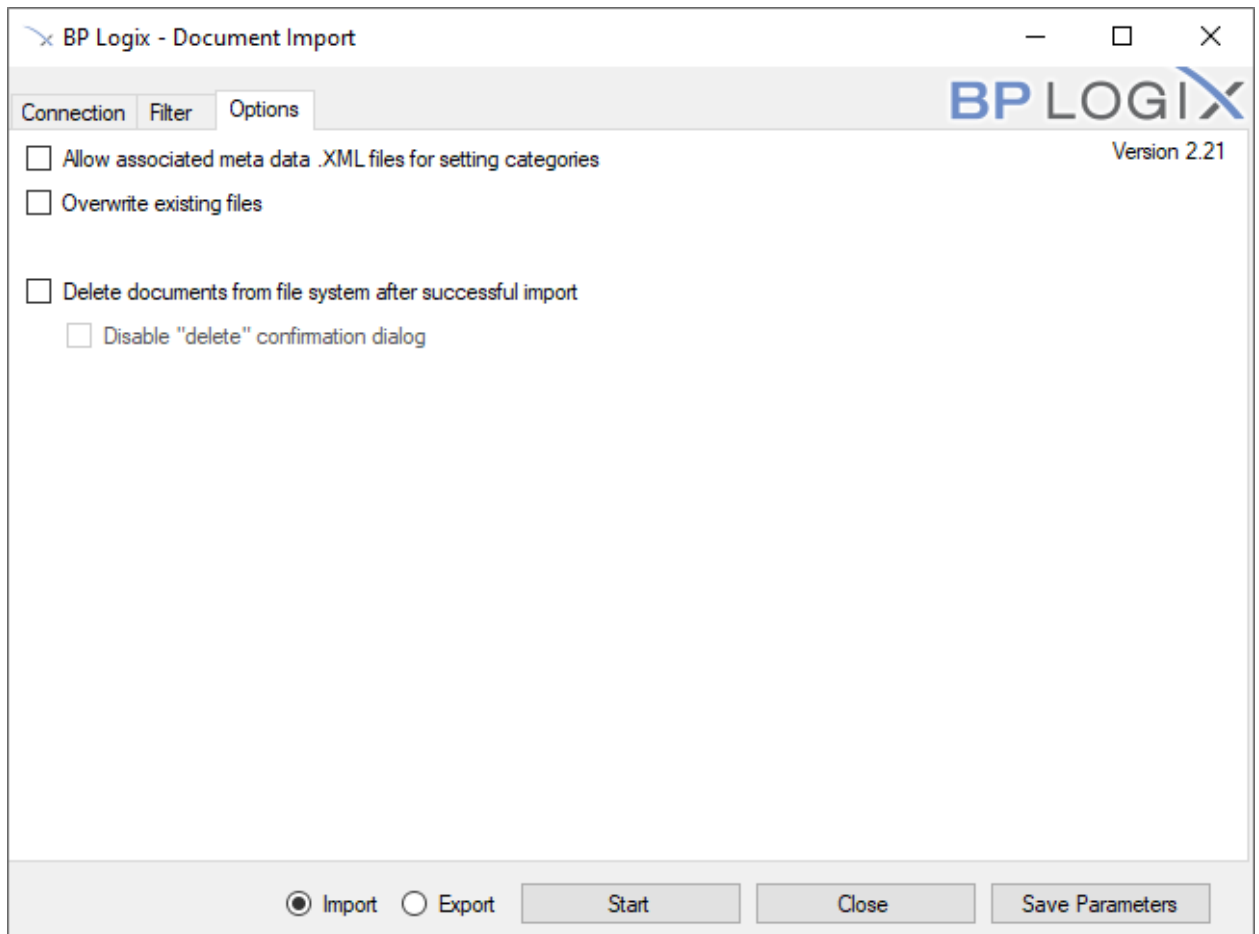
The default order is alphabetical sorting by name.

### Sort In

You can set the import to sort in Ascending or descending order.

### Options Tab (Import) #

When the Import radio button is selected, the Options tab enables you to specify some additional options when importing.



#### Allow associated Meta Data .XML files for setting categories

If this is selected it will attempt to find matching file names with “.XML” appended to the name. These files contain XML Meta Data that can be used to set categories and attributes of the document in Process Director. Refer to the section below regarding the format of the XML file.

### Overwrite existing files

If selected, this will overwrite existing files on the Process Director server. This option is unchecked by default.

### Delete documents from file system/Content List after successful import/export

If selected, this option will remove files from your file system after they are imported, or from the [Content List](#) after they are exported. Files will be deleted only after they are successfully imported into the Process Director database. This option can be used, for instance, to create a drop folder for documents to automatically be imported based on a schedule. Users and/or applications can drop documents into a file system folder, and the scheduled import utility will import and remove them from the folder.

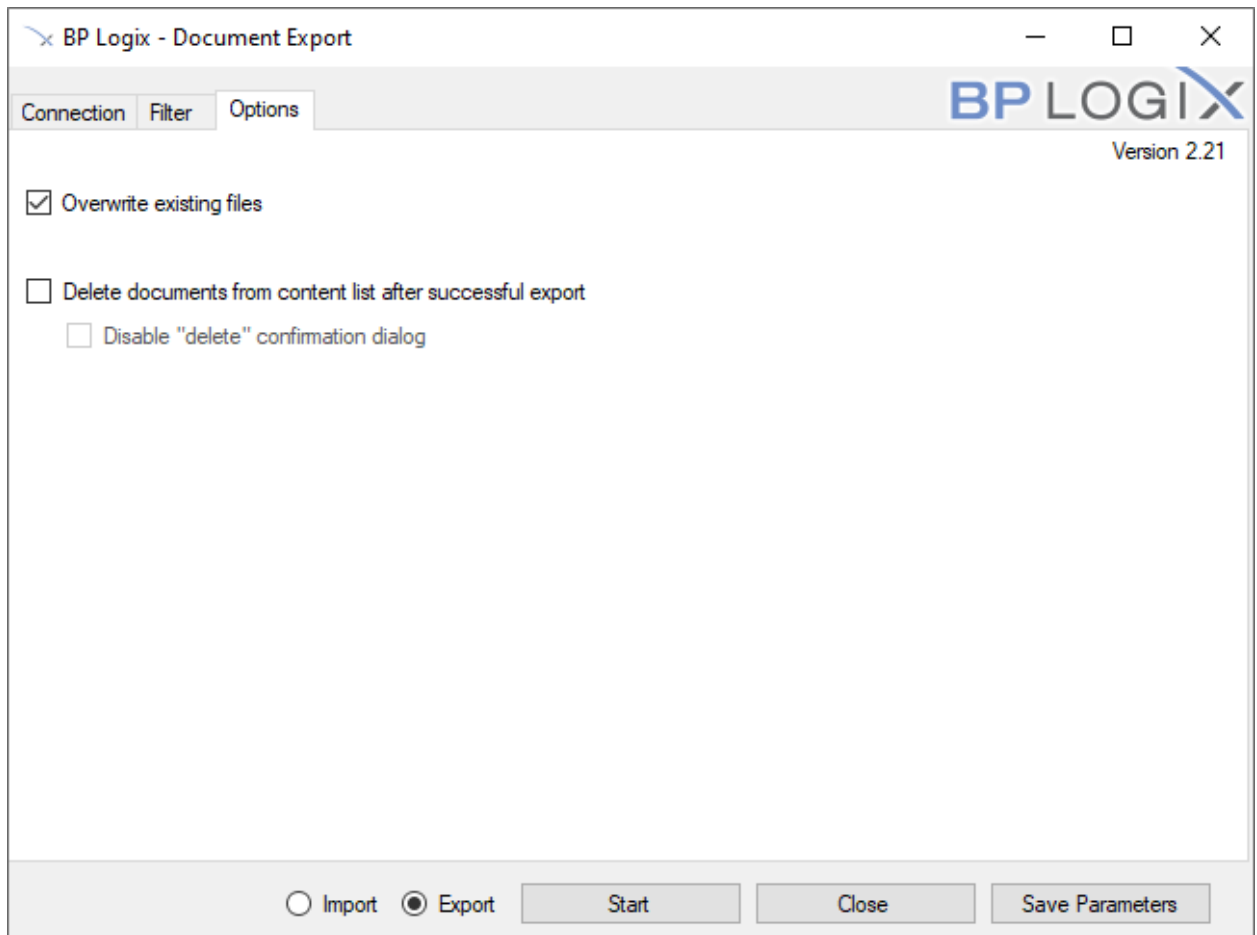
### Disable "Delete" confirmation dialog

If selected, there will be no confirmation dialog for every file that is deleted. If this option isn't selected, you'll be prompted before every file is removed.

### Options Tab (Export) #

When the Export radio button is selected, the Options tab enables you to specify some additional options when exporting.

When exporting items from the [Content List](#), the folder structure of the [Content List](#) will be replicated in the local file system during the export.



### Overwrite existing files

If selected, this will overwrite existing files on the file system. This option is selected by default.

### Delete documents from Content List after successful Export

If selected, this option will remove files from the [Content List](#) in Process Director after they are exported. Files will be deleted only after they are successfully exported into the file system.

### Disable "Delete" confirmation dialog

If selected, there will be no confirmation dialog for every file that is deleted. If this option isn't selected, you'll be prompted by Process Director before every file is removed.

## Scheduling File Imports (Smart Folders) <#>

You may want to automatically schedule the import allowing for Smart Folders. This allows users to drop documents in a folder on a file system and have it automatically imported in to Process Director with the same folder structure used on the file system. This can automatically start a process for each new document added. To facilitate this you'll schedule the non-interactive import command using the command line options. The import GUI has a button called "Save Parameters" which is used to save all of the current settings in the dialog into popup dialog for copying to the clipboard or save as a batch file. The com-

mand line options will specify the fields documented above. Here is the list of command line options available, and the mappings to the dialog fields:

**Properties**

Parameter Name	Description	Alternate Name
directory PATH	Import files from PATH	-d
delay NUMBER	Delay each import by NUMBER milliseconds	
delete	Delete files after successful import	
folder PATH	Import files into Process Director folder at PATH	-f
filter FILTER	Limit import to only files of types in FILTER (comma-separated extensions)	
partition NAME	Import into partition NAME	-i
maxnum NUMBER	Limit the number of files to import to NUMBER	-m
metadata	Include metadata with import files (in corresponding XML files)	
no-gui	Run in non-interactive mode (required for scheduling)	-n
overwrite	Overwrite files on import with the same name already in Process Director	
password PASSWORD	Connect to Process Director as user with password PASSWORD	-p
user USERNAME	Connect to Process Director as USER	-u
webserviceurl URL	Connect to Process Director at URL	-w
Log PATH	Folder path into which log files will be written	-l

Running in non-interactive mode requires the following arguments:

`-n, --webserviceurl, --partition, --user, --password, --directory`



### Example

To run the import utility and delete files after a successful import:

```
bpImport.exe -w "https://hostname/" -u "user1" -p "" -i "Test Partition" -d "C:\Windows\Temp" -n -delete
```

To run the import utility with Meta Data and a 50ms delay after every import:

```
bpImport.exe -n -w "https://hostname/" -u "user1" -p "" -i "Partition" -d "C:\Windows\Temp" --metadata --delay 50
```

When running the import utility non-interactive from a command line or scheduler, any errors will be written to a file named bpU.log in the Logs directory where the bpImport.exe exists.

## Optional Meta Data XML Files #

The import utility can include meta data with each of the files being uploaded to Process Director. This meta data can set the categories and attributes for the document on the server. This occurs if the option is selected to allow associated meta data files on the import dialog. If this option is selected the import utility will attempt to find matching file names with “.XML” appended to the name. For example, if the import utility finds a file named mydoc.pdf, it will attempt to locate the mydoc.pdf.xml file for the meta data. The meta data XML file must be of a certain format to be recognized as valid meta data. The XML file must contain the following structure and tags.

```
<META_DATA>
  <INPUT_META_DATA>
    <META_NAME>external_attribute_name</META_NAME>
    <META_VALUE>attribute_value</META_VALUE>
  </INPUT_META_DATA>
  <INPUT_META_DATA>
    <META_NAME>external_attribute_name_2</META_NAME>
    <META_VALUE>attribute value 2</META_VALUE>
  </INPUT_META_DATA>
</META_DATA>
```

The <META\_NAME> must match an External Name in a category attribute for the document to be assigned to that category. If a matching External Name isn't found in the category tree the meta data is discarded after the upload completes.

For example, assume you have a category named Operations that contains a subcategory named Quality with an attribute name of ActionManager. You can assign this entire category tree (i.e. Operations.Quality) to the imported document by using a matching External Name in the ActionManager attribute. If the external name in the ActionManager attribute is set to “manager\_name”, the XML for the imported document would look as follows:

```
<META_DATA>
  <INPUT_META_DATA>
    <META_NAME>manager_name</META_NAME>
    <META_VALUE>John Smith</META_VALUE>
  </INPUT_META_DATA>
</META_DATA>
```

To set values for checkboxes, the values YES and NO correspond to a checked state and an unchecked state, respectively.

Date values vary based on locale: if you're in the USA, use mm/dd/yyyy. If you're elsewhere, use dd/mm/yyyy.

## Using the bpEmailImport Utility

The bpEmailImport utility (bpEmailImport.exe) allows you to send formatted emails to a process either on a schedule or any time an email is received. Using bpEmailImport.exe, you can set the conditions and format under which you'd like to import the emails.

The executable file, bpEmailImport.exe, is located in the c:\Program Files\BP Logix\Process Director\ directory by default. You can copy this utility to any computer that has Internet Explorer 6 or higher installed.

Cloud installation customers can download the utility from the Downloads section of the [BP Logix support site](#).

## Connection Tab #

BP Logix - Email Monitor

Connection Filtering Options

BP LOGIX

Process Director Information

Process Director URL (e.g. http://server/pd/)

User ID Password  Encrypt Password\* Test PD Credentials

Partition in which to import Optional starting folder in Process Director (e.g. /Purchasing/Documents)

Mail Server Information

Server Address (e.g. pop.gmail.com) Server Protocol Server Port  Use SSL

Username Password  Encrypt Password\* Test Server Credentials

(IMAP only) Folder to Monitor

\*Passwords are encrypted with a machine specific key. Only the machine which encrypted the password can decrypt

Start Close Save Parameters

### Process Director URL

Enter the URL of the Process Director here.

### User ID and Password

Enter your User information in order to let bpLogixImport.exe log onto your Process Director. Click the [Test PD Credentials](#) button to see if the information you entered was valid.

### Encrypt Password

For bpEmailImport v2.21 and higher, this property enables you to encrypt the password in the command line command. The encryption uses a machine-specific key, so the password can only be decrypted on the machine on which the encryption was performed. So, a bplImport command that uses an encrypted password must be run on the machine on which it was created. If the command runs on a different machine, the password can't be decrypted when passing the command to the Process Director server.

### Partition in Which to Import

Enter the name of the partition that you wish to import emails to.

### Optional Starting Folder in Process Director

You can import the emails to a specific folder in the partition, using this field to specify the path.

### Server Address

This is the host name of the server sending your email.

### Server Protocol

Select the mail server type for your email server, i.e., either POP3 or IMAP.

### Server Port

Enter the port you use to connect to your email server. By default, the server port is 110, however if you connect to your server using a different port, enter it there.

### Use SSL

You can use a secure protocol to retrieve your email by checking this box.

### Username and Password

Enter your Username and Password for the email account from which you wish to retrieve incoming emails. Click the [Test Server Credentials](#) button to verify that your input was valid

### Folder to Monitor


If your mail server is an IMAP server, you can specify that the utility map a specific folder in the email account to monitor.

### OAuth Tab #

Version 2.22 of the utility has an [OAuth](#) tab to configure OAuth authentication for mail services that use Microsoft's Modern Authentication system to access the service. The properties on this tab enable access to Azure, and are required *in addition* to the properties on the [Connection](#) tab. While the [Connection](#) properties enable connection to a specific email inbox, the [OAuth](#) properties grant access to Microsoft Azure services.

The screenshot shows a window titled "BP Logix - Email Monitor" with a standard Windows title bar. The window has a tabbed interface with three tabs: "Connection", "OAuth", and "Filtering", with "OAuth" currently selected. The "Options" tab is also visible. The "BP LOGIX" logo is in the top right corner. Below the tabs, there is a section labeled "Client" containing three text input fields: "Tenant:", "Client ID:", and "Client Secret:". At the bottom of the window, there are three buttons: "Start", "Close", and "Save Parameters".

There are three fields that must be configured to enable OAuth authentication with the mail service. Use of OAuth from Microsoft with the E-mail Import Utility requires the existence of a properly configured Azure Active Directory (AAD) Registered App.

 Creating the AAD Registered App and integrating it with OAuth will require in-depth knowledge of Azure. As such, assistance from your IT personnel may be required.

Microsoft OAuth integration requires three pieces of data that correspond to the three properties that must be configured on the **OAuth** tab. These three properties must be obtained from Azure, and can be accessed only after creating the AAD Registered App for the OAuth connection.

The **Tenant**, **Client ID** and **Client Secret** properties are obtained as follows:

#### Tenant

- The ID of the Azure Tenant in which the AAD Registered App resides
- Creation of an AAD Registered App requires the existence of a Tenant
- The Tenant ID will follow login.microsoft.com/... in the Endpoint URLs that the App references

#### Client ID

- The ID of the AAD Registered App

#### Client Secret

- The secret or application password the administrator created to use with the AAD Registered App

Additionally, the AAD Registered App requires the following settings:

### API Permissions

- Delegated Microsoft Graph permission IMAP.AccessAsUser.All
- The administrator must grant consent to this permission after its addition

### Authentication Settings

- Allow public client flows: Yes (on)
- Organizations that use a hybrid Federated Identity environment must enable [Azure Active Directory Pass-Through Authentication](#). Otherwise, the utility will be unable to connect to the specified user accounts, and an error will be generated.

A more detailed explanation of how OAuth works and is configured on Azure-based systems is available in the [SharePoint OAuth Datasource topic](#).

### Filtering Tab #

#### Maximum # of Emails to Process

Enter a number in this field to set the maximum number of emails able to be imported at once.

#### Maximum Age of E-mails to process

You can tell bpEmailImport.exe to only import emails up to a certain age (in seconds, by default).

## From Filter (Regular Expression)

You can use regular expressions to filter which emails will be imported by the sender of the email.

## Subject Filter

You can use regular expressions to filter which emails will be imported by their subjects.

## Body Filter

You can use regular expressions to filter which emails will be imported by the content of their bodies.

## Minimum Delay between each Import

You can set a delay between each email imported to reduce server traffic.

## Options Tab #

BP Logix - Email Monitor

Connection Filtering Options

BP LOGIX

Version 2.21

After successfully processing e-mail messages

Leave e-mail messages on server (select file to keep track of e-mails)

Move e-mail messages to folder (IMAP only)

Delete e-mail messages (not recommended for IMAP)

Maximum Attachment Size (optional):  B (bytes)

Keep Attachments in Message object  Add Attachments as References to Message object in Content List

Import Messages as Format:   Append extension (.eml or .msg) to import

Overwrite existing files on server

Start Close Save Parameters

### Leave e-Mails on server (select file to keep track of emails)

You can store a history of all previous emails in a specific file on your server. When this option is selected, you can click the **Browse** button to browse to the file you desire to use to store the email tracking information.

### Move email messages to folder (IMAP only)

If you're using an IMAP email system, you can move the messages to a different folder for storage. When this option is selected, you can enter the name of the storage folder in the text box provided.

### Delete e-mail messages (Not Recommended for IMAP)

You can elect this option to delete the emails after they've been imported. This option is mainly applicable to POP3 email systems, and the POP3 server must be configured to delete emails after they are downloaded. This option isn't recommended for IMAP systems.

### Maximum Attachment Size (optional)

Enter a number, in bytes, of the maximum attachment size for emails you wish to import, if desired.

### Keep Attachments in Message Object

Selecting this option will encapsulate any email attachments into the EML or MSG file that will be created to contain the email message.

### Add attachments as References to message object in Content List

Selecting this option will add the email attachments as separate files and appended as reference objects to the EML or MSG files. You can then further manipulate the attachments, if desired. For instance, you can run an Item Actions Custom Task to import the attachments into a separate folder in the [Content List](#).

### Import Messages as Format

You can select to use either the MSG or EML file format to store the imported message.

### Append Extension (".eml" or ".msg") to import

Selecting this option will add the appropriate file extension to the imported file. This should be selected by default. For example, this is necessary if you plan to export the message to a local computer and use an email client to open it.

### Overwrite existing files on server

You can choose to have old files be overwritten by new ones.

## Scheduling Imports #

Using the Windows Scheduler, you can run the Email Import utility via a command line argument, e.g.,

```
bpEmailImport.exe -w "http://ServerName.com/" -s "mail.MailServer.com" -u "user1"
-p "" -i "Test Partition" --delete
```

The Email Import utility can be called by using a number of command line switches, which are described below. While in the command line interface, you can also see the full list of available switches by invoking the "/"? help switch, e.g., `bpEmailImport /?`

The available command line switches correspond to the properties that are accessible in the GUI, as described above.

SWITCH	DESCRIPTION
-W	The Process Director's web services URL
-S	The location of the mail server

SWITCH	DESCRIPTION
-I	The partition into which mail will be imported
-U	The Process Director user for web service authentication
-P	The Process Director password for web service authentication
-T	The protocol used to get mail from the server (IMAP or POP3)
-F	The Process Director folder into which the mail is imported
-L	The folder in which bpEmailImport's logs will be saved
--port	The port used to connect to the mail server
--ssl	Tells bpEmailImport to use SSL to connect to the mail server
--popuser	The mail server authentication user
--poppassword	The mail server authentication password
--imapfolder	The mailbox folder from which the mail will be pulled
--leave	Tells bpEmailImport to leave mail on the server
--leavefile	The file in which bpEmailImport records the mail it downloads
--imapmovefolder	The mailbox folder into which mail will be moved
--delete	Tells bpEmailImport to delete mail after download
--limitsize	The maximum attachment size (in bytes) to download
--strip	Tells bpEmailImport not to download mail attachments
--Nextension	Tells bpEmailImport to add file extension to uploaded mail
--addRef	Adds attachments as references to uploaded mail
--oauthtenant	Provides the OAuth Tenant
--oauthclient	Provides the OAuth Client ID
--oauthclientsecret	Provides the OAuth Client Secret
--overwrite	Tells bpEmailImport to overwrite existing uploaded mail
--limitcount	The maximum number of emails to import
--maxage	Maximum email age (in seconds) to download
--delay	The delay (in milliseconds) between each mail upload
--filterfrom	Only import mail where the from field matches this regex
--filtersubject	Only import mail where the subject field matches this regex
--filterbody	Only import mail where the body field matches this regex
-N	Runs bpEmailImport in command line mode
--no-gui	



## Optional Meta Data XML Files #

The import utility can include Meta Data with each of the files being uploaded to Process Director. This Meta Data can set the categories and attributes for the document on the server. This occurs if the option is selected to allow associated Meta Data files on the import dialog. If this option is selected the import utility will attempt to find matching file names with “.XML” appended to the name. For example, if the import utility finds a file named mydoc.pdf, it will attempt to locate the mydoc.pdf.xml file for the Meta Data. The Meta Data XML file must be of a certain format to be recognized as valid Meta Data. The XML file must contain the following structure and tags.

```
<META_DATA>
  <INPUT_META_DATA>
    <META_NAME>external_attribute_name</META_NAME>
    <META_VALUE>attribute_value</META_VALUE>
  </INPUT_META_DATA>
  <INPUT_META_DATA>
    <META_NAME>external_attribute_name_2</META_NAME>
    <META_VALUE>attribute value 2</META_VALUE>
  </INPUT_META_DATA>
</META_DATA>
```

The <META\_NAME> must match an External Name in a category attribute for the document to be assigned to that category. If a matching External Name isn't found in the category tree the Meta Data is discarded after the upload completes.

For example, assume you have a category named Operations that contains a subcategory named Quality with an attribute name of ActionManager. You can assign this entire category tree (i.e. Operations.Quality) to the imported document by using a matching External Name in the ActionManager attribute. If the external name in the ActionManager attribute is set to “manager\_name”, the XML for the imported document would look as follows:

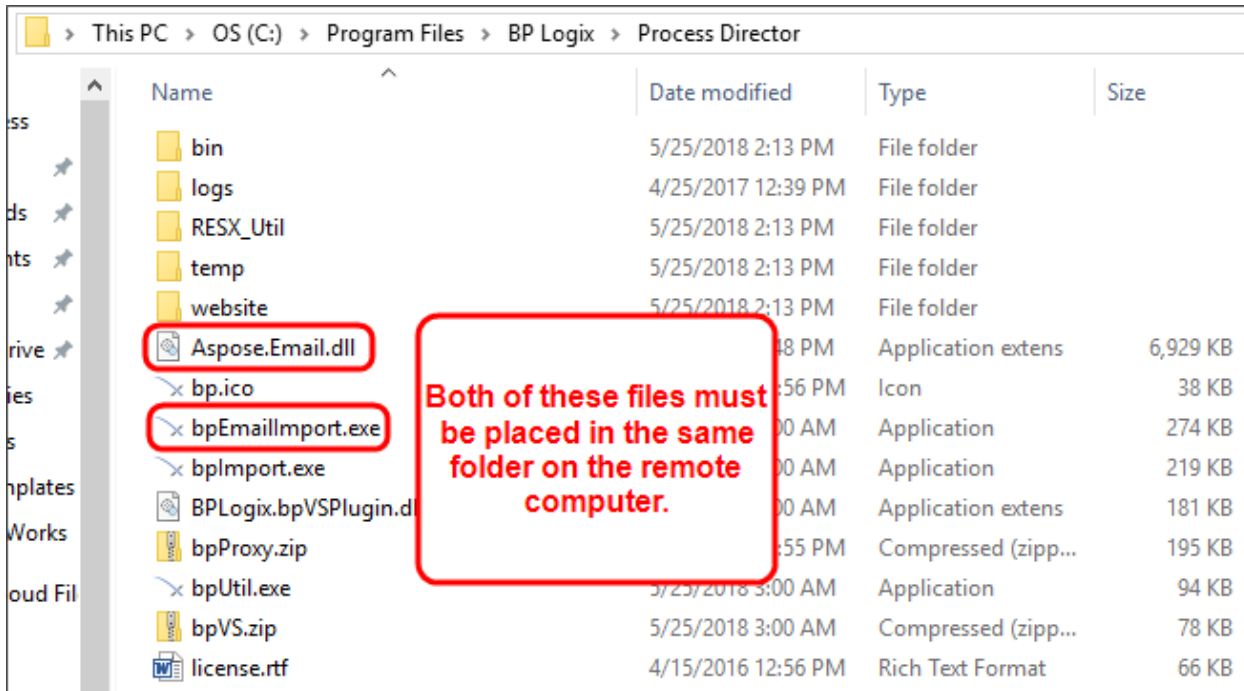
```
<META_DATA>
  <INPUT_META_DATA>
    <META_NAME>manager_name</META_NAME>
    <META_VALUE>John Smith</META_VALUE>
  </INPUT_META_DATA>
</META_DATA>
```

To set values for checkboxes, the values YES and NO correspond to a checked state and an unchecked state, respectively.

Date values vary based on locale: if you're in the USA, use mm/dd/yyyy. If you're elsewhere, use dd/mm/yyyy.

## Using the Utility on a Remote Machine #

The executable file, bpEmailImport.exe, is located in the c:\Program Files\BP Logix\Process Director\ directory by default. You can copy this utility to any computer that has Internet Explorer 6 or higher installed, and has Web Services running. When you do so, ensure that you also copy over the Aspose.Email.dll file into the same folder as you place the email import utility. Both files must be in the same folder for bpEmailImport.exe to run correctly.



Name	Date modified	Type	Size
bin	5/25/2018 2:13 PM	File folder	
logs	4/25/2017 12:39 PM	File folder	
RESX_Util	5/25/2018 2:13 PM	File folder	
temp	5/25/2018 2:13 PM	File folder	
website	5/25/2018 2:13 PM	File folder	
Aspose.Email.dll	4/25/2017 12:39 PM	Application extens	6,929 KB
bp.ico	5/25/2018 2:13 PM	Icon	38 KB
bpEmailImport.exe	5/25/2018 2:13 PM	Application	274 KB
bplImport.exe	5/25/2018 2:13 PM	Application	219 KB
BPLogix.bpVSPPlugin.d	5/25/2018 2:13 PM	Application extens	181 KB
bpProxy.zip	5/25/2018 2:13 PM	Compressed (zipp...	195 KB
bpUtil.exe	5/25/2018 3:00 AM	Application	94 KB
bpVS.zip	5/25/2018 3:00 AM	Compressed (zipp...	78 KB
license.rtf	4/15/2016 12:56 PM	Rich Text Format	66 KB

Simply execute bpEmailImport.exe to launch and configure the import dialog settings.

## Using the Microsoft Windows Scheduler

To schedule any Process Director task (e.g. an import task) using the Microsoft Windows Scheduled Tasks utility, follow one of the procedures below. One of the most common uses of the Windows Scheduler is [running the Activity check page](#) periodically, using the [bpUtil utility](#).



For most use cases that involve activities such as starting processes, or other internal Process Director actions, scheduled activities should be implemented through the use of the Goal object, rather than the Windows Scheduler.

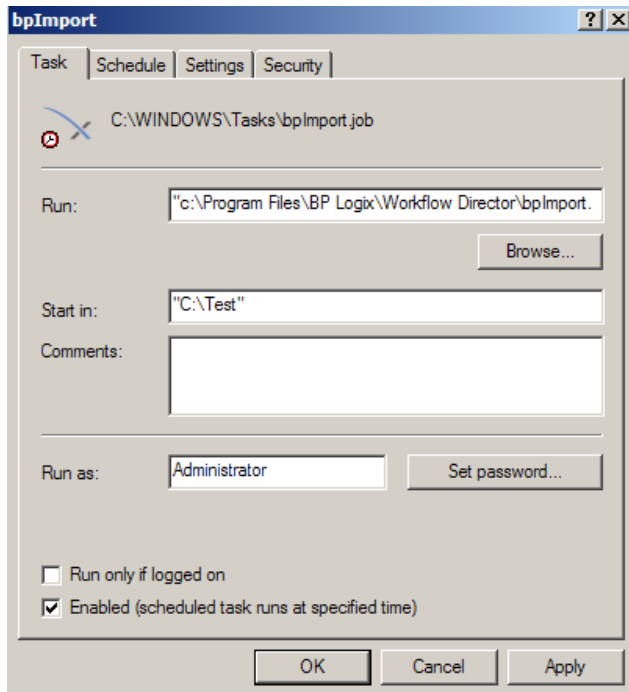
### Scheduled Tasks Utility (Windows 2008 and earlier)

Open the Windows Start > All Programs > Accessories > System Tools > Scheduled Tasks menu item. This will display the Windows Scheduler. Click on the “Add a Scheduled Task” item and browse for the application to run. Go to the Process Director install directory and choose the program (e.g. bplImport.exe). Then continue with the scheduler indicating that it should run every day. When prompted for user credentials, ensure you enter a user that won't have the password change occur too often, because any change to this password will prevent the scheduled task from running. You can optionally create a new user account to run this task under (ensure the account you choose has the appropriate permissions by logging in as that user and running the bplImport.exe).

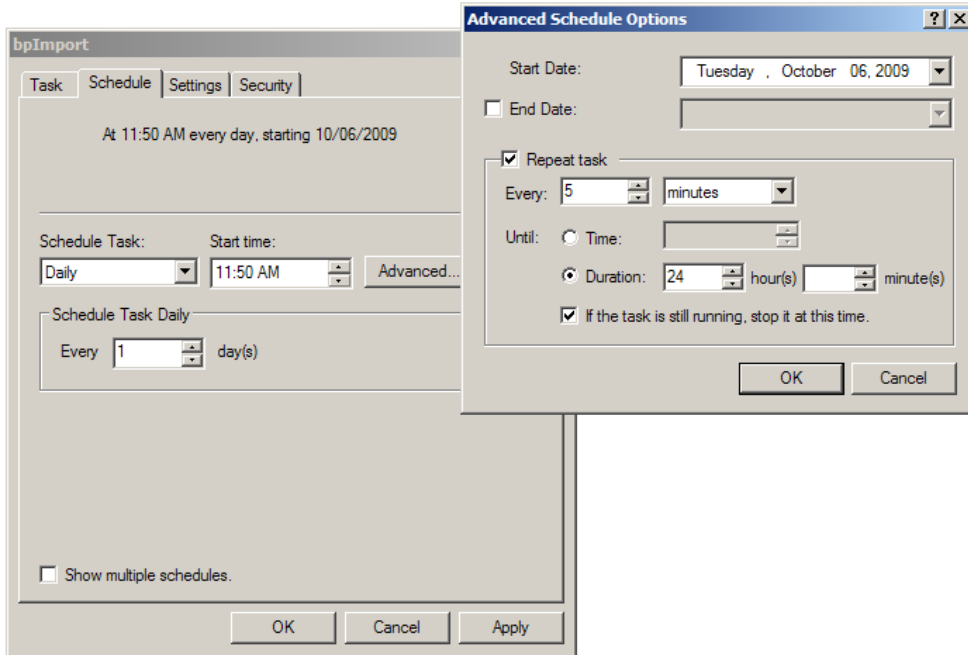
Once the task has been created, open the advanced properties of the item to configure the task. Change the Run input box to include the appropriate parameters to the command.

```
"c:\Program Files\BP Logix\Process Director\bpImport.exe" -w "http://localhost" -
u "user1" -d "m:\My Docs" --delete
```

The following dialog is displayed when the properties of the task are viewed. This is also where the Windows user account that this task is run as can be changed.



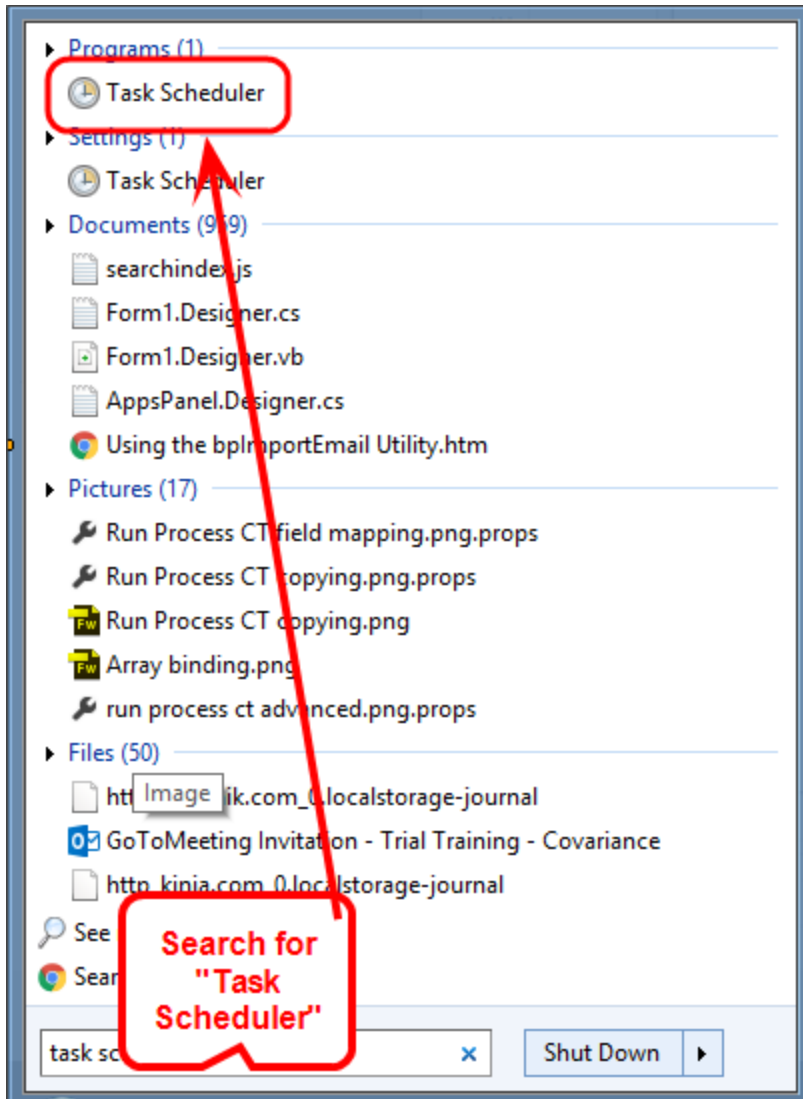
The scheduling wizard only allows the task to be scheduled once a day. To run this import command more often, click on the Schedule tab and then click on the Advanced button to reconfigure the schedule. In the advanced settings, click on the Repeat task check box, this will repeat the task at the interval you define. For example to run this command every 5 minutes all day, set the repeat task interval to every 5 minutes and set it to run for a duration of 24 hours. Consult the Microsoft help for more information on this utility.



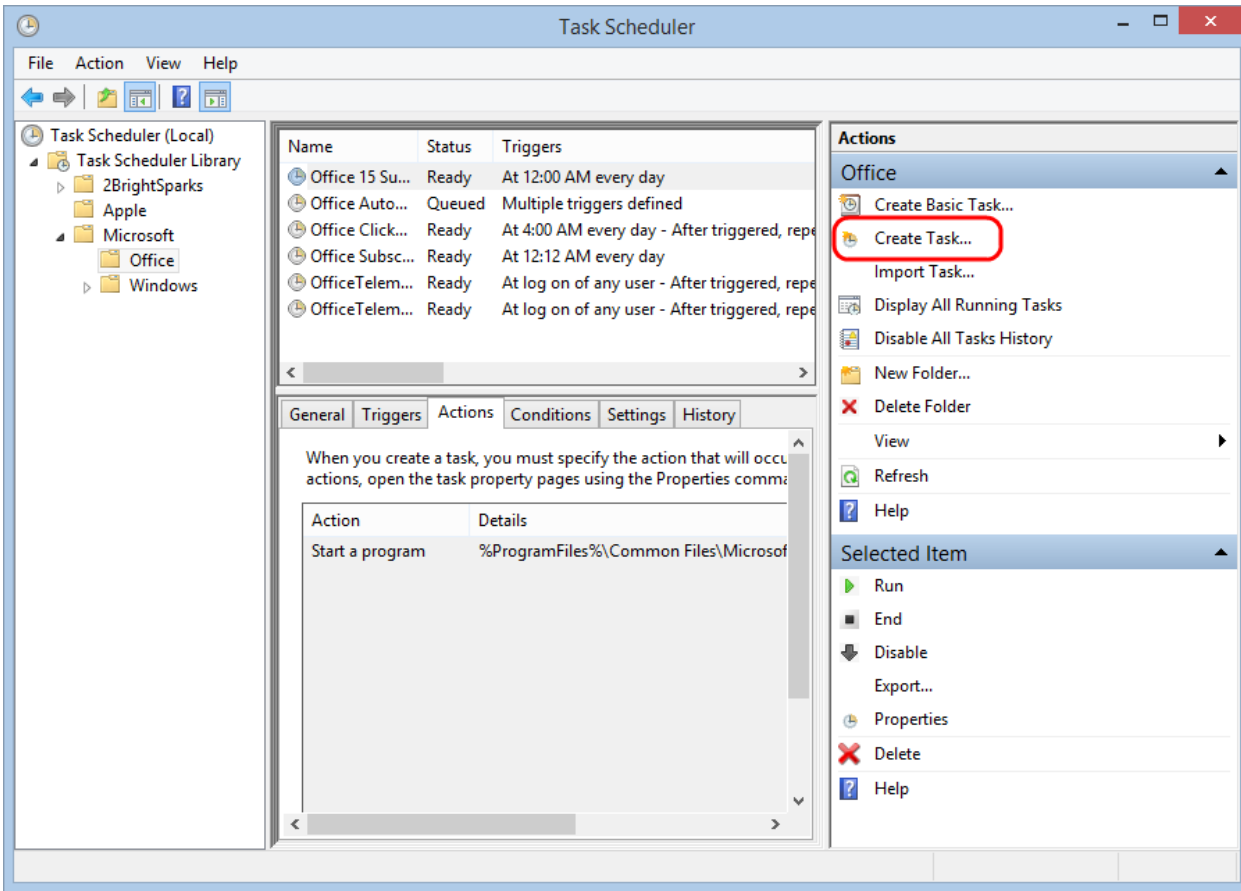
**i** When running the Process Director import utilities from the Windows scheduler, any errors will be written to a file named bpU.log in same the directory where the bpImport.exe exists.

### Task Scheduler (Windows 2013 and Later)

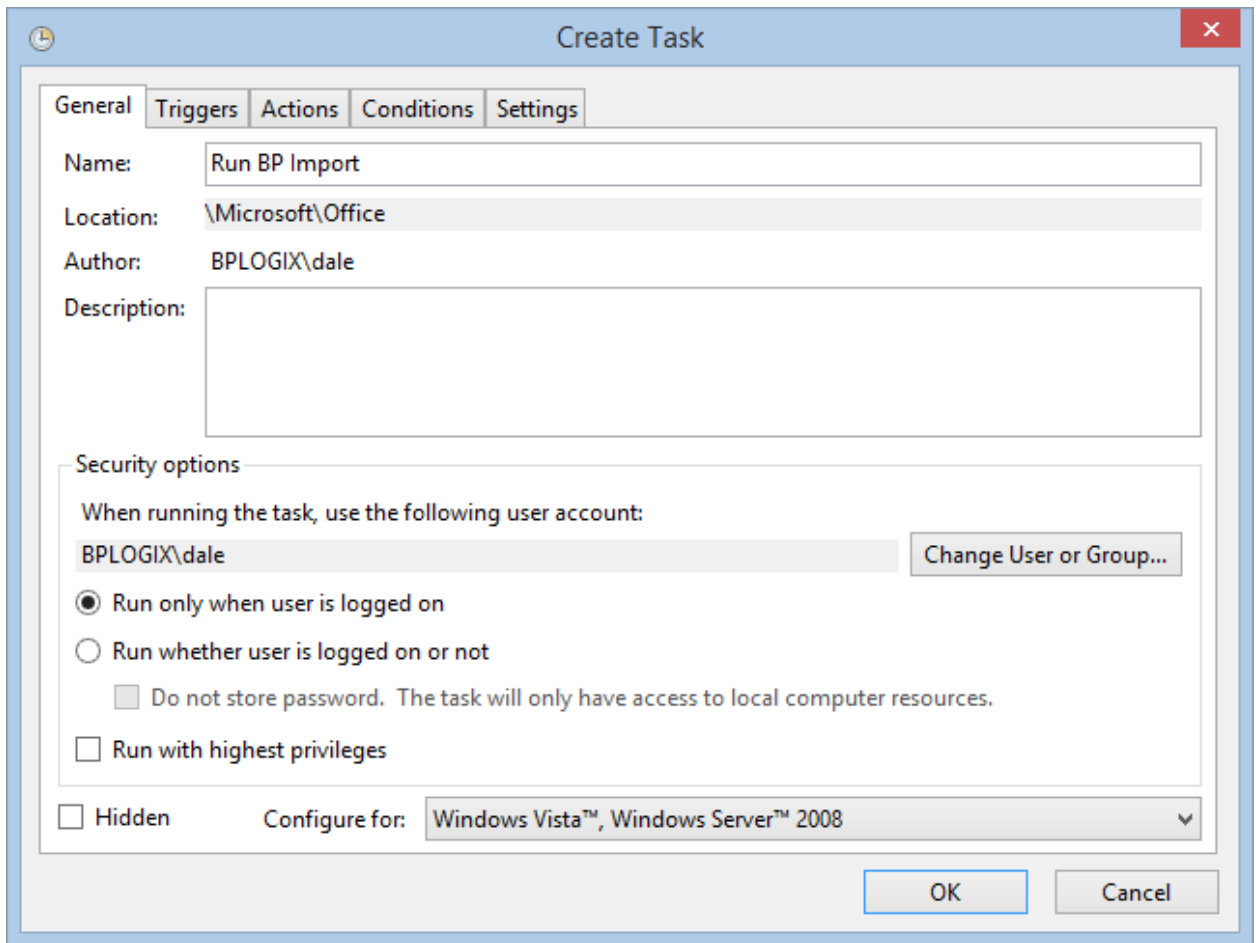
To open the Task Scheduler, click the Start Button, then type "Task Scheduler" in the search box. The utility will appear in the Start menu results. Click on the [Task Scheduler](#) program item.



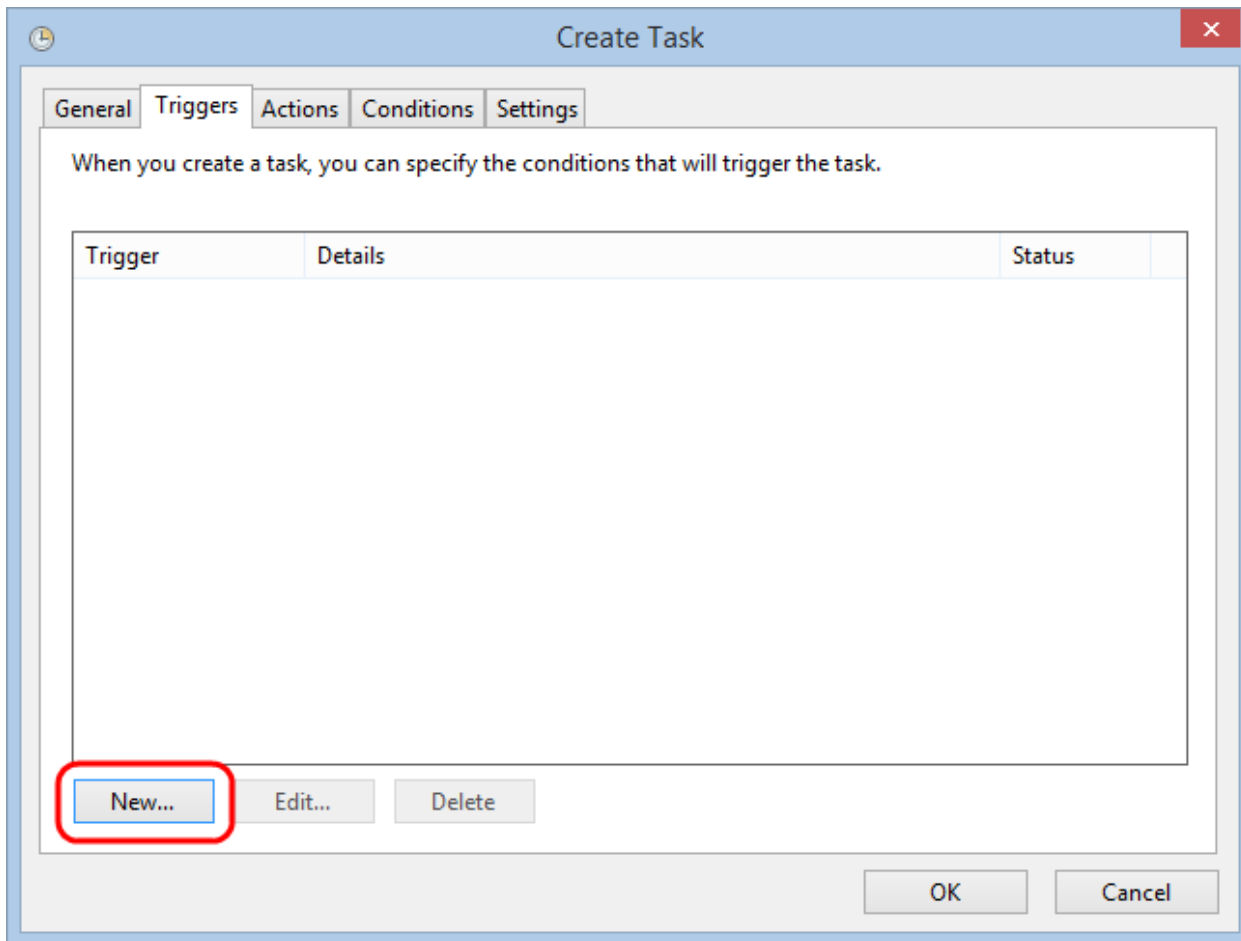
From the **Actions** pane on the right side of the **Task Scheduler** window, click the **Create Task** item to open the **Create Task** dialog box.



In the **General** tab of the **Create Task** dialog box, enter the **Name** of the task, e.g., "Run BP Import", then set the appropriate security options.



To create the schedule for the task, you'll need to add a trigger, so select the **Trigger** tab, then click the **New** button.



Clicking the **New** button will open a **New Trigger** dialog box from which you can set the tasks schedule. In the example below, the trigger is set to run the task every 2 days at 10:30 PM.



**New Trigger**

Begin the task: **On a schedule**

**Settings**

One time

Daily

Weekly

Monthly

Start: 10/26/2015 10:30:00 PM  Synchronize across time zones

Recur every: 2 days

**Advanced settings**

Delay task for up to (random delay): 1 hour

Repeat task every: 1 hour for a duration of: 1 day

Stop all running tasks at end of repetition duration

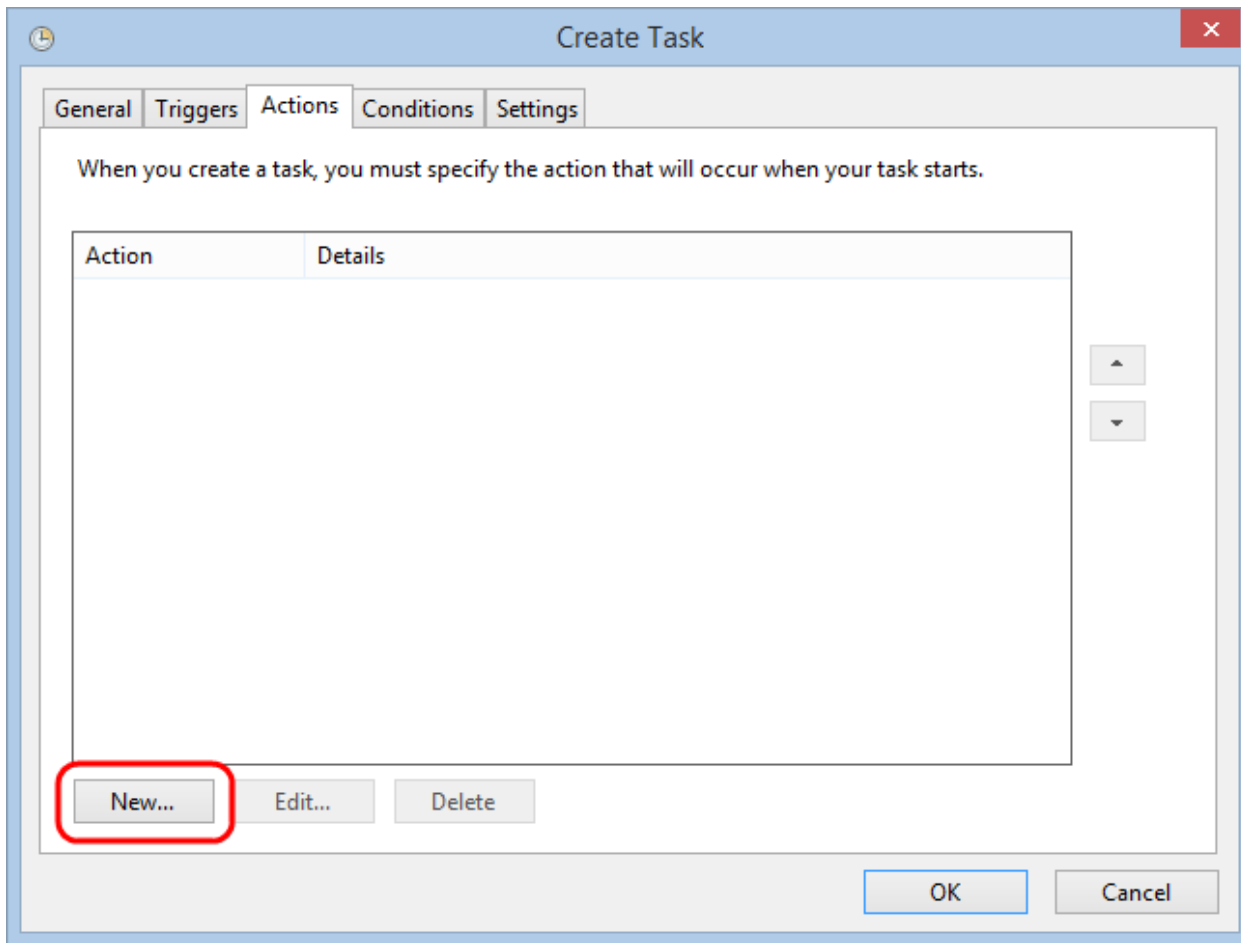
Stop task if it runs longer than: 3 days

Expire: 10/26/2016 10:38:40 AM  Synchronize across time zones

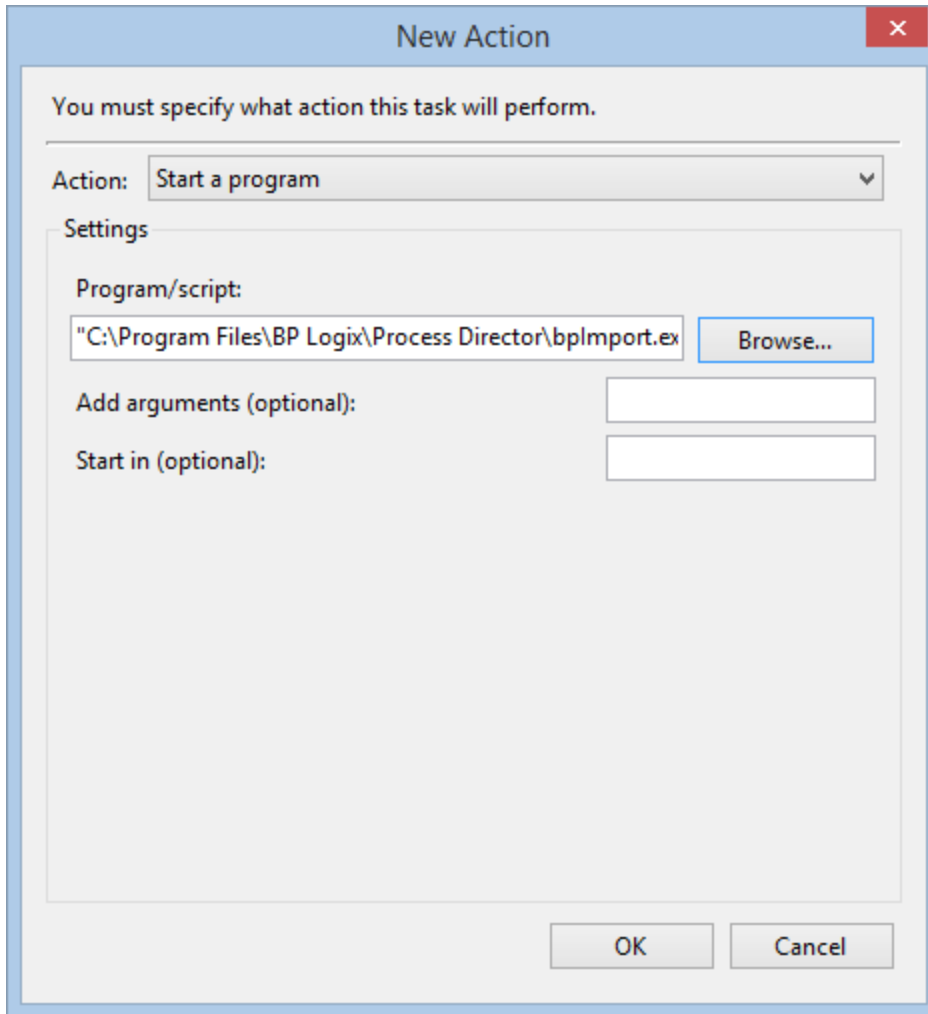
Enabled

OK Cancel

Click the **OK** button to save and close your trigger configuration, then click the **Actions** tab to set the task action. In the **Actions** tab, click the **New** button to open the **New Action** dialog box, from which you can create the action for the task, i.e., to run the `bplImport` task.



In the **New Action** dialog box, select "Start a Program" from the **Action** dropdown, then browse to select the bplImport task.



You can add the additional arguments in the [Add Arguments](#) box, e.g., `-w "https://localhost" -u "user1" -d "m:\My Docs" --delete`, then click the **OK** button to save and close the action.

To finish creating the task, click the [Create Task](#) dialog box's **OK** button.

**i** When running the Process Director import utilities from the Windows scheduler, any errors will be written to a file named bpU.log in same the directory where the bpImport.exe exists.

## Managing Users

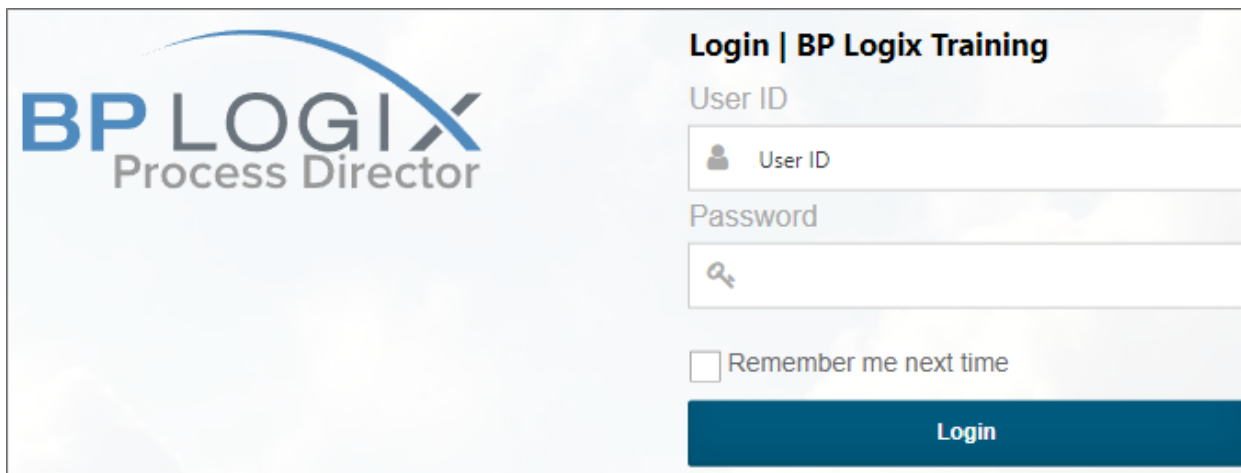
The following User ID's are created on a new Process Director installation.

- Administrator
- User 1
- User 2

The User ID's contain no passwords and have been given full permission to all objects on the server by default.

As part of your implementation, you'll create new User ID's and groups, as well as setting the appropriate permissions for your environment

When users want to login to Process Director, they must navigate to the login page located at <https://ServerName.com/> where [ServerName.com](#) is the host name where the server is installed. This will redirect the user to the login.aspx page in that directory. Depending on the authentication model used, the page below may be displayed, prompting for a User ID and password. If Windows Authentication is enabled, the user may be prompted with a Windows login dialog box to enter the user's credentials.



## User Profile <#>

Each user has a profile that contains settings and information about their User ID. The values include their email address, an optional Display Name and an option to change their password. Please note that only built-in users can edit settings on this page. Active Directory users must contact their administrator for changing profile details. The administrator can also control these options from the Process Director [IT Admin](#) area, allowing the profile information to be configured for users. A user must have the appropriate authority to modify their own settings (by default all users can modify their profile) if the menu item is available.

## Signature Image File

An optional signature image file can be associated with a user ID causing it to be displayed next to their name in the [Routing Slip](#) when they complete their task. You can upload a signature for each user in the User Administration section of the system. If a file exists with the same name as the UserID or UID in the `\BP Logix\Process Director\website\custom\signatures\` directory it will use that for the signature (it must have a “.gif” or “.jpg” file extension such as Susan.gif). It is recommended that a “.gif” file be used with a transparent background so the image can be displayed on web pages with different color backgrounds.

To attach a signature to a user, log in as an administrator and go to the [IT Admin](#) area's [User Administration](#) section, which will open the [Users](#) page automatically.



Click on the **Edit** link for the user to which you want to attach the signature.

The screenshot shows the 'Users' management interface. At the top, there are navigation tabs: Configuration, User Administration (selected), Installation Settings, and Troubleshooting. Below these are sub-tabs: Users (selected), Groups, Authentication Settings, Delegation, and User Directory Synchronization. A toolbar contains 'Actions: Create User', 'Export Users', and 'Import Users'. The main area is a table with columns: User ID, User Name, Email Address, User Type, System Admin, Last Login, and Commands. The 'Administrator' user row has its 'Edit' link highlighted with a red circle. A 'Close' button is at the bottom right.

<input type="checkbox"/>	User ID	User Name	Email Address	User Type	System Admin	Last Login	Commands
<input type="checkbox"/>	Administrator [All Users], admin		demo@bplogix.com	Built-in	True	6/21/2013	<b>Edit</b> Delete
<input type="checkbox"/>	AKelly [All Users], admin		andrew.kelly@bplogix.cc	Built-in	False	8/27/2013	Edit Delete
<input type="checkbox"/>	autoscaner [All Users]	autoscaner	escott@bplogix.com	Built-in	False	Never	Edit Delete
<input type="checkbox"/>	avi BP Logix, [All Users], admin	Aviel Natan Menter	aviel.menter@bplogix.cc	Built-in	False	8/18/2014	Edit Delete
<input type="checkbox"/>	b.mehaffy [All Users], NYAB		demo@bplogix.com	Built-in	False	4/11/2012	Edit Delete
<input type="checkbox"/>	barb Policy Group, Scheduling Group, IDC ...	Barb Stanley	demo@bplogix.com	Built-in	False	9/12/2014	Edit Delete
<input type="checkbox"/>	bob CAB, [All Users], cdsNormalUsers ...	Bob Barker	demo@bplogix.com	Built-in	False	8/29/2014	Edit Delete
<input type="checkbox"/>	bplogixsupport BP Logix, admin, Executive ...			Built-in	False	9/3/2014	Edit Delete

Click **Browse** on the Signature Image field, select the signature's image, and then click **Upload**.

User ID  
barb

User Name  
Barb Stanley

Email Address  
demo@bplgix.com

System Administrator?     System Partition Admin?     User Impersonation Ability?  
 System Configuration Admin?     System User Admin?     System Install Admin?     System Troubleshooting Admin?  
 User Disabled     Disable Emails for this User     User Account Locked     Force password change at next login  
 Change Password

Default Time Zone     Auto DST

Delegate all tasks to (User):    ...    Delegate Tasks To User

Replace this user with (User):    ...    Replace User Now

Signature Image  
Browse  
Upload    Remove

Picture Image  
Browse  
Upload    Remove

Preferred Language / Locale    English

Assign Groups    Policy Group, Scheduling Group, IDC ...

Object ID: 285dfbaf-54c8-4c6c-8f0e-a2ce8770f624  
External GUID:  
Sync Profile:

Save your changes by clicking the **OK** button.

As an aside, you can follow the same process to add a user's image to the **Picture Image** field of the profile.

If a signature file is found it is automatically displayed anytime the **Routing Slip** is viewed. Below is an example of how the **Routing Slip** would appear with signature files uploaded to the system.

Routing Slip						
Participants	Signature	Completed	Status	Result	Comments	
Initiator						
Ron Harris	<i>Ron Harris</i>	8/25/2014	Completed			
Approval Required 8/25/2014 3:25 PM						
Diana Stuart	<i>Diana Stuart</i>	8/25/2014	Completed	✓ Approve	Looks great to me, can't have too many batteries	
Department Head Approval 8/25/2014 3:26 PM						
Barb Stanley	<i>Barb Stanley</i>	8/29/2014	Completed	✓ Approve	I approve	
Buyer Processing 8/29/2014 8:32 AM						
Rick Rob		-	Active			

## External Users #

Process Director can be used by **external users**, which is to say, users from outside your organization. The most common use for external users is to include customers, suppliers, or other outside stakeholders as participants in a process.

In Process Director, **authenticated users** are those users who can be positively identified by Process Director, either through their internal Process Director user accounts, Active Directory, SAML, or Windows Single Sign-On. These users are considered to be internal users.

External users fall into two broad categories. **Anonymous users** are those users who can't be identified, and have no identity within the system, as authenticated users do. Anonymous users are only able to submit forms, and can't be assigned tasks as a process participant. A **Non-Authenticated** user has no user account in the system, but does have a known email address that can be assigned a pseudo-identity. When an Anonymous user is identified with an email address, the user automatically becomes a Non-Authenticated user, and can be assigned tasks as a process participant, via email.

It is important to remember that the Anonymous and Non-Authenticated user designations don't describe two separate classes of external user, but rather describe two different roles that may be assigned, at various times, to the same user. An external user can be an Anonymous user in the context of one operation (e.g., anonymously submitting a public-facing form), while simultaneously being a Non-Authenticated user in the context of a different operation (e.g., being assigned a process task via email address). Whether an external user is considered Anonymous or Non-Authenticated is based entirely upon whether or not the user has been identified by email address in the context of a specific operation. If so, the user is a Non-Authenticated user in the context of that operation, but is still an Anonymous user in all other contexts.



For systems still licensed under the legacy Tiered/Perpetual license models prior to 2017, access to Process Director for unauthenticated or anonymous users is enabled only by an optionally licensed component.

A **process participant**, is any user, authenticated or non-authenticated, who is assigned a task in a process. In the case of non-authenticated users, tasks are assigned by sending an email to the user's email

address. The email contains a special link to open a Process Director object, such as a Form, to enable the non-authenticated user to perform a process task.

All external users who can't be specifically identified by Process Director (including both Anonymous and Non-Authenticated users) are considered part of the Anonymous Users group when setting Process Director object permissions.



As mentioned above, some Process Director licenses don't allow anonymous users. Additionally, for licenses that are granted on a per-user basis, any participants in a process—whether authenticated or anonymous—are counted as Process Director users for the purposes of licensing. Contact BP Logix if you are unsure whether your installation of Process Director is licensed to enable anonymous users.

Anonymous users can be [granted permissions](#) to access objects just like any normal user by assigning permissions for an object to the Anonymous Users group. For example, you'd assign Read permissions, and perhaps Modify Children permissions, to Anonymous Users on public-facing Forms and Knowledge Views that you'd like external users to be able to access.



If an object is given Anonymous User permissions, then all other Process Director users will also have those permissions.

If a non-authenticated user participates in a process, Process Director will automatically grant the required object permissions to the user to view or modify objects, usually Forms, when the user needs to participate in or complete an assigned task in a process. Since non-authenticated users can't log into Process Director, their every action and notification is based on their email address. Process Director sends email messages to non-authenticated users that give them the appropriate links to Process Director objects. As long as Process Director has a valid email addresses for the non-authenticated users, they can do the following:

- Non-authenticated users can have a [Task List](#), but they can only access their [Task List](#) from an email that assigns them to a task.
- A non-authenticated user can be assigned a task just like a normal user, and can even be assigned a task in the same step as a normal Process Director user.
- The [Routing Slip](#) will display the appropriate indication when an anonymous user completes a task, usually by showing their email address as the participant name.

When a Process Timeline Activity is assigned to a user with an invalid UID or user ID, Process Director will immediately stop the process, and place the Process Timeline Activity into an error state. This is also true for anonymous user assignments if the email address isn't provided in a valid format.



Process Director can't validate whether the email address is valid, only that the email address is provided in the proper format.



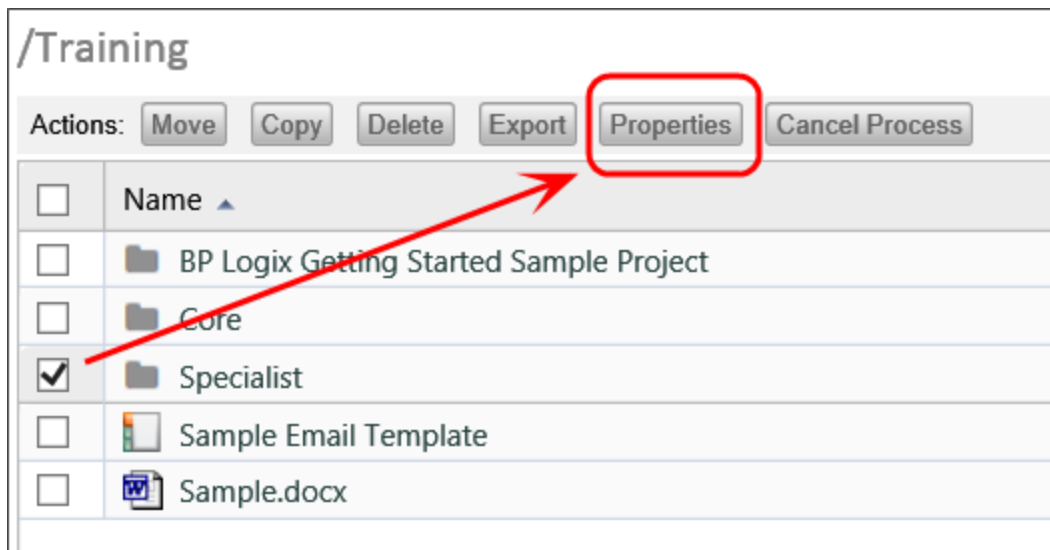
All user Timeline Activities have an option to enable non-authenticated users to complete a task based on the user's email address. This is a required option when using the [Email Anonymous Task List](#) system variable. In the vast majority of use cases, however, the email address will be taken from a Form field identified on the Participants tab.

In the case of user steps/activities that may be assigned to non-authenticated users, exercise care in using the “Automatically show user’s next task if it is in this process” setting. Process Director doesn’t distinguish among non-authenticated users when determining if the next task should be displayed, so if the current and subsequent tasks belong to different non-authenticated users, the subsequent task will still be displayed in the window of the current user. In the case of assignment to non-authenticated users, only use this option when you are certain that the same non-authenticated user will be assigned both tasks.

## Permissions

Permissions control who has access to different objects (document, Form, folder, etc.) in the Process Director database. Sometimes referred to as Access Control Lists (ACLs), permissions provide the foundation for the securing of objects in the database. They control who can perform what functions on which objects. Permissions can also be used to provide a simpler interface to end-users, hiding the more advanced functionality from them.

Permissions can also be accessed from inside any [Content List](#) object by opening the object's definition. Open the definition by selecting the object, then clicking the [Properties](#) button.



Once the object is open, the permissions for the object can be configured on the object's [Permissions](#) tab.

The screenshot shows a web interface for setting permissions. At the top, there is a 'Folder Name' field containing 'Specialist' and an 'Icon' field with a folder icon. To the right is a lock icon and an 'OK' button with a dropdown arrow. Below this is a section titled 'PERMISSIONS'. Underneath is a button labeled 'Add Permissions'. A table lists permissions for the 'Managers (Group)' user. The table has columns for Name, View, Modify, Delete, Run, View Children, Modify Children, Delete Children, and Commands. The 'Managers (Group)' row shows 'Yes' for View, 'No' for Modify, 'No' for Delete, 'Yes' for Run, 'Yes' for View Children, 'No' for Modify Children, 'No' for Delete Children, and 'Edit' and 'Delete' links in the Commands column. At the bottom, there are two buttons: 'Replicate Current Permissions to All Child Objects Only' and 'Replicate Current Permissions to All Child Objects & Instances'.

Name	View	Modify	Delete	Run	View Children	Modify Children	Delete Children	Commands
Managers (Group)	Yes	No	No	Yes	Yes	No	No	<a href="#">Edit</a> <a href="#">Delete</a>

See also:

[Permissions Methodology on Production Systems](#)

[Setting Permissions](#)

## Permissions Methodology on Production Systems

In Process Director, permissions are actually records that are stored in the Process Director database, and each record applies to a different user, group or class of users. Permissions records that are applied to a user or a group are static, which is to say that they always apply to those specific users. Records that apply to a class of users are dynamic, meaning that they can apply to different users at different times.

For instance, one class of users is Process Users. Any users who fall into this class, by virtue of participating in the process, receive the permissions for the class when they are accessing Process Director in the context of that process. At all other times, their permissions revert to the permissions of their group or their user account. So, a user might not have permission, under normal circumstances, to view a particular Form. But, when participating in the process, that same user might receive View and Modify permissions while they are participating. When they are done participating in the process, their permissions automatically revert to the normal permissions applied to their user account.

## Users

Anyone who participates in a process, or anyone who has an identity in the system, is a user. Anyone who is identified in Process Director, either through having an internal Process Director user account, or identification through Active Directory, SAML, or Windows Single Sign-on authentication is considered an authenticated user. Anonymous users are users who participate in a process, but whose identity can't be positively authenticated. For instance, you can assign tasks to users who aren't part of your organization, by assigning them a task via email. These are anonymous users who access Process Director through a special URL provided in the email. Process Director can't authenticate these users, and assumes that anyone who accesses a task via the email link is the desired user.

## Children

In Process Director, when a new child object is created, it automatically inherits the permissions of its parent. There are different types of objects that are considered children. For instance, in the [Content List](#), a folder is a parent object, while all of the objects contained in the folder, such as definitions for Forms, Process Timelines, Business Rules, etc., are the children objects for that folder. Similarly, every time a new Form is submitted, a new Form instance is created, and the Form instances are the children of the Form definition. Each Form instance, once created, now has its own permissions that apply only to it. Changing the permissions for a specific Form instance won't affect the permissions for any other Form instance.

Quite often, in Process Director, you attach documents to a Form or Process Timeline instance. When you do so, the attached document automatically inherits the permissions of the object instance to which it is attached. In essence, the attachment is treated as a sibling of the instance.

## Permissions Methodology

Not all permissions are meaningful for all objects. For instance, there is a Run permission, which is applicable to process objects like Process Timelines, but isn't applicable to a document like a Word or PDF attachment. Setting a Word document, therefore, with Run permissions doesn't accomplish anything useful.

When setting permissions, some permissions automatically imply others. For instance, if you give someone Modify permission to an object, you are also granting View permission as well. Similarly, Delete permission also applies View and Modify permission. Applying a more destructive permission always implies—and grants—any less destructive permission. In Process Director, the implied permissions are made explicit, in that, when you check the box to select a more destructive permission, the less destructive permissions will automatically become checked as well, so that you can see the granting of any applicable implied permissions as soon as you select the desired permission.

As mentioned previously, objects automatically inherit the permissions of their parent object. A Form instance inherits the permissions of the Form definition, a document attachment inherits the permissions of the instance to which it is attached, and so forth. You may, however, wish permissions for child objects to be different than their inherited permissions. Process Director contains a Custom Task to address this issue, the [Item Actions](#) Custom Task. In the [Item Actions](#) Custom Task, you can modify the permissions of attached objects. This does add some extra logic to your process, however, in that you must select when the Custom Task will run, e.g., from a Form or Process Timeline. But, however you choose to do it, you must explicitly change the permissions for the attachment via the [Item Actions](#) Custom Task.

When changing permissions via the Custom Task, you may create a permissions rule that conflicts with an existing one. For example, look at the screen shot below:

Form Instance Name    Icon    OK ▾

Permissions Form Submitted On 5/25/2022 10:58 ,

## PERMISSIONS

^ Add Permissions

Permissions For: All Authenticated Users ▾

Permissions To Be Granted

- Check All
- View
- Modify
- Delete
- Run

Add New Permission

Name	View	Modify	Delete	Run	Commands
All Authenticated Users	Yes	Yes	No	No	<a href="#">Edit</a> <a href="#">Delete</a>
Anonymous Users	No	No	No	No	<a href="#">Edit</a> <a href="#">Delete</a>
Anonymous Users	Yes	Yes	No	No	<a href="#">Edit</a> <a href="#">Delete</a>
admin (Group)	Yes	Yes	Yes	No	<a href="#">Edit</a> <a href="#">Delete</a>

In this example, Anonymous Users have two different permission rules. One of them grants View and Modify permissions to the form instance, while the second one, added via the [Item Actions](#) Custom Task, removes both permissions. Remember, the system will parse the entire permissions list and grant all permissions found. So, in this conflict, the most permissive rule will win, meaning that the View and Modify permissions will be granted, irrespective of the more restrictive permissions rule. So, you must think carefully about how permissions are applied, to avoid such conflicts.

Permissions should be set early on. If you have separate development and production systems, for example, you should have folder permissions already set on the production machine prior to importing any objects from the Development machine. Similarly, when you begin creating a project, you should create a project folder and set permissions on the folder prior to creating the project objects, since new objects always inherit the permissions of the parent object.

The question then arises, "What default permissions should be set on the parent objects?" Sadly there's no simple answer to this question, though there are general guidelines. You should always be sure to harden the permissions adequately to prevent unauthorized access to the objects, which means setting permissions so that as few people as necessary can access the objects. Commonly, hardening is done on the basis of Lines of business. You may have a parent folder that contains all of your HR processes, for example, and permissions to that folder might be limited to only members of the Human Resources department. Subfolders containing individual projects might be hardened even further, so that only specific sub-groups of the HR personnel can access a specific project. Line of Business hardening is, perhaps, the most common method of determining permissions.

As mentioned previously, permissions don't import to or export from a system. Indeed, permissions are specifically prevented from importing/exporting. Usually, the permissions on a development system are far more permissive than on a production system, if for no other reason than designers/developers

shouldn't be able to access many processes, like sensitive financial or human resources processes, in a production environment. Preventing permissions from exporting or importing enables you to set up hardened permissions on the production server, while leaving the development server relatively open to support collaborative design efforts.

User administrators have the ability to replace one user with another. This is useful as people change jobs or leave the organization. When using the Replace User function, user replacement automatically replaces the old user in all of the permissions records with the new user. In other words, the permissions assigned to the old user will be assigned to the replacement user. This is a convenient way to replace a user without having to recreate all of the permissions records for the new user from scratch. User replacement is, essentially, a universal replacement with permissions, just as it is with process participation.

In cases where you'd like users to open and submit a form—even if they don't have permission to view or modify the Form in any other context—you can assign all users to have Run permissions for the Form. After the user submits the Form, all of the regular permissions are applied, so if the user doesn't have Read or Modify permission, they'll no longer be able to see the Form in most cases. The exception is if the user is assigned a task later in the process. Process users are always temporarily granted the minimal permissions level they need to complete a task, so, even users who don't normally have permission to see a Form can open the Form in the context of completing the task. Once the task is complete, regular permissions are, again, applied to limit the user's access.

In addition to permissions, Forms offer another level security, in that you can show or hide controls on a Form by using the visibility properties, or disable controls with the Readonly properties. In both cases you can set conditions the hide or disable Form controls based on the user's identity, or any other evaluable condition. This enables you to hide sensitive data even for users who need to see other portions of the form. This isn't permissions-based, but it does add another level of security, in addition to controlling access to the Form itself.

When setting permissions, the question always arises of where to implement the desired permissions settings, since you can set permissions on any object. As a best practice, you should set permissions at the folder level for [Content List](#) objects. When you create or import objects in the folder, the new objects will automatically inherit the permissions of the folder. So, in the case of importing from development to production, if you set the permissions on the parent folder on the production system prior to the import, all of the permissions for the folder will be replicated to the imported children objects automatically.

Finally, you should exercise care when moving a [Content List](#) object from one folder to another. When you create an object in a folder, it inherits the folder permissions on creation. Process Director has a Move function that enables you to move objects from one location in the Content List to another. An existing object that is moved from one folder to another will bring its existing permissions with it, rather than overwriting them with the permissions of the folder to which it is moved. So, when moving objects to a different folder, you must go to the new folder's permissions list and click one of the Replicate Current Permissions buttons to overwrite the existing permissions of the moved object with the permissions of the new parent folder. Alternatively, instead of moving the object, you can export and reimport it to a new location, as exporting does remove all permissions from the object. In most cases, though, it is probably easier to simply move the object, then replicate the parent folder's permissions.

## Delete Permissions

No user account in a production version of Process Director should have delete permissions on any object, nor should any user account in Production be designated as a partition administrator. The only account that should have delete permissions, or be designated as a partition administrator, is a built-in administrative account. No one should ever log into this account unless they are specifically doing so to delete items from the [Content List](#). You may wish to have a delete-enabled administrative account for each user to whom you wish to give delete permissions, in order to track specific users who might delete objects, but these accounts should *not* be the accounts the users regularly use to log into the production system.

Deleting an object definition from the [Content List](#) automatically deletes all instances of that objects. Deleting any [Content List](#) object is a permanent action. There is no "undo". Objects deleted from the Process Director database are gone forever. The only remediation for deleting an object and/or its instances is to restore the database from the most recent backup.

For this reason, delete permission on production systems should be very tightly controlled. No one should be able to delete any [Content List](#) object when logged in using their regular identity. Force users to log into specific administrative accounts to delete any [Content List](#) object from a production system.

## Permissive Access

By default, all permissions in Process Director are permit-style permissions, which means that users are denied access to objects unless they are specifically permitted to see the objects. The permit-style permissions model adequately hardens Process Director for nearly all purposes.

There are a limited number of use cases that might require the denial model. For example, you can't deny a user who is otherwise included in a permitted group without deny permissions. So, in a law office, you might want to bar one of the attorneys from participating in a specific case because of a conflict of interest. Deny permissions would be required to prevent the attorney from viewing the objects that are included in that case instance. Use cases like this should be rare, however.

Process Director can implement the denial model by setting the [fEnableDenyPermissions](#) custom variable to true in the custom vars file. Implementing the denial model isn't cost free, however, because it requires the system to perform additional permissions checks every time a user opens an object. This may not be noticeable when opening a form, but it will make Knowledge Views run perceptibly slower. Unless you have a specific use case for implementing denial permissions, you shouldn't implement the denial model.

## Setting Permissions

Any object in the Process Director database may contain a list of permission rules that determine who can access the object and what functions they can perform against it. A permission rule only grants a user access. The absence of a permission rule granting a user access to an object will automatically deny access to it. If there are conflicting rules which either grant or deny a user access, the user will be granted access. When the server attempts to determine whether a user has access to an object in the database, it will search through the object's entire permission list until it finds a permission rule that grants the user the requested access, or until the end of the list. The system will apply the **highest level of access** that has been granted to that user in the list of permissions.

Different object types in the database have different permissions and options available to them. Objects that can be run (e.g. process definition, Form definition) will have more permissions options available to them, as will objects that can have children objects (e.g. folders, Form definitions, etc.).

Folder Name  Icon

**PERMISSIONS**

^ Add Permissions

Permissions For:

Permissions To Be Granted

- Check All
- View
- Modify
- Delete
- Run
- View Children
- Modify Children
- Delete Children

Add New Permission

Name	View	Modify	Delete	Run	View Children	Modify Children	Delete Children	Commands
All Authenticated Users	Yes	Yes	Yes	Yes	Yes	Yes	Yes	<a href="#">Edit</a> <a href="#">Delete</a>

Replicate Current Permissions to All Child Objects Only    Replicate Current Permissions to All Child Objects & Instances

The Permissions list is displayed in the lower portion of the permissions screen and will display the user and group names sorted alphabetically.

## User Permission Types #

Permissions are granted to a user. A permission record gives permission to something for a specific object in the Process Director database. Different types of user permissions can be granted.

### Authenticated Users

An authenticated user is anyone who performs a login to access the system. Authenticated users have an identity (i.e. User ID) and exist in the Process Director database. They may be authenticated by Process Director, your Windows Domain, an LDAP server, or an external application.

### Specific User

This identifies a specific authenticated user by name. All users known to Process Director have a User ID that is used to login. They may also have an optional display name or Alias. If an Alias exists for a user, that name is used to identify the user on all Process Director screens. If a user appears multiple times in a permission list, they'll be granted access according to the highest level of permission listed.

### Specific Group

A group is a collection of users in the Process Director database. A user can belong to multiple groups. Only authenticated users can belong to a group. When access to an object is being determined by the server, it will give the user the highest level of permission found in any specific user record or any group

the user is a member of.

### Anonymous Users

Anonymous users aren't authenticated. They don't perform a login and they have no identity in the Process Director database. Only certain functions and objects are available to anonymous users. If an “anonymous” permission is given to an object, authenticated users will automatically be given that same permission.

### Process Users

Any users who participate in a process as a task assignee receive the permissions for this class when they are accessing Process Director in the context of that process instance. At all other times, their permissions revert to the permissions of their group or user account.

### Object Creator

This is a special permission type only available to objects that can have children (i.e. folders, process definitions, Form definitions). This will automatically give the permissions specified to the author/creator of any child objects that are created. For example, if a user starts a process definition that gives the Object Creator a permission of Modify; that will automatically be converted to the initiator users ID with Modify permission for that specific process instance the user is creating (starting).

## Object Permission Types #

Setting the permissions for a folder doesn't automatically set the permissions for the existing objects contained within it. You have the ability to replicate all permissions of the parent folder to all of the existing child objects in the folder. You have two options for doing so.

First, you can Replicate permissions to all child objects, which won't only overwrite the existing permissions on definition objects, it will also overwrite permissions on all of the form and process instances—including those instances that are active and in-progress. This may completely change the ability of users to complete active Process Timeline or Form instances, so you should exercise caution in using this option in the production environment.

You also have the option, though, to replicate the permissions to child all child objects except for form, process, and Process Timeline instances. This is probably the preferred option for replicating folder permissions in the production environment where there are active processes running. Future process and form instances will be created under the new permissions regime, but existing instances will still run under the permissions regime that was valid when they were created.

Replicate Current Permissions to All Child Objects Only

Replicate Current Permissions to All Child Objects & Instances

### View

This permission gives a user read access to the folder. This doesn't grant the ability to modify the folder name or description. If a user only has View permission to a folder they won't be able to create any new objects below this folder.

### Modify



This permission gives a user the ability to modify a folder, but not delete. A user with this permission to a folder can create new objects beneath this folder. A user must have Modify permission to view or set the permissions for the folder.

### Delete

This permission allows a user to delete a folder. However, to delete a folder, the user must have Delete permission to all objects and sub-folders under this folder. If any object exists below this folder that the user doesn't have Delete permission to, the delete operation will be canceled and none of the objects will be deleted. A user must have Delete permission to be able to grant Delete permission to another user.

## Document Permission Types <#>

### View

This permission gives a user the ability to view, read or download a document. This permission doesn't grant the ability to modify the document.

### Modify

This permission gives a user the ability to modify the document. The document modifications will appear in the document history. A user must have Modify permission to view or set the permissions for a document.

### Delete

This permission gives a user full control over a document. A user with this permission can delete the document. Delete permission is required for a user to perform a rollback to an older version of a document, or to delete any portion of the document history. A user must have Delete permission to be able to grant Delete permission to another user.

## Process Permission Types <#>

Permissions are defined for a process definition, e.g., a Process Timeline. When the process definition is run, a child is created as an instantiation of the process definition. This “running” process is located beneath the process definition in the [Content List](#). This instantiated, running process will inherit the Child Permission settings from the process definition.

### View

This permission gives a user the ability to view a process definition in the [Content List](#). It doesn't grant the ability to run or modify the process definition, and doesn't grant any permission to the running processes instantiated from this definition.

### Modify

This permission gives a user the ability to modify a process definition in the [Content List](#). It doesn't grant the ability to run a process definition, and doesn't grant any permission to the running processes instantiated from this definition. A user must have Modify permission to view or set the permissions for a process definition.

### Delete

This permission gives a user the ability to delete a process definition. This will also delete all running and completed process Instances beneath this process definition in the [Content List](#). A user must have Delete permission to be able to grant Delete permission to another user.

### Run

This permission allows a user to run or start this process definition. If a user doesn't have Run permission to a process definition, it won't be displayed on their home page, nor will they be given access to the **Run** button for that object in the [Content List](#).

### View Children

This permission gives a user the ability to view the children of a process definition (e.g. running processes). When a process is started, the running process will copy all of the View Children permission rules as View permissions. If a user is granted View Children permission on the process definition, they'll be able to see all running and completed processes instantiated from the process definition. They will also be able to see the process definition in the [Content List](#) to enable the navigation to the running and completed process children. Users won't be view the actual process definition properties without View permission.

### Modify Children

This permission gives a user the ability to modify the children of a process definition (e.g. running processes). When a process is started, the running process will copy all of the Modify Children permission rules as Modify permissions. If a user is granted Modify Children permission on the process definition, they'll be able to modify running processes instantiated from the process definition. This gives the user administrative control over the running process (e.g. add user, remove user, skip step, restart step, etc.).

### Delete Children

This permission gives a user the ability to delete the children of a process definition (e.g. running processes). When a process is started, the running process will copy all of the Delete Children permission rules as Delete permissions. A user must have Delete permission to be able to grant Delete permission to another user.

## Form Permission Types <#>

Permissions are defined for a Form definition. When a Form is filled out and submitted, a child object is created as an instantiation of the completed Form. This completed form is located beneath the Form definition in the [Content List](#). Completed forms will inherit the Child Permissions from the Form definition.

### View

This permission gives a user the ability to view a Form definition in the [Content List](#). It doesn't grant the ability to run or modify the Form definition, and doesn't grant any permission to the completed forms from this Form definition.

### Modify

This permission gives a user the ability to modify a Form definition in the [Content List](#). It doesn't grant the ability to fill out and submit a Form, and doesn't grant any permission to the completed forms. A user must have Modify permission to view or set the permissions for a Form definition.

### Delete

This permission gives a user the ability to delete a Form definition. This will also delete all completed forms beneath this Form definition in the [Content List](#). A user must have Delete permission to be able to grant Delete permission to another user.

### Run

This permission allows a user to fill out and submit this Form definition. If a user doesn't have Run permission to a Form definition, it won't be displayed on their home page and they won't be able to open it in the [Content List](#).

### View Children

This permission gives a user the ability to view the children of a Form definition (e.g. completed forms). When a Form is submitted, the completed form data creates an entry in the [Content List](#) and copies all of the View Children permission rules as View permissions. If a user is granted View Children permission on the Form definition, they'll be able to see the completed forms for this Form definition. They will also be able to see the Form definition in the Content List to enable the navigation to the completed form children. Users won't be able to view the actual Form definition properties without View permission.

### Modify Children

This permission gives a user the ability to modify the children of a Form definition (e.g. completed forms). When a Form is submitted, the completed form will copy all of the Modify Children permission rules as Modify permissions. If a user is granted Modify Children permission on the Form definition, they'll be able to modify the completed forms under this Form definition.

### Delete Children

This permission gives a user the ability to delete the children of a Form definition (e.g. completed forms). When a Form is submitted, the completed form will copy all of the Delete Children permission rules as Delete permissions. A user must have Delete permission to be able to grant Delete permission to another user.

## Knowledge View Permission Types <#>

Knowledge Views display objects contained in the Process Director database. Authenticated and anonymous users can be given access to the objects displayed in a Knowledge View. A user must have at least View permission to an object for it to be displayed in the Knowledge View.

### View

This permission gives a user the ability to see the Knowledge View definition in the [Content List](#). This permission doesn't grant the ability to run the Knowledge View.

### Modify

This permission gives a user the ability to modify the Knowledge View definition in the [Content List](#). A user must have Modify permission to view or set the permissions for a Knowledge View.

### Delete

This permission allows a user to delete a Knowledge View definition. A user must have Delete permission to be able to grant Delete permission to another user.

### Run

This permission is required for a user to be able to run the Knowledge View. If a user doesn't have Run permission to a Knowledge View definition, it won't be displayed on the user's home page or when they select the Knowledge navigation button.

## Category Permission Types <#>

The category permissions are set in the category schema. To view the category schema use the Administration navigation button and select Meta Data Administration.

### View

This permission allows a user to view the category in the category schema. If a user doesn't have View permission to a category they won't see this category name in the schema definition.

### Modify

This permission allows a user to view and modify the category in the category schema. It allows the user to add sub-categories below this category. A user must have Modify permission to view or set the permissions for a category.

### Delete

This permission allows a user to delete the category in the category schema. It also allows the user to delete any sub-categories below this category. A user must have Delete permission to be able to grant Delete permission to another user.

### Run

This permission is required for a user to be able to assign the category to an object in the database.

## Default Permissions for New Objects <#>

Any time a new object (e.g. document, Review Set, sub-folder) is added to a folder it will be given the same permissions as the parent folder. If an object is added to the top-most folder (i.e. "<top>") it will be given the same permissions specified in the default Folder Permissions hotlink. If an object is copied or moved to a folder, it will keep its original permissions.

## Administrators <#>

Any user that is checked as the "System Administrator" is given full administrative privileges. A user that is a System Administrator has full permission (View, Modify, and Delete) to all objects in the Process Director database.

## Adding/Removing Permissions #

Permission records can be modified for an object by selecting the **Permissions** button. Each permission record has to be removed or added individually. Existing permission records can't be modified, they must be removed and a new record added. To remove a permission record, click on the checkbox for that entry and select the Delete hotlink. To add a new permission record, select the type of user, the type of permission and click on the "Add New Permission" button. You can't delete a permission record that would eliminate Write permission for yourself; the system will return an error message. If this occurs, add a new permission record granting your User ID Write permission, then delete the other permission record.

## Permission Exceptions

At the bottom of the **Permissions** tab a button labeled, **Show Permission Exceptions**, will, when clicked, display a chart of all child objects, if any, whose settings for the same assigned permissions class are different from the object you are viewing.

The screenshot shows a table titled "Child Permission Exceptions" with columns: View, Modify, Delete, Run, View Children, Modify Children, Delete Children. The table lists four child objects: Permissions Form, Permissions KView, Permissions TL, and Sample Form.pdf. Each object has a row for "[Authenticated]" and a row for "admin". Red circles indicate permissions that differ from the parent, and orange circles indicate permissions that match. Checkmarks indicate permissions that are present in the child but not in the parent.

Object	User	View	Modify	Delete	Run	View Children	Modify Children	Delete Children
Permissions Form	[Authenticated]	Red	Red	Red	Red	Red	Red	Red
	admin	Orange	Orange	Orange	Orange	✓	✓	✓
Permissions KView	[Authenticated]	Red	Red	Red	Red	Red		
	admin	Orange	Orange	Orange	Orange	✓		
Permissions TL	[Authenticated]	Red	Red	Red	Red	Red	Red	Red
	admin	Orange	Orange	Orange	Orange	✓	✓	✓
Sample Form.pdf	[Authenticated]	Red	Red	Red	Red	Red		
	admin	Orange	Orange	Orange	Orange	✓		

In the case of this example, the parent object is the application's parent folder. A form contained inside this folder, named **Permissions Form**, has different permissions from the folder in which it is contained. This chart can be used to determine which child objects may need to have permissions replicated.

**See also:**

[Permissions Methodology on Production Systems](#)

[Setting Permissions](#)

## Changing Existing Processes

You'll occasionally need to alter an existing Form or process. Perhaps more data must be collected from the Form, or the process itself must change. Since the process exists, and probably has some in-flight processes already active, you should understand how changing a process or Form definition will affect existing instances.

In general, all process changes are implemented immediately, as soon as the edited process or Form is saved or imported. The change can affect running process instances in various ways. Now, for the most part, changes to a process tend, in actual practice, to be fairly minor. So, in most cases, the change will require no administrative intervention. Process director will apply the change, and any existing processes

will simply use the new definition to complete the in-flight process, while retaining the data from tasks that are already completed.

In running processes, the **Routing Slip** will reflect the tasks that were completed in the old process, as they existed before the change, then subsequently mirror the new process after the point of change. Additionally, Process Director will check the running instances of the process to see if there are any notifications that need to be resent, and, of course, resend the notifications where appropriate. In most cases, therefore, changes to the process will be seamless.

There may be some exceptions, however:

- When you attempt to delete an activity from the Process Timeline definition, if there are any active processes that are currently running that activity, process Director will remove the activity from the dependence tree in the process Timeline definition, but will not delete the activity until all currently running instances of it are complete. Once the users complete the task, Process Director will, in most cases, resume with the next task that was previously scheduled. However, if you delete multiple timeline activities, Process Director may not be able to determine what activity should run next. In that case, the Timeline Instance will be placed in an error state, and will require administrative intervention. *Note that for versions of Process Director prior to v5.44, the system will allow you to delete Timeline Activities, but will not try to determine which activity should be invoked next, and place the timeline into an error state immediately.*
- When you add new activities to an early stage of the process, any existing instances that have passed the new activities won't return to complete them. The new activity won't run in that instance without some intervention, either administrative, or built into the process, such as by implementing a rollback for instances where the activity results have not been stored.
- When you add a field to a Form, all new instances will store the data properly, but existing instances won't contain any value for the new field. The value will, therefore be "null". In an existing process instance, if that field is used to make a determination of what process activity to start, the system won't evaluate the criteria as expected, which can lead to unpredictable results. For instance, Let's say you add a new Checkbox field to a form called **NewField**, and, in your process, you add a 'Needed When' condition to a Timeline Activity like "**NewField** is false". Your assumption might be that, since **NewField** is a check box and hasn't been marked "true", it must logically be false. But, actually, the field has a null value, because it didn't exist when the form was submitted, and is therefore, neither true nor false. In that case, the field will, when evaluated, fail the condition.
- When you delete a form control from a Form, existing Form instances will retain that data. The Form field will still exist in the Process Director database, even though the control has been removed from the Form's design surface. Always remember, however, that deleting a form control from the Form template will eliminate the control from the Form's display in **all** Form instances. The data will exist, but the form control won't be displayed, because Forms always display the current version of the Form template. Also, any form conditions that used the deleted field will no longer work once the control is deleted from the form's design surface. Please see the [Deleting Controls and Form Fields](#) topic for more information on how to handle Form data from deleted controls by deleting or remapping it.

**i** For users of process Director v4.06-5.44, when an Activity is deleted, any process that is running in that Activity will now be placed into an error state. It will remain in an error state until an administrative action is taken, e.g. jump to an activity, start an activity, rollback to a previous activity, etc. Any processes that went into an error state will appear in any Knowledge View that shows processes in error, and users will see a message indicating the reason for going into the error state.

**i** For users of process Director v4.31 or higher, when a new Activity is added to a Timeline Definition, the new Activity will be marked as "Not Required" in any running Timelines if all of its dependencies have already completed.

**i** For users of process Director v4.56 or higher, when a parent Activity is added above existing activities, and there are activities running in existing Timeline instances under that new parent, the system will now automatically start the new parent Activity.

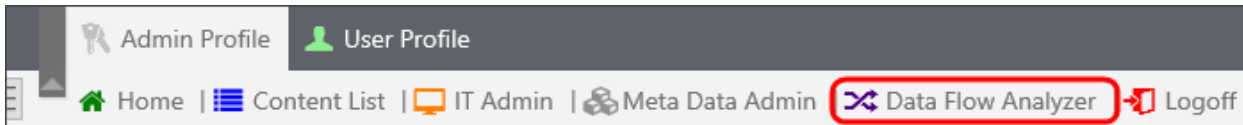
Though Process Director will try to make the Process seamless, in many cases process administrators will still have to intervene in the running processes. To make the intervention easier, you should create a Knowledge View that returns only processes that are in an error state. This will provide you with a list of all the processes that require intervention, making it easier to clear the errors. Similarly, you can create a Knowledge View that displays processes where new activities have been introduced, but which have not been completed in existing instances. In both cases, the Knowledge View feature is your friend.

In general, you shouldn't attempt to create a new Process Timeline when you change a process, unless the change is so fundamental that it constitutes a new process. If the change is that fundamental, then you should recreate the entire application. Otherwise, you should always edit the existing process. Doing so retains the accuracy of your auditing and historical data, whereas creating an entirely new application will have its own, independent data. Depending on your regulatory requirements, you'll still need to retain the old process intact, so that it can be audited, which means keeping an inactive process in your production environment until the regulatory requirement expires.

## Data Flow Analyzer

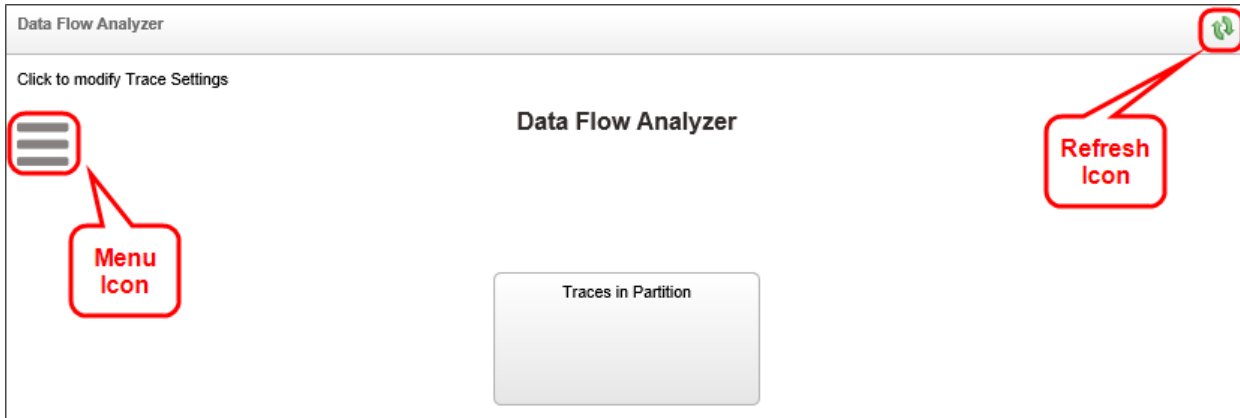
Administrators and implementers using Process Director v4.53 and higher have access to a general data viewer called the Data Flow Analyzer (DFA). The DFA enables you to track the data being used by Business Values, Knowledge Views, Business Rules, etc., so that you can see how the system retrieves data, and what data is retrieved by a particular object. The DFA assists administrators and implementers in testing and troubleshooting data calls made by Process Director objects.

By default, access to the DFA is exposed to Administrators via a navigation button in the Admin User workspace. Administrators can, at their discretion, add the DFA access button to other workspaces, such as those used by process implementers and testers.



Clicking the **Data Flow Analyzer** button will open the DFA in a new browser window.

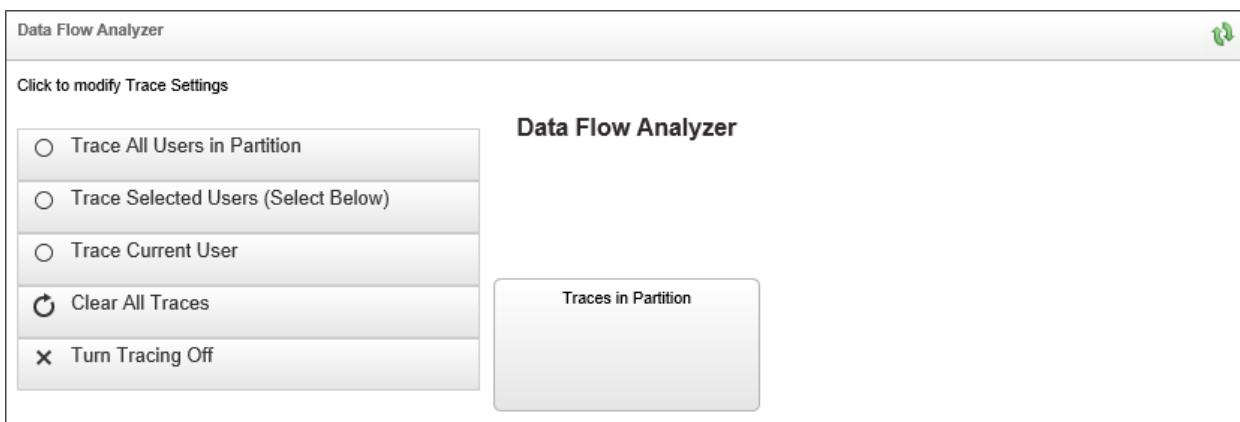
To configure the DFA, click on the **Menu** Icon in the DFA screen. To refresh the traces, click on the **Refresh** icon in the upper right corner of the DFA window.



Clicking the **Menu** icon opens the configuration menu for the DFA. Administrators can view data flows for

1. All of the users in the Partition,
2. For selected users, or
3. For themselves.


Non-administrative users can only track the data flows that they initiate, i.e. data flows for the Current User.




Clicking the **Clear All Traces** button will erase all of the currently visible tracking information, while clicking the **Turn Tracing Off** button will stop all data tracking.

The DFA operates automatically. Any time you run an object that is tracked by the DFA, a Trace for that object will appear in the DFA window after you refresh the window by clicking the **Refresh** icon. In the example below, clicking the **Test Query** button in a Business Value will result in a trace diagram similar to this:





Data Flow Analyzer 


Click to modify Trace Settings


 **Data Flow Analyzer**


Traces in Partition



  
Dale Franks  
Dale.Franks@bplogix.com

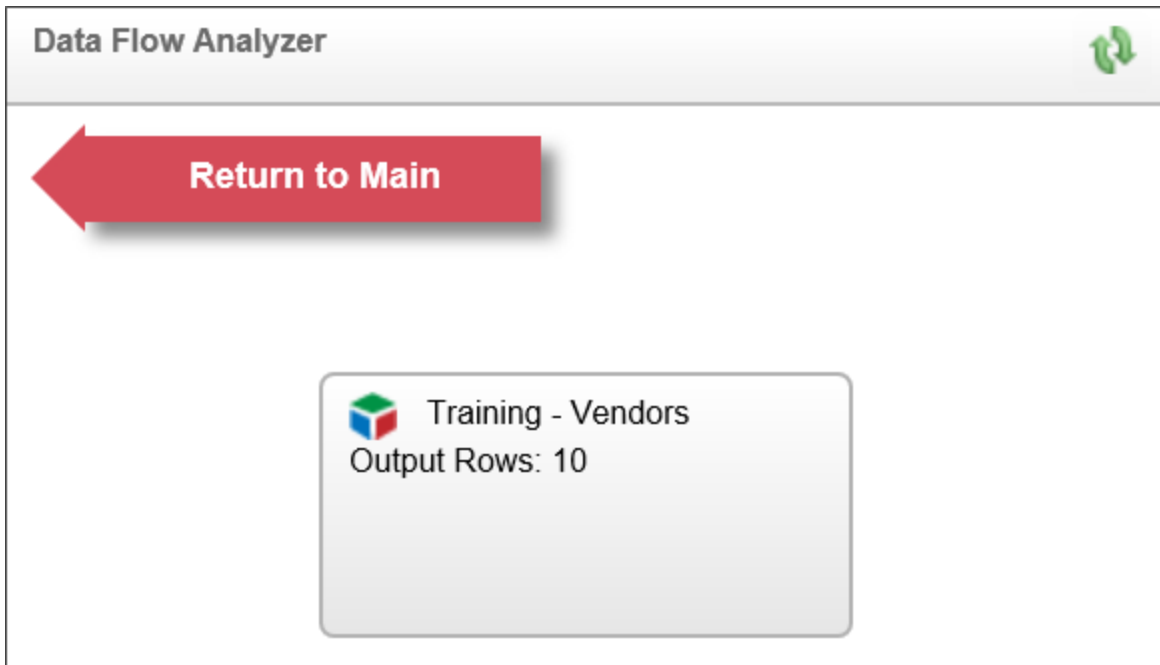


 **Business Value**  
Training - Vendors



**Execution**  
Training - Vendors  
2:53:52 PM

The trace appears as a tree that starts at the top with the [Traces in Partition](#) object. The second level of the tree consists of all users who are being traced. In this example, only one user is being traced. At the next level of the tree, the individual objects that have been traced will appear. Each traced object will then have an [Execution](#) box below it that displays the name, execution time, and a [Click to Zoom](#) link to display the details about the trace. In this case, we want to look at the Business Value that we tested, so if we click the [Click to Zoom](#) link for the [Training - Vendors](#) Business Value, the DFA will zoom to that node, and display the Business Value that was called by the form. We can exit the zoom by clicking the [Return to Main](#) button.



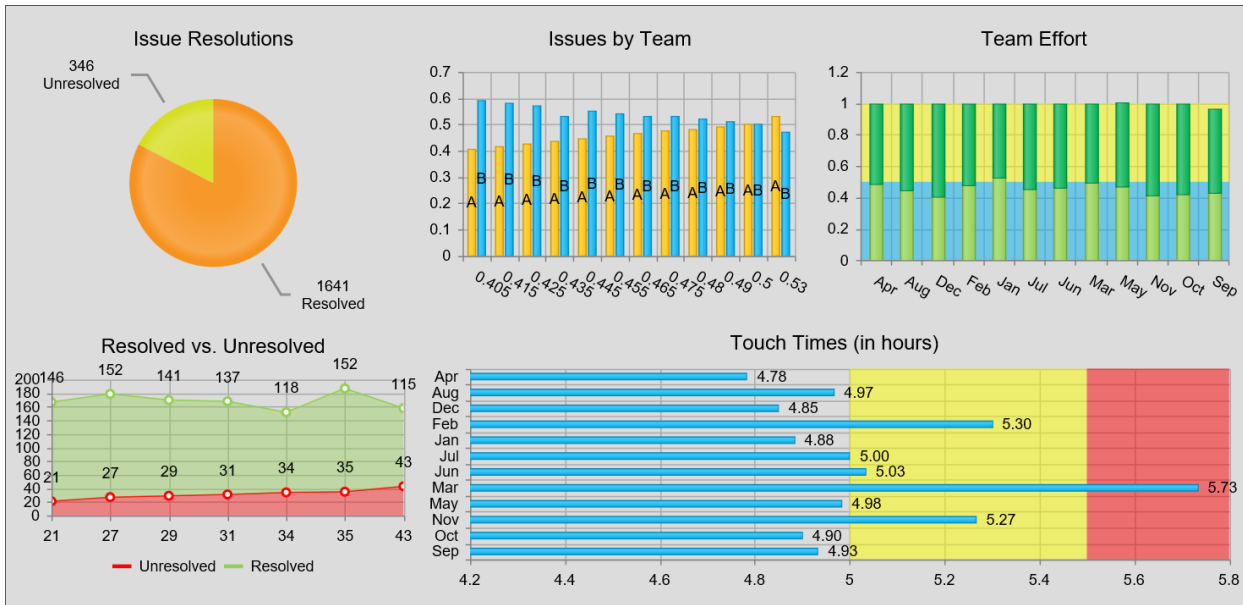
We can also view the details of the [Training - Vendors](#) Business Value by clicking on the [Training - Vendors](#) node.

The screenshot shows the 'Data Flow Analyzer' window. At the top left, there is a red arrow button labeled 'Return to Main'. In the center, a tab titled 'Training - Vendors' is active, showing 'Output Rows: 10'. Below the tab, the SQL command is displayed: **COMMAND:** SELECT VendorName FROM Training\_Vendor ORDER BY VendorName ASC. Underneath, the **OUTPUT:** is shown as a table with 10 rows of vendor names.

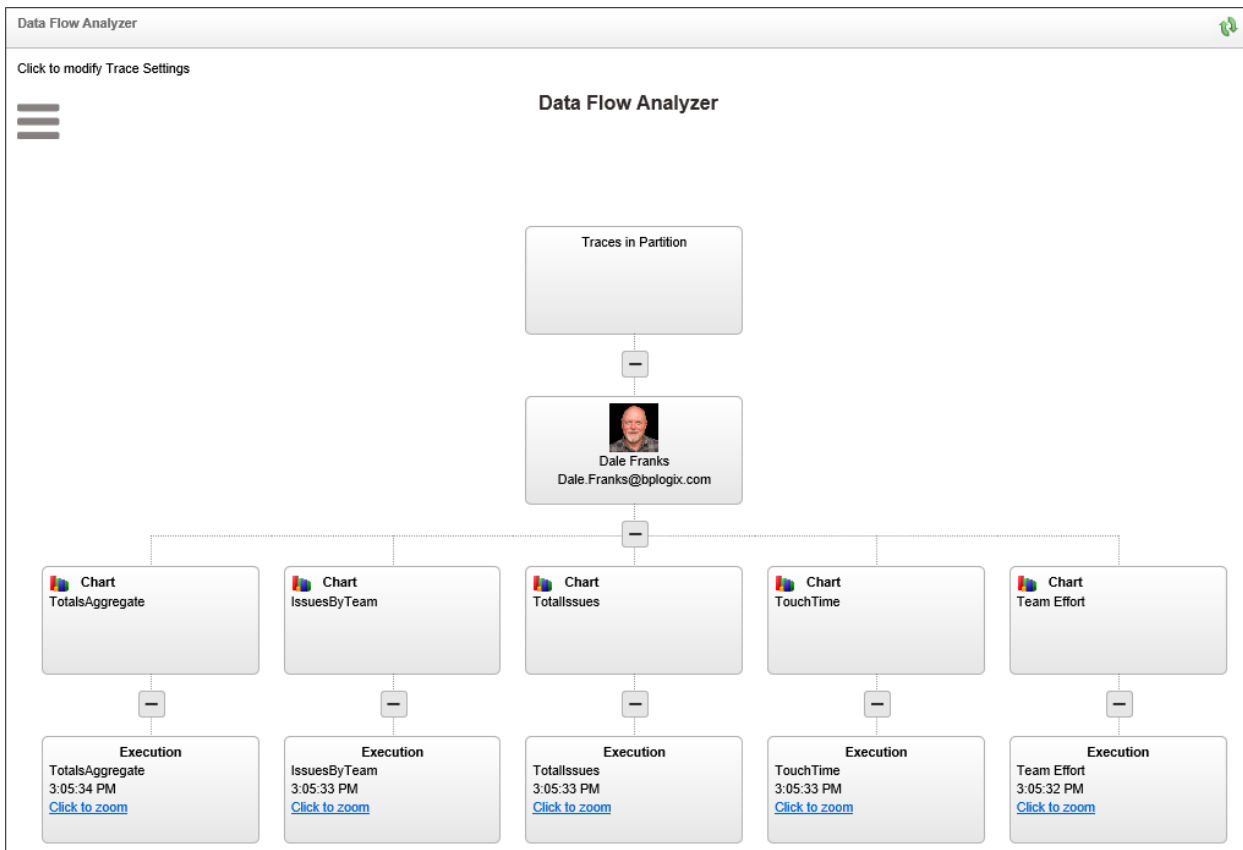
Arrow Electronics, Inc.
Baxter International Inc.
Birds Eye Co. Inc.
Chubb Corporation
Colgate-Palmolive Company
CVS / Caremark Corporation
General Electric Company
Kimberly-Clark Corporation
Nulka City Foods
Paychex, Inc.

On this details screen, we can see the SQL Command that generated the query, the Parameters, if any, that were used by the Business Value, and a table containing the records that were returned by the Business Value.

The more complex the object, the more complex the trace will be in the DFA. For instance, let's say that we start the DFA, and open a Dashboard that contains several Charts.

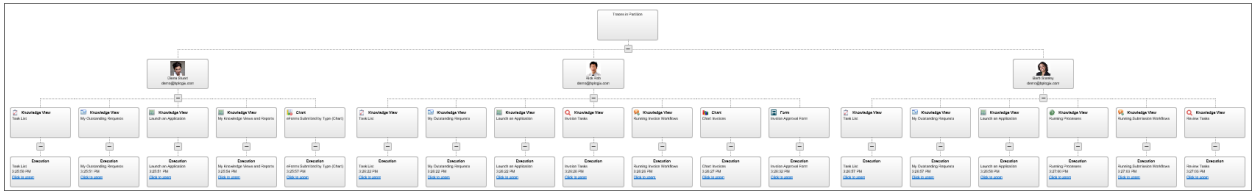


When we generate this Dashboard, the charts, with all of their underlying Business Values also run, which results in a trace stack that looks like this:



Each chart in the tree has a chart node below the user node. Below that, each chart node has an execution node that points to the underlying Business Value. You can zoom into each execution node to see the Business Value and the data it returns, just as we did in the first example above.

The Trace stack grows dynamically, so each time you perform another data operation, more traces are added. Similarly, if you choose to trace the entire system as an administrator, and choose to trace all users, your trace stack can become very large, very quickly, on a busy system. In the example below, we can see the stack trace generated by just three users in a few minutes.



You can zoom in or out of the trace, enabling you to view close-ups of different sections of the trace stack, by using the scroll wheel on your mouse, or by clicking the [CTRL] + [+] or [CTRL] + [-] keys on the keyboard. If you're zoomed in, you can use the horizontal and vertical scroll bars of the browser windows to move around and view different areas of the tree in your Trace stack.

## BP Logix Mobile Application

The BP Logix Mobile Application (Mobile App) component is a separately-licensed component for Process Director Cloud installations or on-premise installations that use the Subscription licensing model. The Mobile App component enables you to use a mobile application that supports both online and offline form submission on phones, tablets, and other mobile devices. Forms can be filled out by mobile users even when the user has no internet connectivity on the mobile device, and the mobile application will sync Form instances with the mobile server when internet connectivity is restored. Forms submitted via the mobile application are stored on the mobile server until they are imported into Process Director.

The Mobile App component provides access to the following features.

- Additions to the Form definition that enable the export of the Form definition to an Excel file.
- A mobile server into which:
  - The Excel file can be imported, and the form converted to a mobile-specific format, and
  - Datasources can be imported and used to fill dropdowns, set form data, etc.
- A Mobile Application that can be downloaded from the [Google Play Store](#) or [Apple App Store](#).

Once the license for the Mobile App component is purchased, administrators will be provided with a login to access the Mobile Server at <https://mobile.bplogix.com>. Additionally, mobile component custom variables will be provided for on-premise customers, or set by BP Logix for Cloud customers. These custom variables must be configured in the Custom Variable file prior to using the component, and are documented in the [Mobile Application Custom Variables](#) topic of the Developer's Guide.

## Mobile Application Users #

Once the Custom Variables for the component have been configured, access to the Mobile Application component is granted from the [User Administration](#) section of the [IT Admin](#) Area on the [Users](#) page. From the list of users, select the user you'd like to edit by clicking the [Edit](#) link to open the [User Profile](#) page for that user. Two properties listed in the [User Profile](#) page, [Mobile Admin User](#) and [Mobile App User](#), as

described in the [User Administration topic](#) of the System Administrator's Guide will, when checked, enable the user with appropriate access to the Mobile App.

To access the mobile server's administrative features, such as importing a Form or Data Source, a user must be specified as a **Mobile Admin User**. Access to the Mobile app is granted separately by checking the **Mobile App User** property. When these properties are checked, the users are automatically added to the mobile server with the appropriate level of access. In most cases, an administrator will need to have **both** the **Mobile Admin User** and **Mobile App User** properties checked.

Please note that, as a mobile administrator, you should **never** add or delete users from the mobile server directly. User access should be managed **only** from the User Administration page in Process Director, and not from within the mobile server itself.

## Importing a Form into the Mobile App Component #

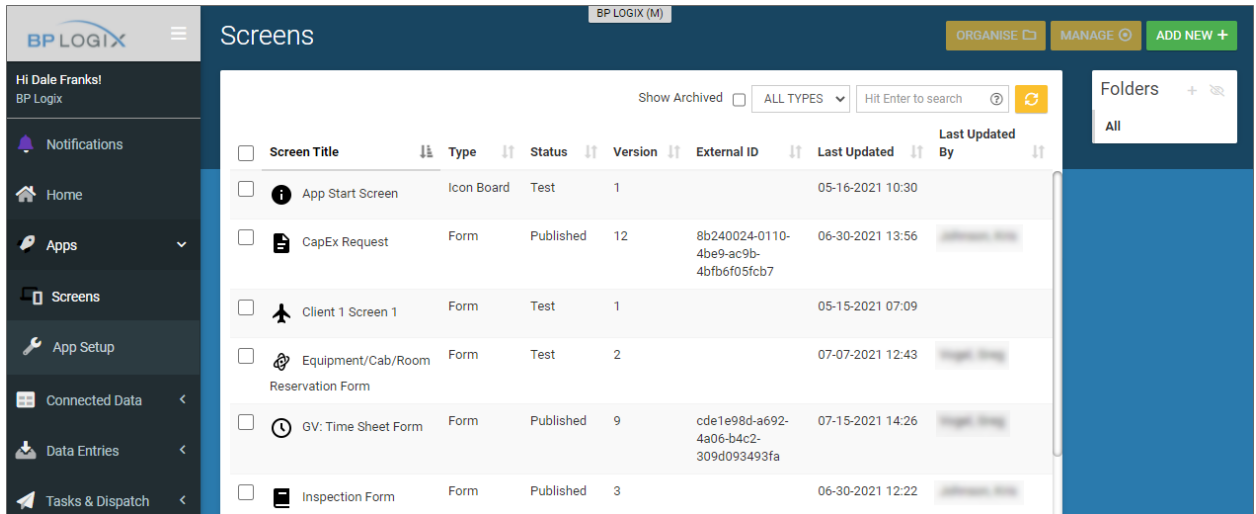
Form definitions for properly configured installations will display a new action link, labeled **Download Field List**, on the **Properties** tab of the Form definition.

The screenshot shows the 'Properties' tab for a form definition. The form name is 'Webinar Mobile Form Demo'. The description field is empty. The instantiated form name is '{FORM\_DEF\_NAME} Submitted On {CREATE\_DATE}'. The form fields are enabled by default. The entire form is read-only when the options and default styles are expanded. The form URL and ID are displayed. The 'Download Field List' link is highlighted with a red circle.

Clicking this link will create a new Excel spreadsheet that will be automatically downloaded to your local machine.

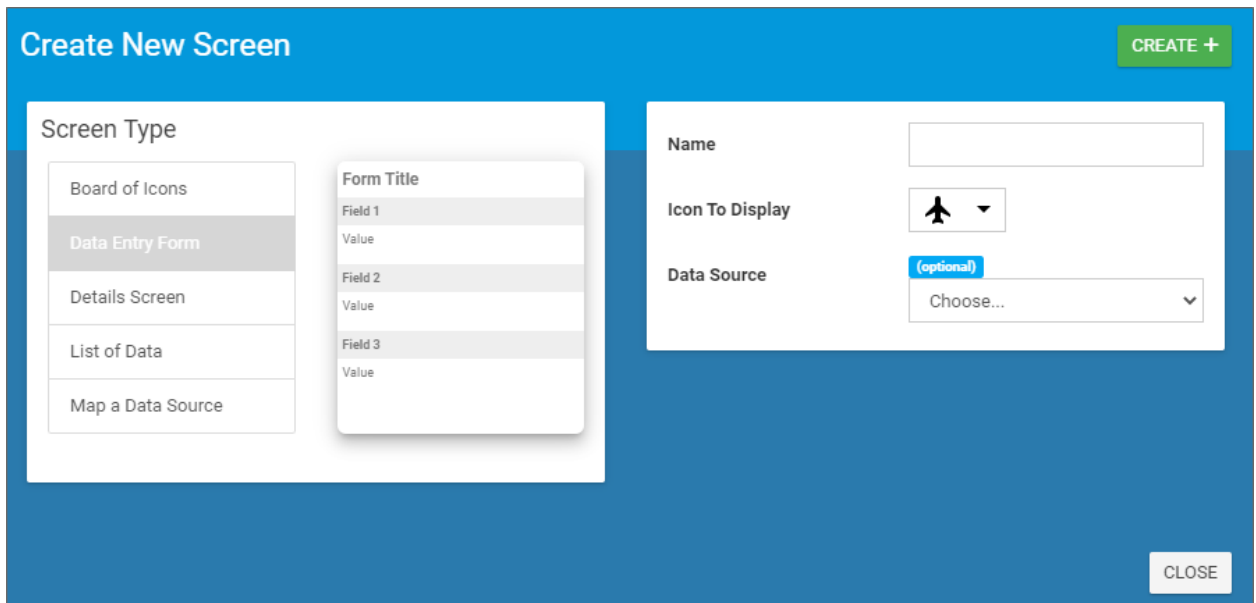
Using a web browser, navigate to the URL assigned as the mobile server URL for your installation, e.g., "https://mobile.bplogix.com".

The user interface for the BP Logix Mobile Server has a navigation menu on the left side of the screen, and displays the configuration page for the selected menu item on the right side of the screen. Please note that the menu items available to you on the left side of the screen may not display all of the items shown in the screenshots for this topic, based on your level of access to the mobile server.



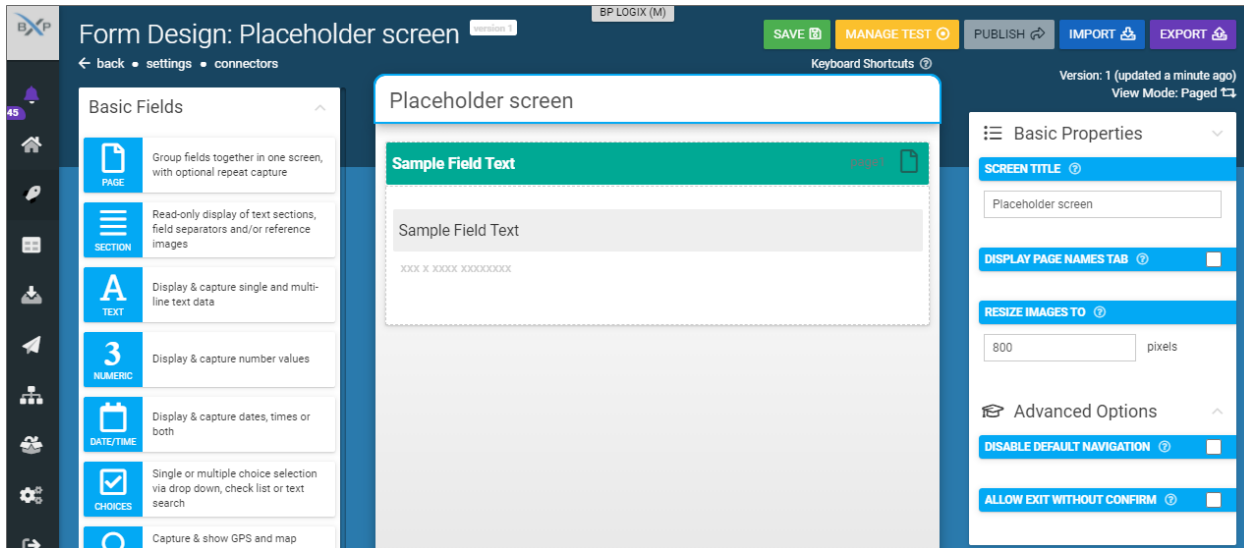
On the mobile server, each Form that you import from Process Director will be called a "Screen ". To create a new Screen, or to see a list of existing Screens, click on the **Apps** menu, then click the **Screens** sub-menu to display all of the form Screens that exist on the mobile server.

To import an Excel file for a new Screen, you must first create a new, blank Screen by clicking the **Add New** button at the top right corner of the screen. Clicking this button will open the **Create New Screen** dialog box.



On the left side of the dialog box, keep the default setting of **Data Entry Form**. On the right side, enter a **Name** for the new form. This name will be temporary if you are importing a Form, since the Screen's **Name** will be overwritten by the Process Director Form's name when the Excel file is imported. Next, select an icon to use as the Screen icon from the **Icon to Display** dropdown. Finally, if you wish to associate the new Screen with a data source (for instance, to fill a set of dropdown options), select the data source from the **Data Source** dropdown. We'll discuss importing data sources elsewhere in this topic.

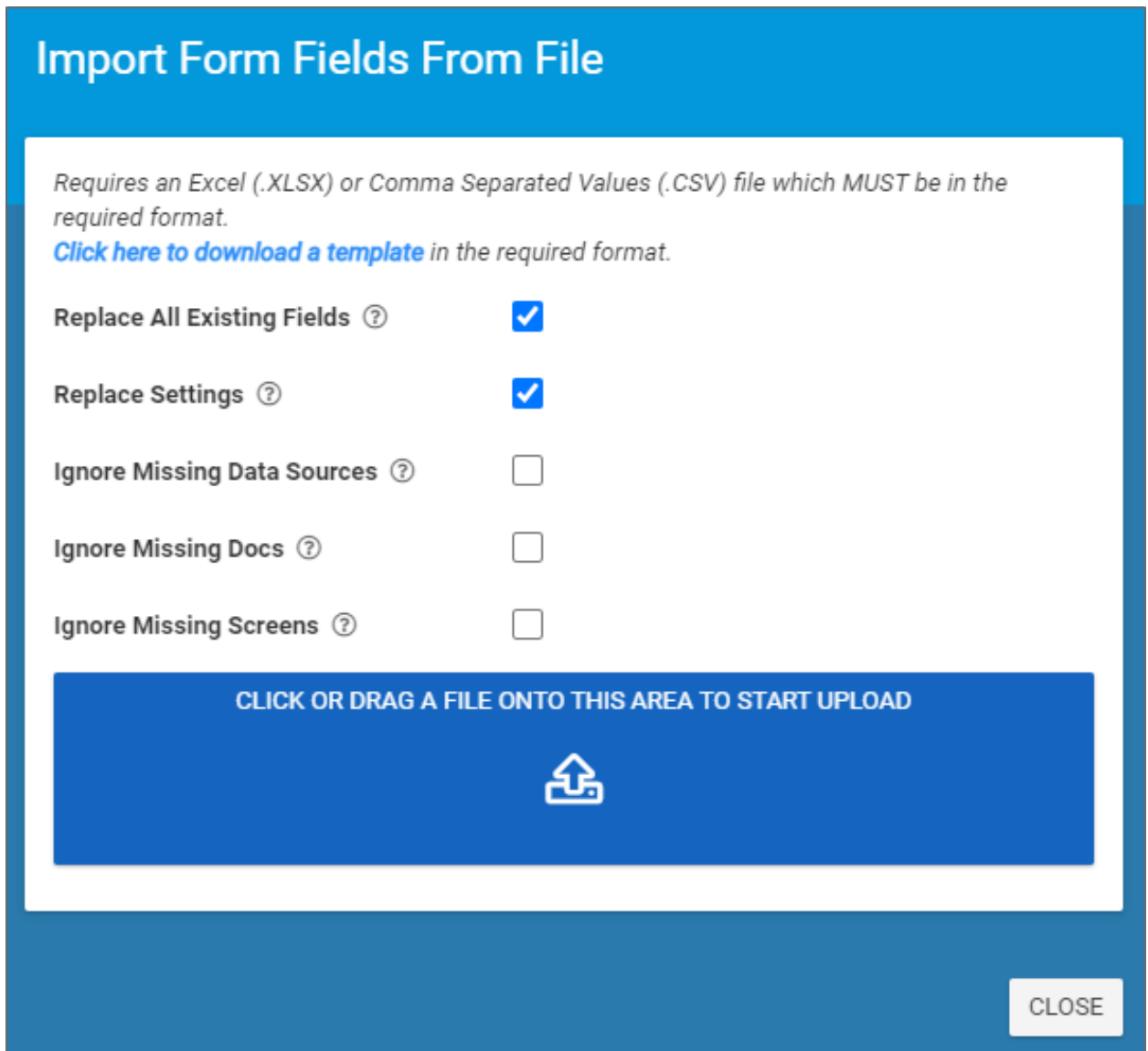
Once you've configured the placeholder for your Form, click the **Create** button to create the placeholder Screen. The new Screen will be created, and the Design view for the new screen will appear. For this example, we have named the Screen, "Placeholder Screen".



We are now ready to import the Excel file for our Form definition.

Click the **Import** button to open the **Import Form Fields From File** dialog box.

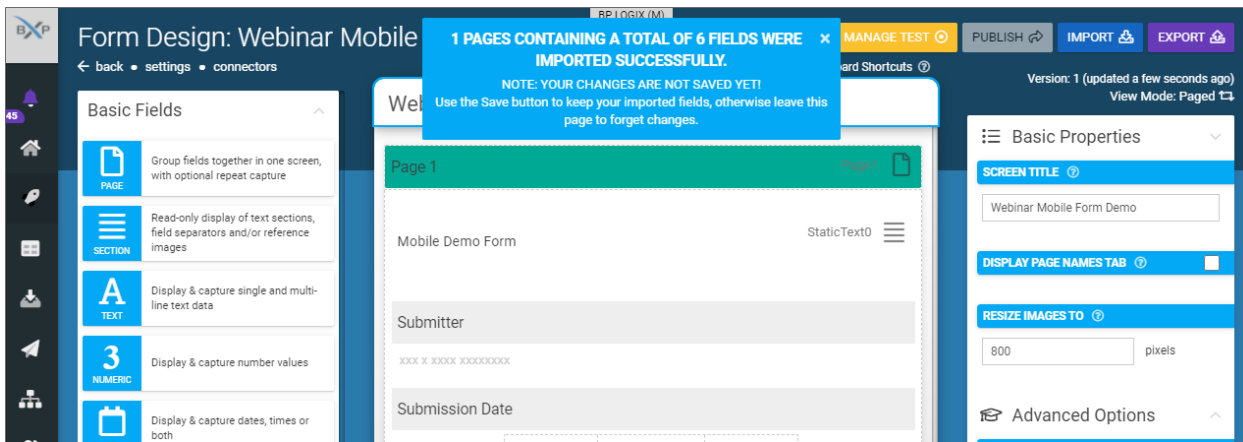




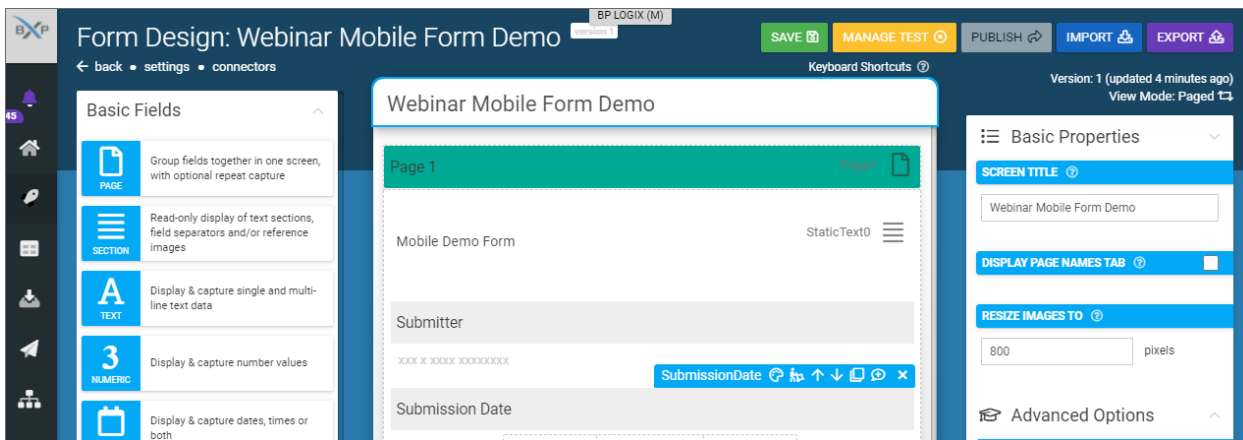
In the dialog box, ensure that both the **Replace All Existing Fields** and **Replace Settings** properties are checked. The **Replace All Existing Fields** property is checked by default, but you must manually check the **Replace Settings** property. The **Replace Settings** property is very important, because in addition to the form fields, the Excel file also contains the Process Director Form settings, including the Form ID, that must be imported into the mobile server. ***If you don't check this property, the correct form settings will not be imported into the Mobile Server.*** We'll address this again at the end of this section.

Next, click the button labeled, **Click or drag a file onto this area to start upload**. You can drag and drop files onto this button, or simply click the button and navigate to the Excel file using the **Open** dialog box.

Once the Excel file is selected, the mobile server will automatically import the Excel file, overwrite the existing settings for the Screen with the settings contained in the Excel file, and import the Form fields to the mobile Screen.

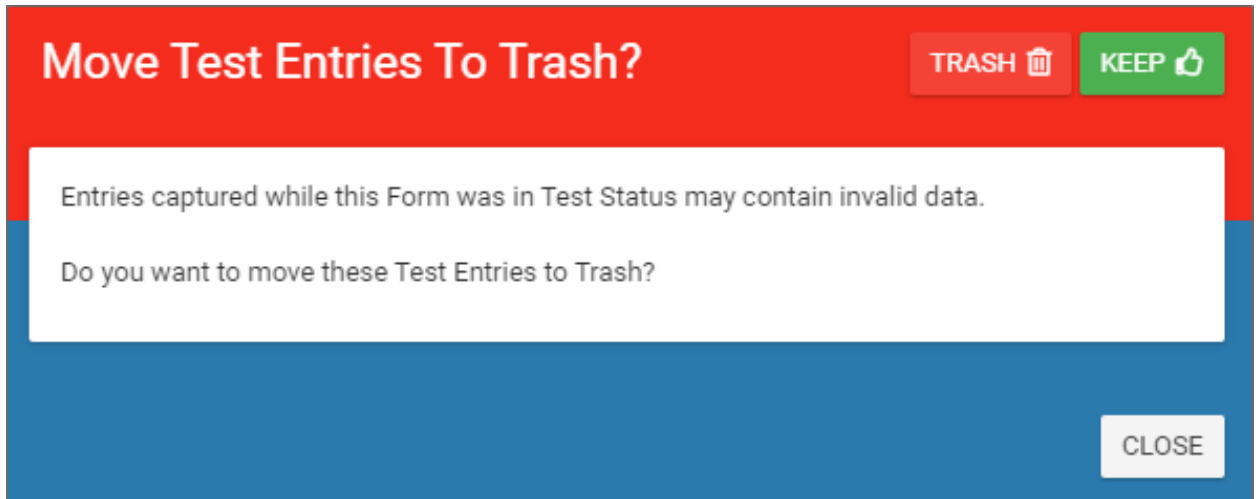


You now need to save the new Screen by clicking the **Save** button. The **Save** button will be hidden behind the notification banner that appears at the top of the page to notify you that the import was successful. You must close the banner to access the **Save** button.



Once the form is saved, you may, optionally, edit the Screen to change fonts, colors, apply data sources to fill dropdowns, etc. Editing options for any selected item on the screen are configured via the properties that appear in the column on the right side of the window. Once the Screen has been edited to your satisfaction, you can click the **Save** button to save your changes.

When you are ready to make the form available to users of the Mobile Application, click the **Publish** button. The **Move Test Entries To Trash?** dialog box will appear. You can publish a form multiple times to test and edit the form. This dialog box gives you the opportunity to send any existing Form entries to the trash bin.



You can choose to **Trash** or **Keep** the existing entries by clicking the appropriate button. Once published you can navigate back to the Screens window to see the new Screen, which has now been renamed, in this example, [Webinar Mobile Form Demo](#).

<input type="checkbox"/>	Screen Title	Type	Status	Version	External ID	Last Updated	Last Updated By
<input type="checkbox"/>	Mobile Test Form	Form	Published	5	d791a83c-1339-42e3-b05d-36ae5f476bd4	07-12-2021 09:29	
<input type="checkbox"/>	Mobile Test Form - old	Form	Published	1		07-07-2021 13:47	
<input type="checkbox"/>	Mobile Test Form 2	Form	Published	3	fe36de53-7ece-4f87-bd6b-47525220ef0b	08-09-2021 15:13	
<input type="checkbox"/>	Test Form - new	Form	Published	18	71713858-4714-41a6-ad20-73c7dfc89d42	07-21-2021 12:08	
<input type="checkbox"/>	Time Sheet Form OLD	Form	Test	8	02f487c1-b528-4a51-bb5d-f6def2a82a2e	08-05-2021 22:03	
<input type="checkbox"/>	Webinar Mobile Form Demo	Form	Published	1	7c67f825-b504-4b89-b9e7-f65e43f966d2	08-16-2021 13:22	Franks, Dale
<input type="checkbox"/>	Zeeba Mobile Test	Form	Test	2	f50c45ca-4622-	08-09-2021 13:38	

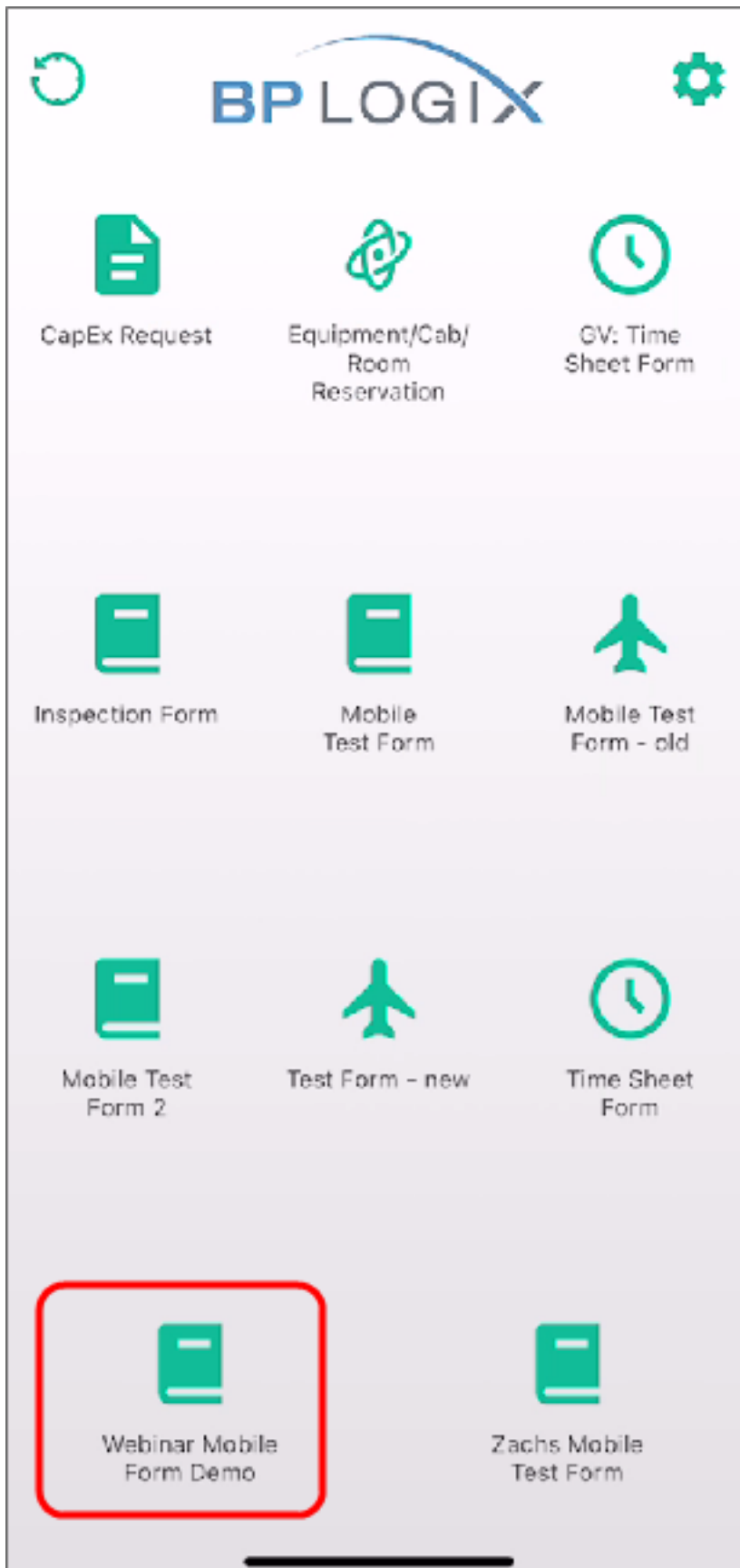
Now that the form has been published, users of the mobile application will be able to access it.

As mentioned above, the **Replace Settings** property will import the FormID from Process Director. This FormID will be displayed in the Screens page in the **External ID** column, as shown in the image above. ***The External ID must match the FormID of the Form in Process Director, or you won't be able to import the Form instances back into Process Director from the mobile server.*** The Form will work inside the mobile application, and submissions will be saved to the mobile server, but without the correct Form/External ID reimport to Process Director won't be possible.

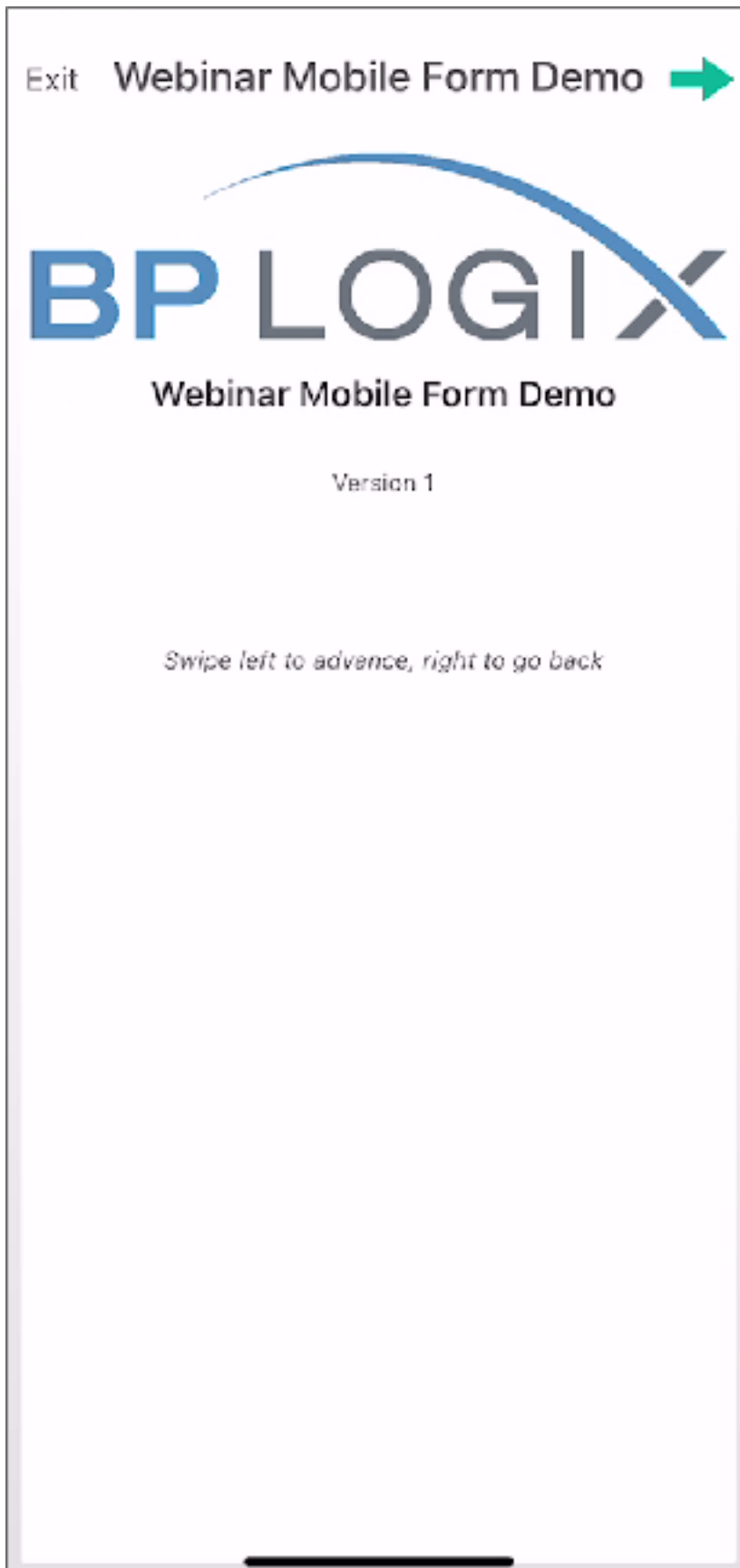
## Mobile Application #

The BP Logix mobile application is available for both Android and iOS mobile devices. When a new Screen is published, it will be synchronized to the mobile device of every user. This synchronization will occur the first time the user opens the mobile device in a place where they have Internet connectivity to access the mobile server. Assuming a user has mobile connectivity, the synchronization occurs when the mobile app is opened on the user's device. The amount of time it will take to synchronize depends on the amount of data that must be transferred.

When synchronization is complete, the new Screen will be available to the user's Mobile App.





Pressing the icon for the screen will open it.



You'll initially be shown a title page for the screen. You can advance forwards and backwards through the pages by swiping left or right, respectively. You can also press the arrows that appear at the top corners of the page. Swiping left will advance to the Form page, enabling you to fill out the Form.



Exit Webinar Mobile Form...  

**Page 1**

Mobile Demo Form


**Submitter**

Enter text...

**Submission Date**

June	14	2019
July	15	2020
<b>August</b>	<b>16</b>	<b>2021</b>
September	17	2022
October	18	2023

**Item Dropdown**



**Selection Text**

Enter text...

BR returns  $\geq 5$

After the form is filled out, swiping left again will take you to the submission page, from which you can submit the form by pressing the **Upload** button. Additionally, if your mobile device is properly configured and has access to a printer, you can click the **Upload & Print** button to both submit the form and print a copy of it.

Exit Webinar Mobile Form Demo ←



Tap below to complete and upload

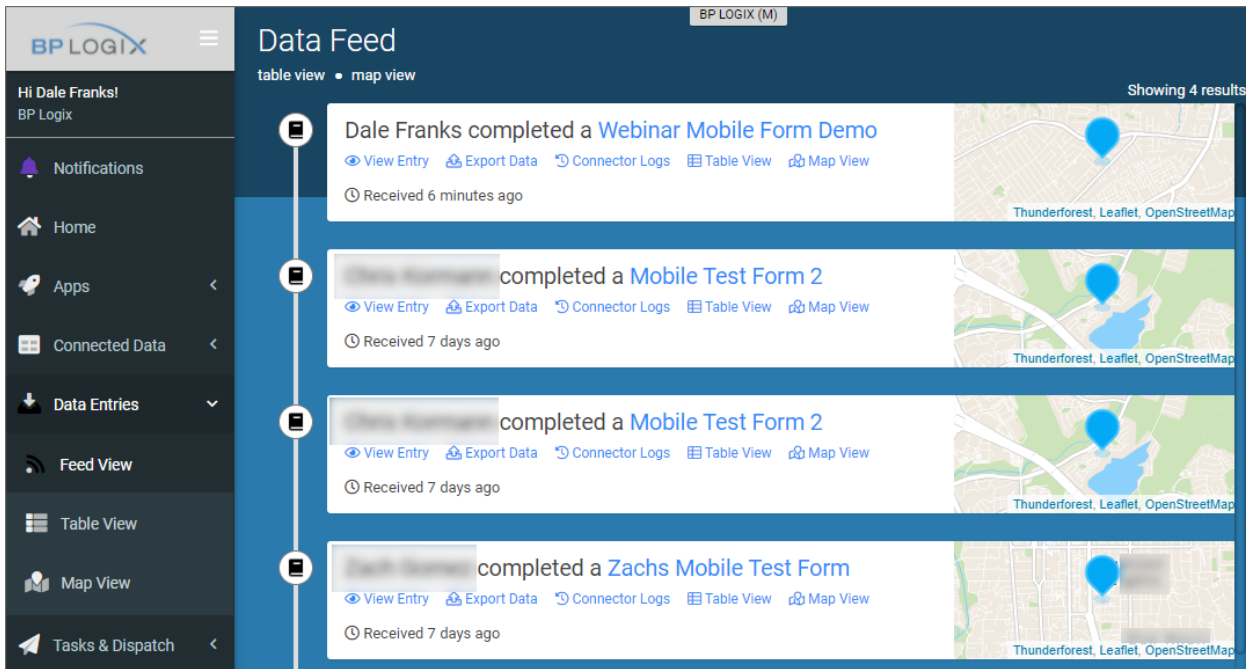
Upload

Upload & Print

If your mobile device has Internet connectivity, the submitted form will be instantly synchronized to the server. Otherwise, the synchronization will occur the next time the mobile application is running with Internet connectivity. The form submission will be stored on the mobile device, if the device is offline, until such time as Internet connectivity is restored.

## Importing Form Instances Back into Process Director #

The Mobile Server will store all form submissions it receives. You can see all stored Form submissions on the mobile server by selecting the **Data Entries** menu, then selecting the **Feed View** submenu.



These form instances will remain stored on the Mobile Server until they are imported into Process Director.

In Process Director, a new Custom Task, [Mobile Form Import](#), will import all of the Form instances from the Mobile Server to Process Director, where they'll appear as normal form instances. The Custom Task can be run manually or automatically, and requires no user configuration. Instead, the Custom Task uses the Configuration settings that are contained in the Custom Variables folder.

BP Logix recommends that you configure the Custom Task as the only Activity that runs in a Process Timeline, then create a Goal object that runs on a scheduled basis to call the Process Timeline every time the Goal runs. This will automatically import all form instances on the schedule you configure in the Goal.

**i** v2022.10.04 and higher for this Custom Task now import groups of images set using the gallery control on mobile devices.

## Importing Data Sources #

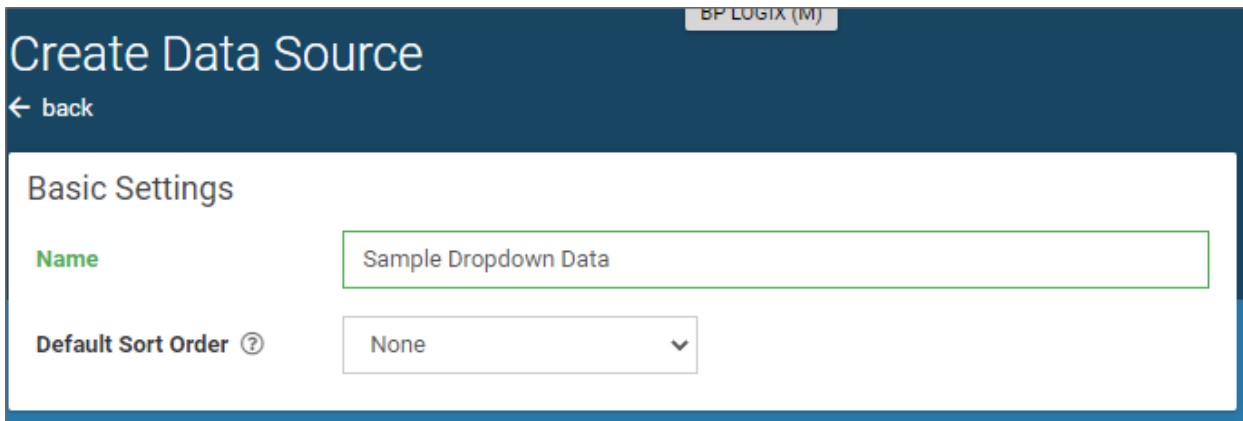
External data that you wish to use in a Screen must be imported to the mobile server as a Data Source. The easiest method for importing this data is to import it via an Excel spreadsheet. In this example, we'll import a simple Excel spreadsheet to use to fill a **Dropdown** control on the **Webinar Mobile Form Demo** Screen we imported in the previous sections, above.

	A	B
1	ID	Name
2	1	San Diego
3	2	Austin
4	3	Atlanta
5	4	Denver
6	5	Lexington
7		

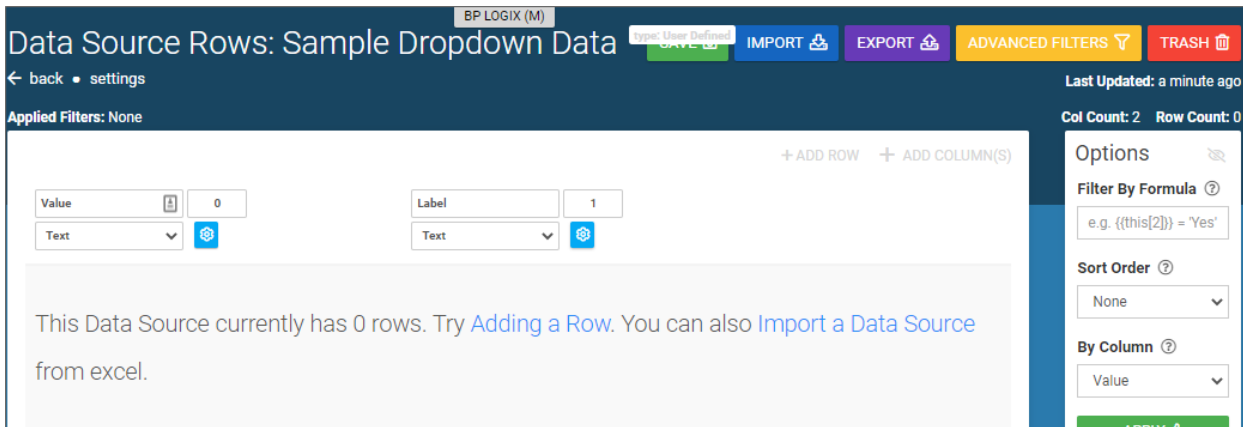
You can access the list of existing data sources on the Mobile server by navigating to the **Connected Data** menu item, then the **Data Sources** submenu.

<input type="checkbox"/>	Name	Type	External ID	Rows	Last Updated	Last Updated By
<input type="checkbox"/>	Active Users	System	ACTIVEUSERS	10	08-05-2021 18:50	
<input type="checkbox"/>	App: Entry Drafts	App	APPDRAFTS	1	10-06-2020 05:21	
<input type="checkbox"/>	App: Entry History	App	APPHISTORY	1	10-06-2020 05:21	
<input type="checkbox"/>	App: Incomplete Tasks	App	APPTASKS	1	10-06-2020 05:21	
<input type="checkbox"/>	Fruits	User Defined	BPL:2d5c1636-9f6c-4e8e-ab87-0e8e0d8d21c1	4	08-05-2021 14:12	
<input type="checkbox"/>	Published Docs	System	PUBLISHEDDOCS	1	10-06-2020 05:21	
<input type="checkbox"/>	States	User Defined		3	06-30-2021 12:21	
<input type="checkbox"/>	Test Data Source	Connected		16	06-30-2021 13:49	

To import a new data source, click the **Add New** button to open the **Create Data Source** screen.



Set a **Name** for the data source, which, in the case of this example, will be **Sample Dropdown Data**, then click the **Create** button, which will open the new, blank data source.



You can add rows manually to the data source, but in most cases, you'll want to import data, as we will in this example. Imported data sources must meet some required constraints to be imported:

- The Imported data must be in Excel (XLSX) or Comma-Separated Text (CSV) format.
- The first row of your file is assumed to be the data source's column headers and the rest are assumed to be data rows.
- The file must contain at least one row (excluding the header row) and two columns.
- The first column in all rows must be populated with unique values.

Clicking the **Import a Data Source** link will open the import screen.

## Import Rows From an Excel/CSV File

- The first row of your file is assumed to be the data source headers and the rest are just rows.
- The file must contain at least one row (excluding the header row) and two columns.
- The first column in all rows must be populated with unique values.

Requires an Excel (.XLSX) or Comma Separated Values (.CSV) file which MUST be in the required format.

[Click here to download a template](#) in the required format.

Auto Detect Column Data Types ?

CLICK OR DRAG A FILE ONTO THIS AREA TO START UPLOAD



CLOSE

Click the button labeled, [Click or drag of file onto this area to start upload](#). You can drag and drop files onto this button, or simply click the button and navigate to the Excel file using the [Open](#) dialog box. Assuming your Excel or CSV data meets the constraints specified above, the import will complete and the resulting data source will be displayed.

ID	Name
1	San Diego
2	Austin
3	Atlanta
4	Denver
5	Lexington

Click the **Save** button to save the data source information. You can navigate back to the **Data Sources** menu to see your new data source displayed in the list.

## Applying a Data Source to a Screen #

If you navigate to the **Screens** menu, you can hover your mouse over a Screen in the list to see editing options for that screen.

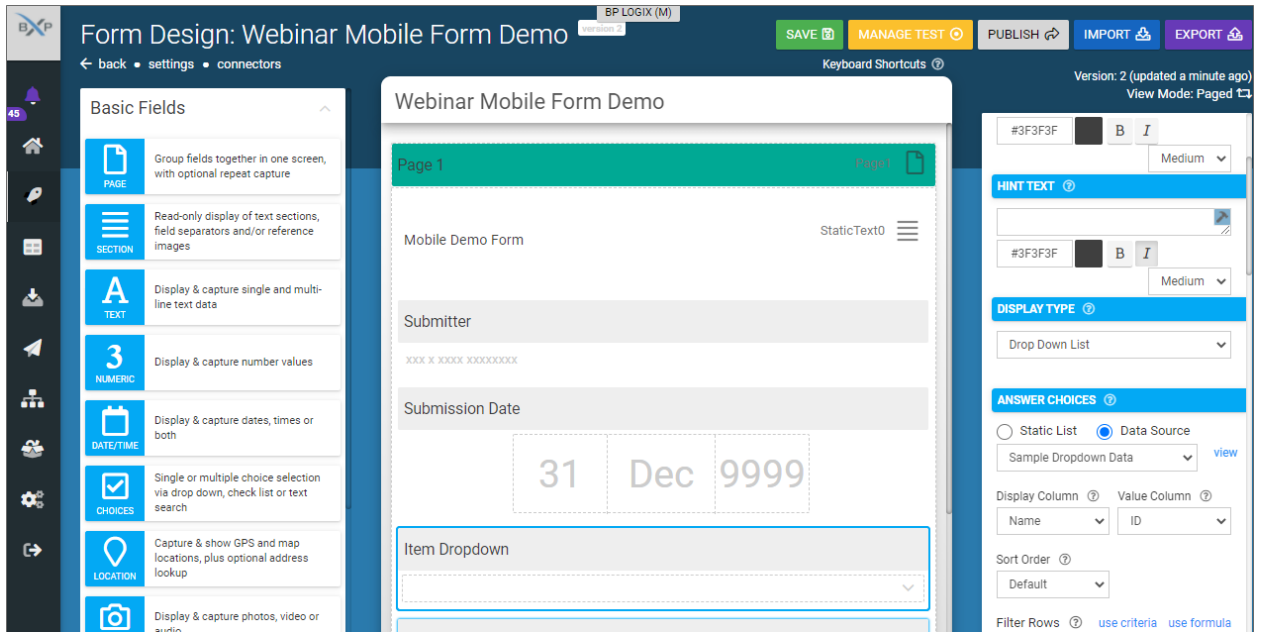
	Name	Type	Status	Version	ID	Date	User
<input type="checkbox"/>	Webinar Mobile Form	Form	Published	1	7c67f825-b504-4b89-b9e7-f65e43f966d2	08-16-2021 13:22	Franks, Dale

Demo

- settings
- design
- connect
- clone

Selecting the **design** option will open the screen in design view. Since we have already published this form, we will need to click the **New Version** button to create a new version of this form that we can edit. Once we do so, the Screen will open in a fully editable format.





This Screen contains a **Dropdown** control named **Item Dropdown**. Clicking the control will select it, and will display editing options for this control on the right side of the screen. We can scroll down through the editing options until we reach the section labeled, **Answer Choices**. We can apply our new data source to fill this **Dropdown** control, using the following procedure.

1. Select the **Data Source** radio button to indicate the options will be taken from a data source.
2. From the data source dropdown, select the data source to use for the options, which, in this case, is the **Sample Dropdown Data** data source we just created.
3. Set the **Display Column**, i.e., the data source column that will be used to display options to the users, by selecting the appropriate column from the dropdown.
4. Set the **Value Column**, i.e., the data source column that will be used to save the value of the dropdown, by selecting the appropriate column from the dropdown.
5. Optionally, select the **Sort Order** for the **Display Column**.

Once you've done this you can click the **Save** button to save your changes, then click the **Publish** button to publish the new version of the Screen.

Once published, the updated form will synchronize to users of the Mobile Application, after which the new dropdown options will be available on the mobile form.

Exit Webinar Mobile Form... ← →

Page 1

Mobile Demo Form

Submitter

Enter text...

Submission Date

June 19  
July 20  
August 21  
September 22  
October 23 2023

*Dropdown options  
added from data  
source*

Item Dropdown

Atlanta ▼

S Done

San Diego

Austin

Atlanta ]

## Configuring Azure for Process Director Integration

Microsoft Modern Authentication (an OAuth-based authentication system) provides much more secure access to SharePoint, SMTP email, and other Azure services from Process Director, but does require a complex setup process. To set up Modern Authentication between Azure and Process Director, you must complete the following steps.

1. [Create a certificate](#) to authenticate Process Director with Azure.
  - a. Using Microsoft's certreq.exe, installed on all modern Windows OS versions.
  - b. Using PowerShell, also included with all modern Windows OS versions.
2. [Add Process Director as a Registered Active Directory application](#) in the Azure Active Directory portal.
  - a. Add the public key certificate to the Process Director application in Azure.
  - b. Configure the appropriate Azure settings.


In this topic, we'll address each of these required steps in detail. Additional information about this topic can also be obtained from [Microsoft's online documentation](#).

 You cannot configure any OAuth settings for SharePoint Datasources or SMTP Email in Process Director until you have created and registered an Azure Active Directory Application in Azure by completing the steps described in this topic.

### Create a certificate to authenticate Process Director with Azure #

Microsoft prefers the use of certificates for authentication. Each certificate includes both the public and private keys used to encrypt data. The public key (in a CER file) is used by SharePoint Online to authenticate Process Director. The private key is packaged in a password-protected PFX file and is used by Process Director to authenticate with Azure Services. There are two methods that can be used on Windows-based systems to create a proper certificate.

- Using Microsoft's certreq.exe, installed on all modern Windows OS versions.
- Using PowerShell, also included with all modern Windows OS versions.

 Keep in mind that certificates expire after a set period of time. Most organizations specify the maximum length of time certificates should be used. By default, the instructions that follow will generate certificates valid for one year. You should, therefore, generate and install new certificates well before existing certificates expire. This implies that your organization also has a mechanism in place to be reminded when expiration is approaching, to ensure that service interruptions don't occur.

### Creating a Certificate with certreq.exe

This method of certificate creation might be preferred if you're less comfortable with command-line operations and don't intend to automate the generation of certificates. [Microsoft's online documentation](#) has

additional information about certreq.exe.

## Instructions

First, using a text editor like Notepad, copy and paste the following text into a new document:

```
[Version]
Signature = "$Windows NT$"

[Strings]
szOID_ENHANCED_KEY_USAGE = "2.5.29.37"
szOID_KEY_ENCIPHERMENT = "1.3.6.1.5.5.7.3.1"

[NewRequest]
Subject = "cn=BP Logix Process Director"
MachineKeySet = false
KeyLength = 2048
HashAlgorithm = Sha1
Exportable = true
RequestType = Cert

KeyUsage = "CERT_KEY_ENCIPHERMENT_KEY_USAGE"
; The following values can be changed to generate certificates that expire
; sooner or later depending on your organizations needs. The default is 1 year.
ValidityPeriod = "Years"
ValidityPeriodUnits = "1"

[Extensions]
%szOID_ENHANCED_KEY_USAGE% = "{text}%szOID_KEY_ENCIPHERMENT%"
```

Once you've done so, save the document as an INF file in a folder on your system, e.g., `c:\Users\Some.User\Documents\PD Certificate\CertReq.inf`.

Open the Windows Command Prompt. You can press the [WINDOWS] key, type "cmd", then select the "Command Prompt" application.

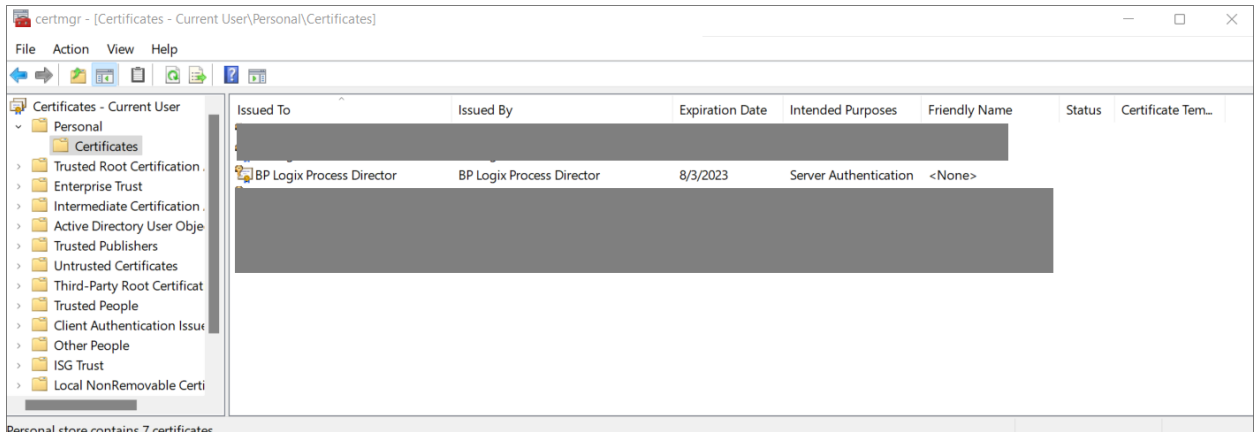
In the Command Prompt, open the directory in which you installed the INF by using the `cd` command, and the folder path to the INF file, then pressing the [ENTER] key. Using the example above, you'd need to type:

```
cd c:\Users\Some.User\Documents\PD Certificate\
```

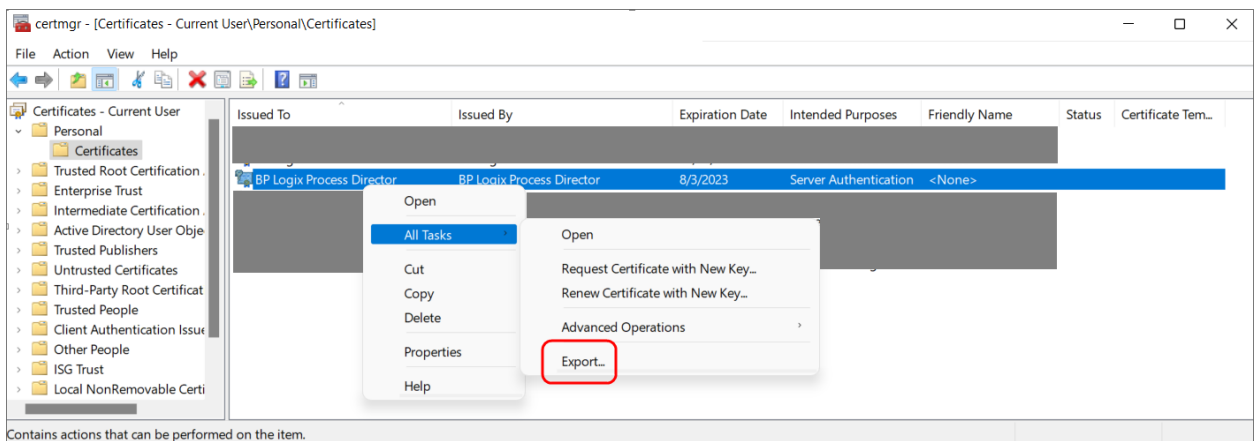
Once the directory changes, type the following and press the [ENTER] key to run the certreq application.

```
certreq -new certreq.inf PublicKey.cer
```

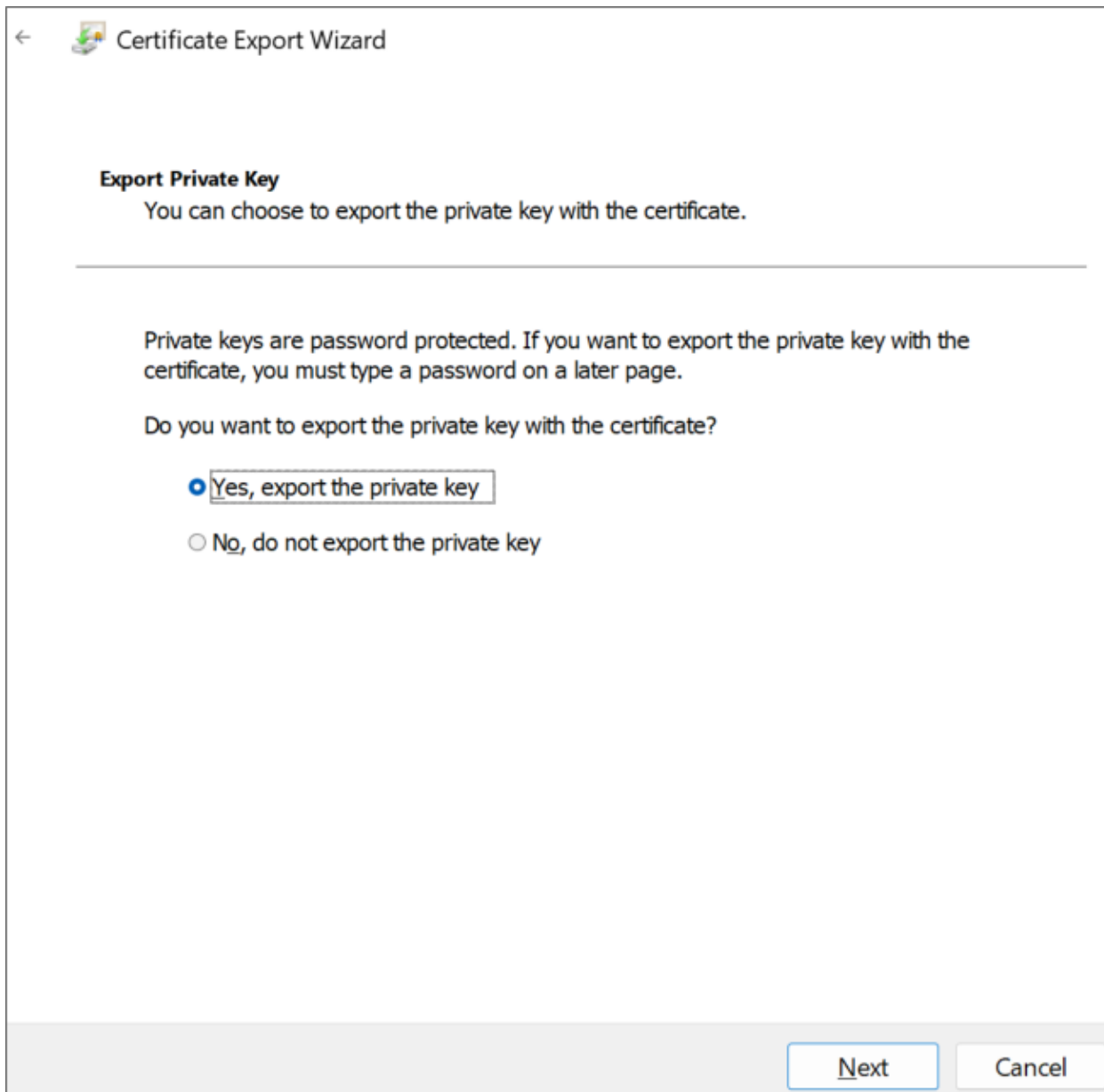
Running the certreq application will create the certificate, and add it to the Windows Certificate Manager. To continue, you'll need to open the Certificate Manager to access the new certificate. To open the Certificate Manager, you can press the [WINDOWS] key, type "certmgr", then select the "Manage computer certificates" option. When the Certificate Manager opens, you'll need to navigate to the `Personal\Certificates` folder, where you should see the certificate issued to and by BP Logix Process Director.



Right-click that certificate and then select **All Tasks > Export**.

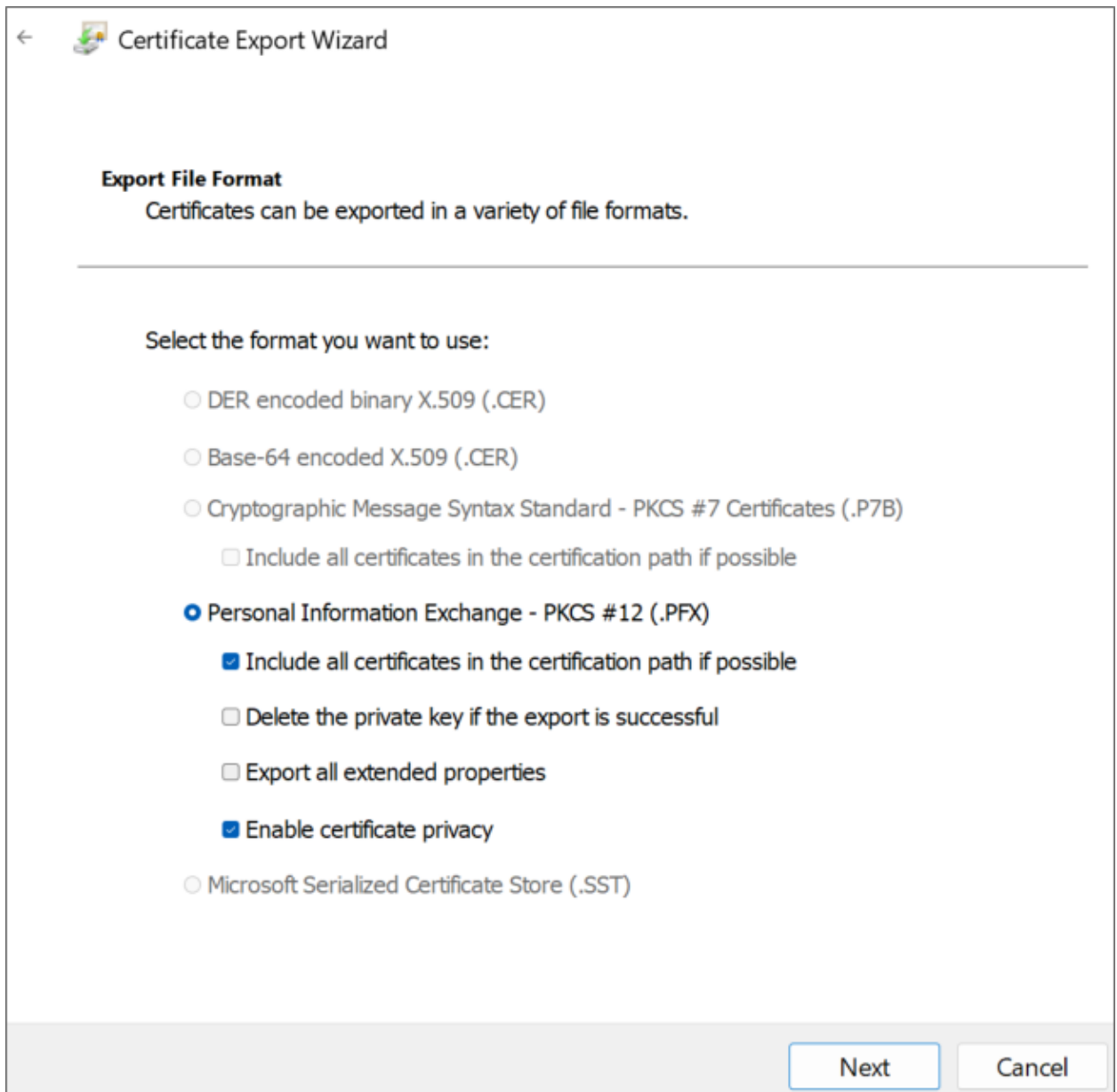


The Certificate Export Wizard will open. On the first screen, click the **Next** button. On the **Export Private Key** screen, select **Yes, export the private key**, then click the **Next** button.





On the **Export File Format** screen of the Wizard, Ensure that you select the following options:

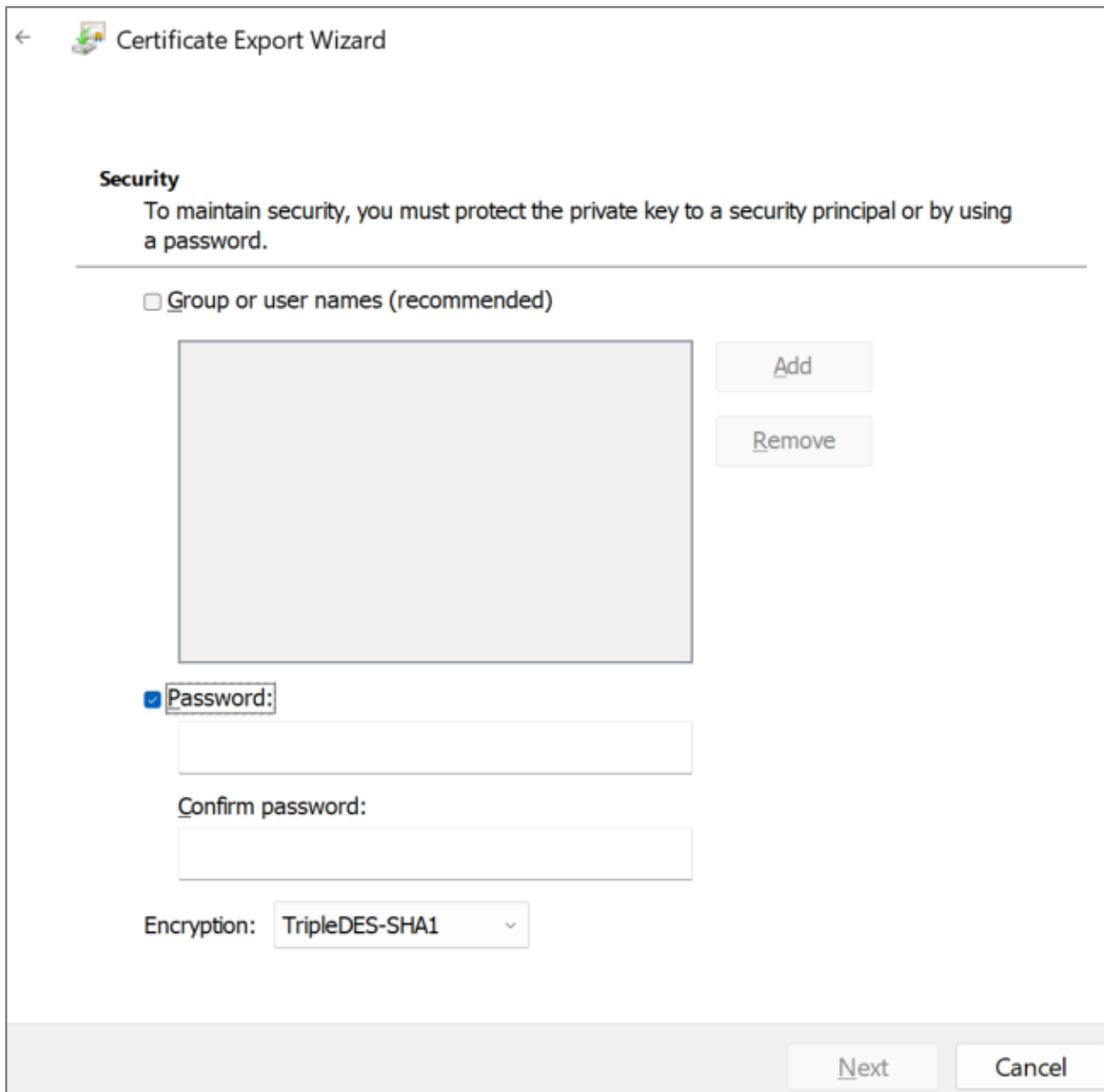
- Personal Information Exchange - PKCS #12 (.PFX)
- Include all certificates in the certification path, if possible
- Enable certificate privacy



On the **Security** screen, check **Password** as the security protocol and enter a password twice.

 Be sure to store this password securely, you'll need it in future steps.

 Be sure to use a long, sufficiently complex password in line with your organization's cryptographic standards.



← Certificate Export Wizard

**Security**  
To maintain security, you must protect the private key to a security principal or by using a password.

---

Group or user names (recommended)

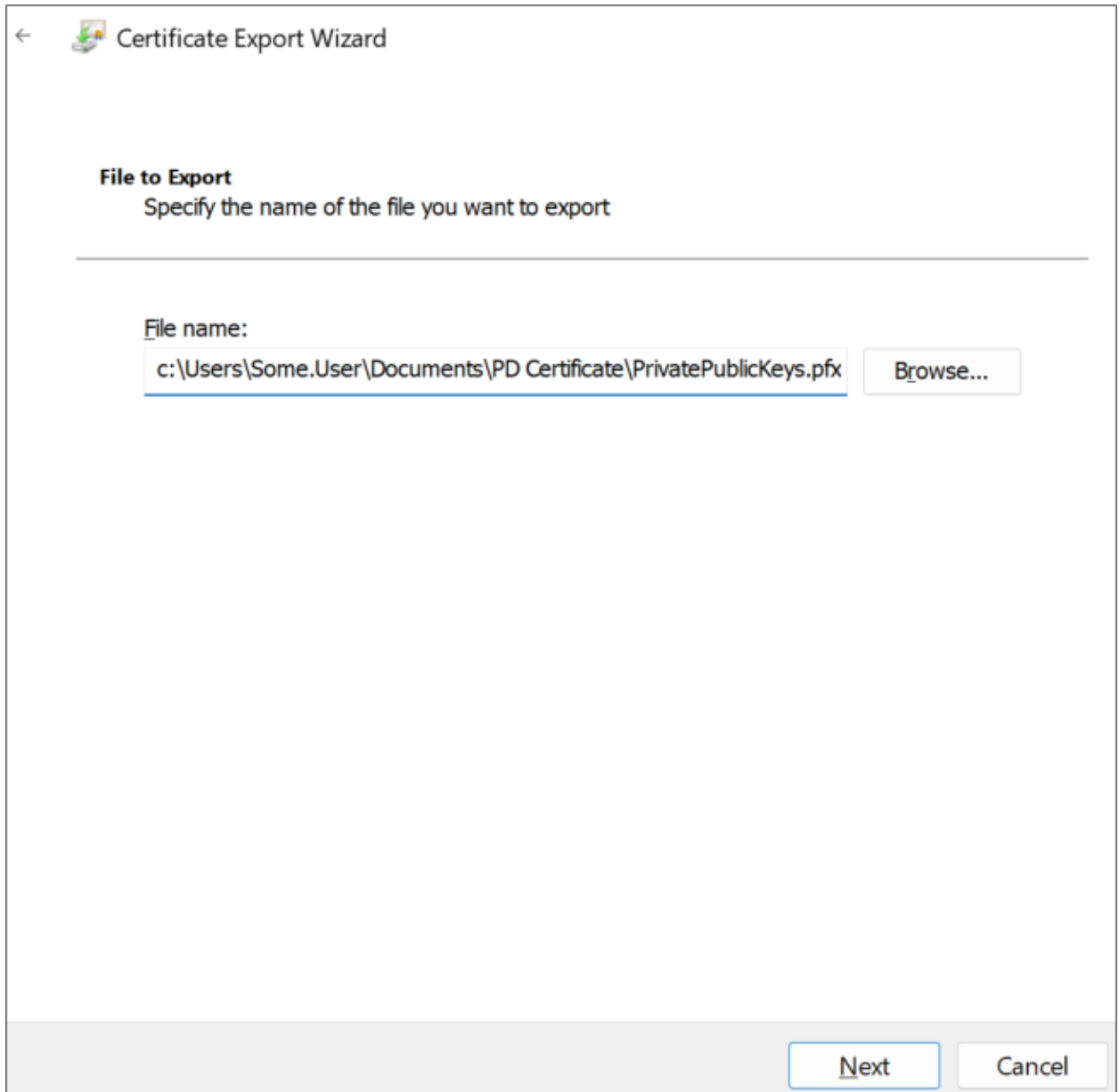
Password:

Confirm password:

Encryption: TripleDES-SHA1 ▾

On the **File to Export** screen, store the resulting PFX file in the same folder as you stored the CertReq.Inf and PublicKey.Cer files, then click the **Next** button.





Click the **Finish** button on the next Wizard screen, then **OK** to finish the Wizard and close it.

BP Logix recommends that you remove the certificate installed in the Certificate Manager by right-clicking it and then selecting **Delete** followed by **Yes** to delete it in the confirmation dialog.

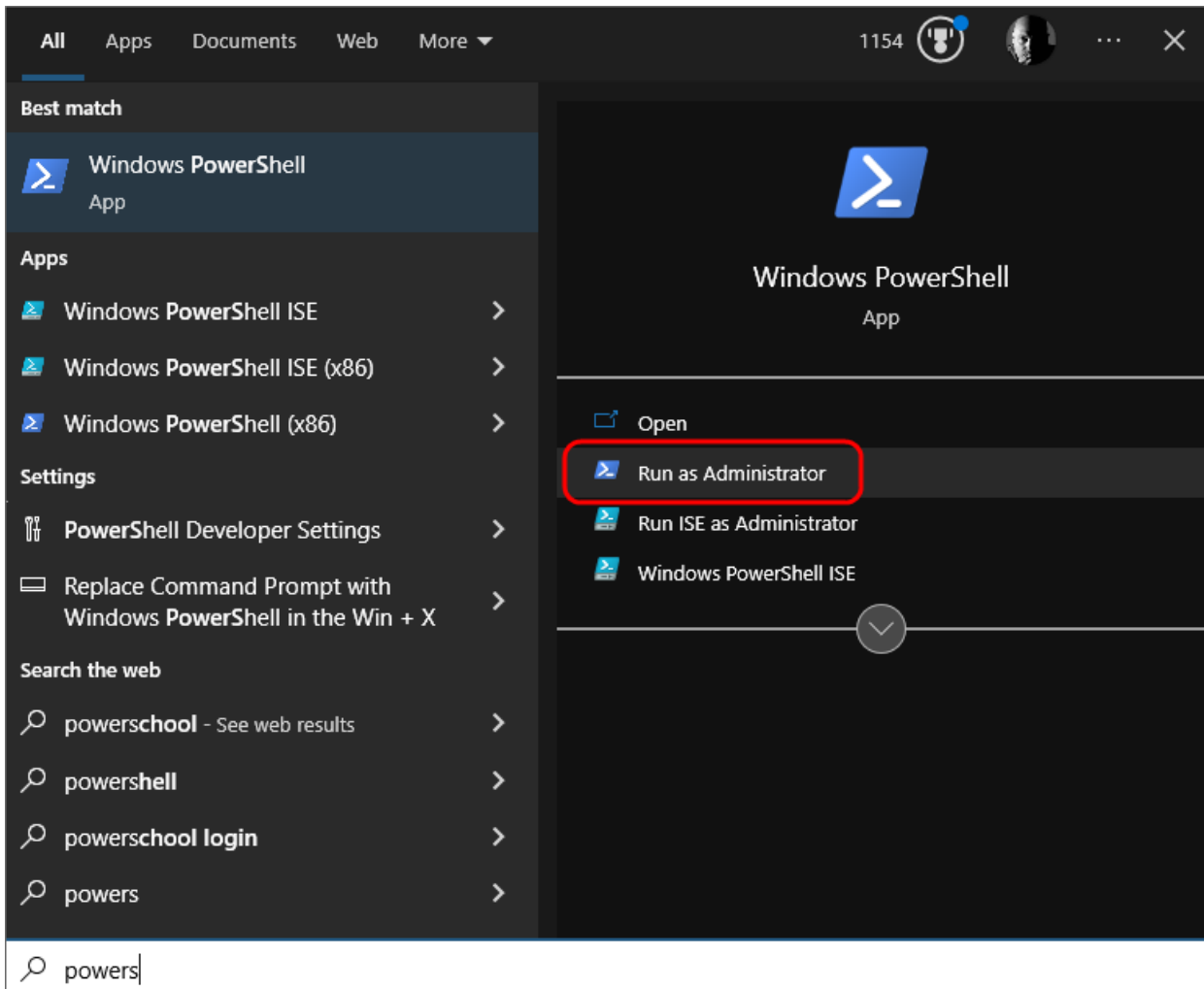
Keep both the `PublicKey.cer` and `PrivatePublicKeys.pfx` files handy for subsequent steps in this setup process. You should also archive them in a secure, backed up location as well.

## Creating a Certificate with PowerShell

PowerShell is a command line application that's included with all modern versions of Windows. You can choose this method if you're comfortable with PowerShell and might want to automate certificate generation on a recurring basis.

## Instructions

Open PowerShell by pressing the [WINDOWS] key, typing "PowerShell" then selecting the **Run as Administrator** option to open Windows PowerShell.



In PowerShell, create or navigate to the directory you'd like to use to store the certificate files. Once you're in the desired directory, run the following command:

```
$cert = New-SelfSignedCertificate -CertStoreLocation Cert:\LocalMachine\ -  
KeyUsage KeyEncipherment  
-KeyAlgorithm rsa -KeyLength 2048 -subject "BP Logix Process Director"  
-DnsName "BP Logix Process Director" -Type SSLServerAuthentication  
-TextExtension @("2.5.29.37={text}1.3.6.1.5.5.7.3.1")
```

Next, run these commands in PowerShell, replacing `<password>` with a password of your choosing. Ensure the password is cryptographically secure, in accordance with your organization's standards. Be sure to also store this password securely, as you'll need it in future steps.

```
$pwd = ConvertTo-SecureString -String '<password>' -Force -AsPlainText  
$path = 'cert:\LocalMachine\My\' + $cert.Thumbprint
```

Finally, run these commands to create the .PFX and .CER files. Modify the `<path>` value to store the file in a location of your choosing.

```
Export-PfxCertificate -cert $path -FilePath <path>\PrivatePublicKeys.pfx -Pass-  
word $pwd  
Export-Certificate -cert $path -FilePath <path>\PublicKey.cer
```

Keep both the PublicKey.cer and PrivatePublicKeys.pfx files handy for subsequent steps in this setup process. You should also archive them in a secure backup location as well.

## Add Process Director to Azure #

To add Process Director as an application in your Azure Active Directory portal at the Tenant level, complete the steps below after signing into your Azure portal ([portal.azure.com](https://portal.azure.com)):

### 1. Register Process Director as an Application

- A. If you have access to multiple tenants, use the Directories + subscriptions filter in the top menu to switch to the tenant in which you want to register the application.
- B. Search for and select **Azure Active Directory**.
- C. Under **Manage**, select **App registrations** > **New registration**.
- D. Enter a display **Name** for your application, e.g., “Process Director”. This name can be changed later, if needed.
- E. Specify who can use the application. Typically, only accounts in this organizational directory should be used. See the Microsoft documentation titled [Quickstart: Register an application with the Microsoft identity platform](#) for more information.
- F. Add the **Redirect URI**, which is the URI for your Process Director installation, e.g., `https://my-org.bplogix.net`.
- G. Click the **Register** button to register the application.

### 2. Add Your Public Key Certificate

To add your public key certificate to the Process Director application in Azure, complete the steps below.

- A. In the Azure portal, in **App registrations**, select the Process Director application you created previously, e.g., “Process Director”, as in step 1D, above.
- B. Select **Certificates & secrets** > **Certificates** > **Upload certificate**.
- C. Select the PublicKey.cer file you created earlier.
- D. Upload the certificate file to Azure.

Your AAD Application should now be properly registered and secured with a certificate.

## Conclusion

Congratulations! Assuming that you've correctly followed the instructions above, you've now configured an Azure Integration with Process Director. To complete the integration, you'll need to perform some additional, specialized configuration in Azure, depending on whether you're trying to:

- [Create a Sharepoint data source](#) or
- [Set up SMTP email access on the Properties page](#) of the **IT Admin** area's **Installation Settings** section, using the "Office365/Microsoft OAuth" **SMTP Authentication Type**.

## SharePoint Data Sources

With the implementation of Microsoft's move to **Modern Authentication**, using the Microsoft identity platform, logging into cloud-based versions of SharePoint is no longer possible by simply using a user name and password. Legacy installations that use older versions of SharePoint may still do so, but SharePoint has largely implemented an OAuth-based authentication scheme, with additional security provided by the use of encryption certificates.

In Process Director v5.44.1000, Modern Authentication for SharePoint was implemented using the [SharePoint OAuth](#) Datasource, which only gives access to SharePoint at the Tenant (organizational) level.

For Process Director v5.44.1103, The [SharePoint OAuth](#) Datasource was renamed to [SharePoint OAuth \(Tenant\)](#), while a new Datasource [SharePoint OAuth \(Site\)](#), was added to give access to SharePoint at the Site level, rather than at the entire tenant.

The existing [SharePoint](#) Datasource, which uses the simple username/password authentication scheme, is still available for customers who are using older versions of SharePoint. This legacy authentication method should be relevant to only a very small minority of customers, and has been renamed to [SharePoint Legacy](#).



This update to the SharePoint Datasources will require updating the SharePoint Custom Tasks!

## Configuring a SharePoint OAuth (Tenant) Datasource #

Modern Authentication provides much more secure access to SharePoint, but does require a more complex setup process. To set up Modern Authentication between SharePoint and Process Director, you must first create and register an Azure Active Directory (AAD) application. The System Administrator's Guide has instructions for creating the AAD application in the [Configuring Azure for Process Director Integration](#) topic.

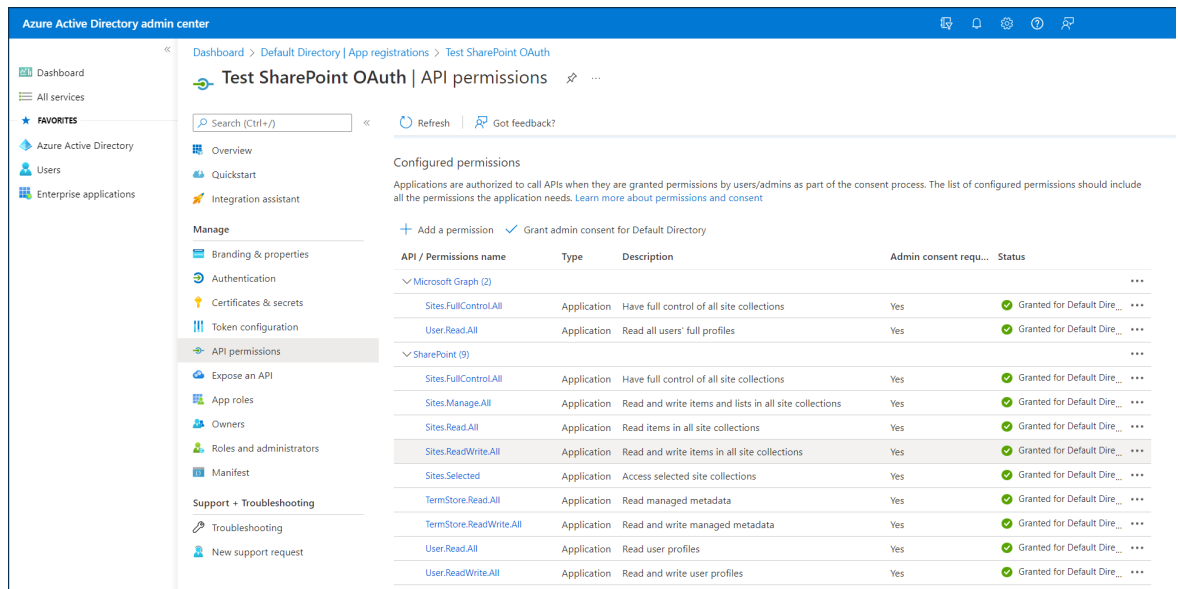
Once you've created the AAD Application, you can begin the process for configuring SharePoint Online.

## Configure SharePoint Online permissions #

To configure the AAD application to use SharePoint with Process Director, you'll need to perform the following configuration steps:

1. If you have access to multiple tenants, use the Directories + subscriptions filter in the top menu to switch to the tenant in which you want to register the application.
2. Search for and select **Azure Active Directory**.
3. Under **Manage**, select **App registrations**, then select your Process Director application. In this example, we'll use "Test SharePoint OAuth" as the AAD Application name, though, of course, the name you use may vary.

4. Click **API permissions**.
5. Click **Add a permission** and add all permissions displayed below to the **SharePoint** section of the **API Permissions** area:

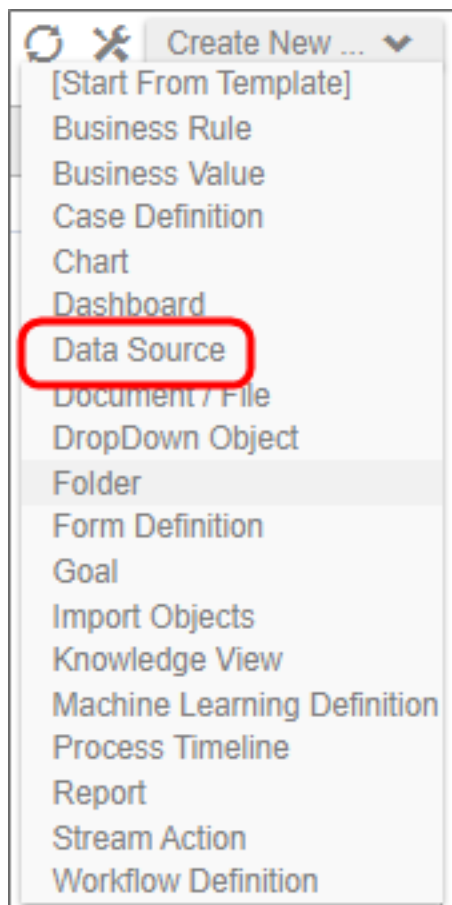


## Create the SharePoint OAuth (Tenant) Datasource #

Now that the application has been fully registered in Azure, and the appropriate SharePoint API permissions have been set, you can create the SharePoint OAuth Datasource in Process Director. Be sure to keep the Azure window open, however, as you'll need to transfer some information from Azure to configure the SharePoint OAuth Datasource. Ensure you've opened the **Azure Active Directory admin center** window to the **Overview** tab of the **App registrations** page of your Process Director integration app. In this example, we'll use the "Test SharePoint OAuth" application we used in the steps above.

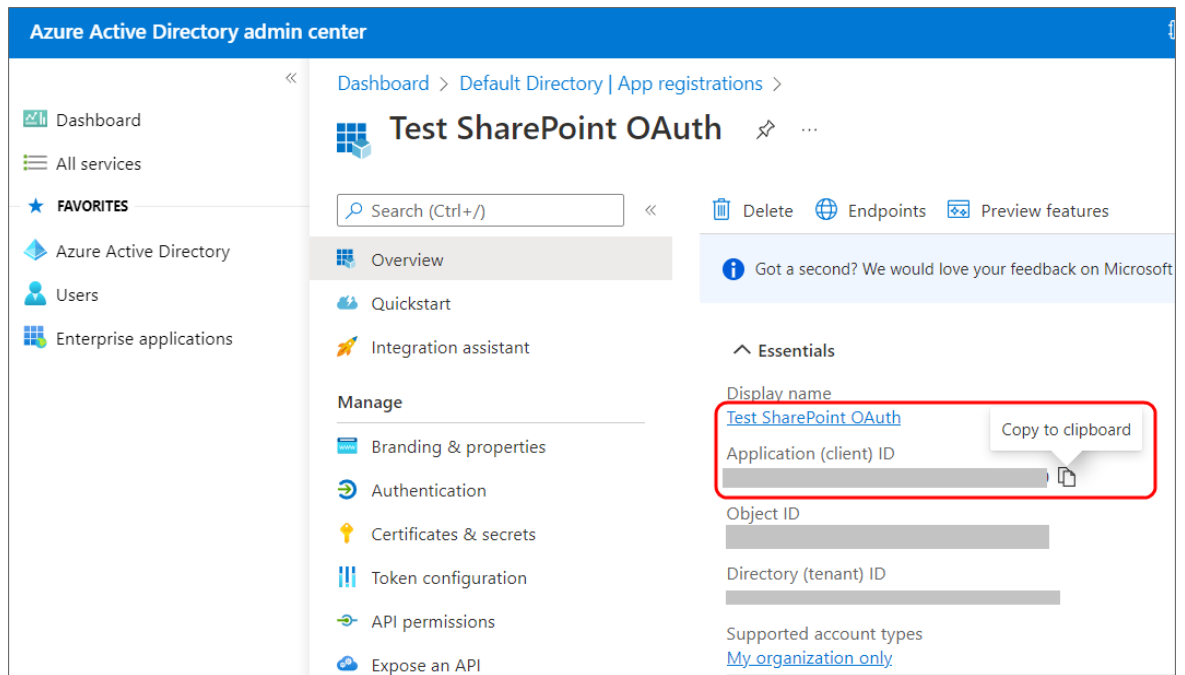
### Instructions

1. Navigate to the Process Director folder in which you want to store the new Datasource, then select **Data Source** from the **Create New** menu.

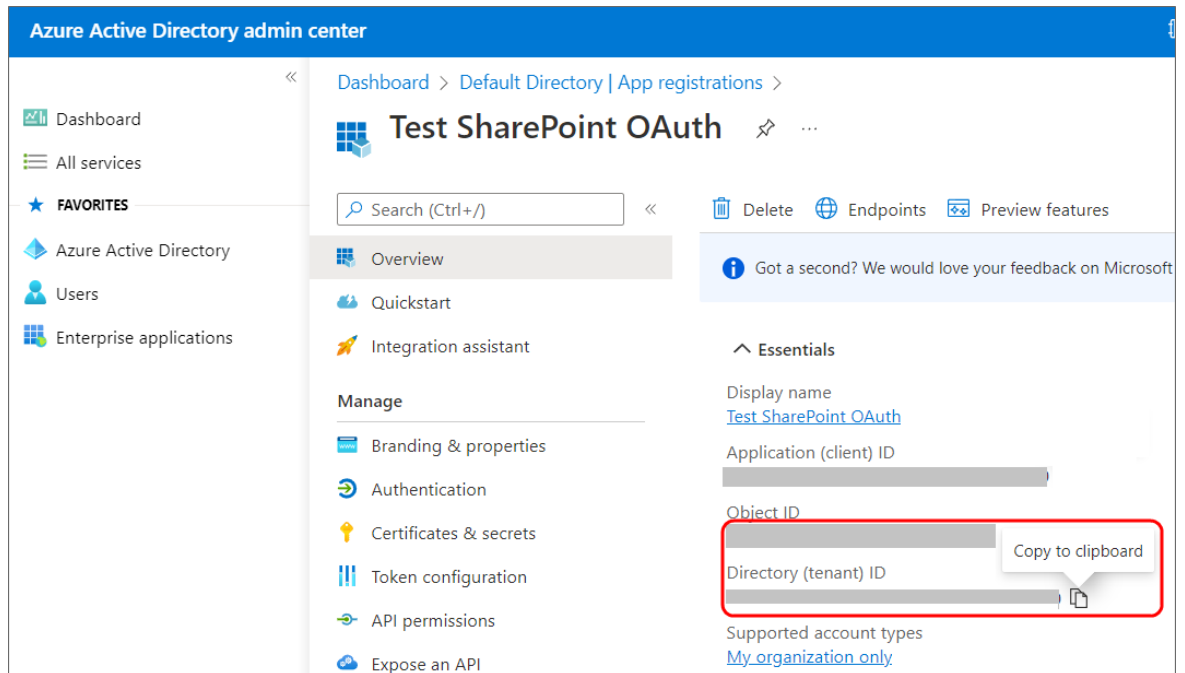


2. In the **Create New Data Source** screen, enter an **Name** for the Datasource, then click the **OK** button to create the Datasource and open its configuration screen.
3. On the **Properties** tab of the Datasource definition, change the **DataSource Type** to "SharePoint OAuth (Tenant)".
4. Set the **SharePoint Site URL** to the URL your SharePoint Online server.
5. To set the **Client ID** property, go to the Azure window, and using the "Copy to Clipboard" icon, copy the value in the **Application (client) ID** property, then paste it into the **Client ID** Property of the

Datasource definition.




6. Similarly, you'll need to copy the value of the **Directory (tenant) ID** property in Azure to the **Tenant ID** property of the Datasource definition.



7. To set the certificate to use for this Datasource, click the **Browse** button for the **SharePoint Certificate File** property, then locate and select the PrivatePublicKeys.pfx file you created earlier (either with certreq.exe or PowerShell).
8. Enter the certificate **Password** that you created for the PrivatePublicKeys.pfx file.

9. Click the **OK** button to save your changes, then update the Datasource definition by selecting **Update** from the **OK** dropdown menu at the upper right corner of the page.
10. Click the **Test Connection** button to ensure that the Datasource can connect properly to SharePoint.

## SharePoint OAuth (Tenant) Datasource Properties

Datasource Connection Name Icon 

SharePoint Datasource

---

### PROPERTIES

**Description**

Enter a brief description of this Object

**Datasource Type**

SharePoint OAuth ▼

**Sharepoint Site URL**

Client ID

Tenant ID

**Sharepoint Certificate File (\*.pfx)**

Browse

**Certificate Password (optional)**

In addition to the standard **Description** property, setting the **Datasource Type** property to *SharePoint OAuth* enables configuration of the connection properties listed below.

### SharePoint Site URL

The fully-qualified URL that connects to the SharePoint installation.

### Client ID

The value of the **Application (client) ID** property contained in the App Registration screen in Azure.

### Tenant ID

The value of the **Directory (tenant) ID** property contained in the App Registration screen in Azure.

### SharePoint Certificate File

A **Content Picker** than enables you to browse to and upload the certificate (.PFX) file to Azure.

### Certificate Password

The password that you configured for the certificate (.PFX) file when you created it.



## Configuring the SharePoint OAuth (Site) Datasource #

Configuring the [SharePoint OAuth \(Site\)](#) Datasource is far less complex than configuring the tenant-level Datasource, and requires no certificate to be created or uploaded to Azure. To add Process Director as an application in your Azure Active Directory portal at the Site level, complete the steps below after signing into your Azure portal ([portal.azure.com](http://portal.azure.com)):

### 1. Configure SharePoint Site Permissions

1. Navigate to the site you want to configure access for in your tenant. This is typically of the form <https://mytenant.sharepoint.com>, replacing “mytenant” with the appropriate name.
2. Adjust the URL to [https://mytenant.sharepoint.com/\\_layouts/15/appregnew.aspx](https://mytenant.sharepoint.com/_layouts/15/appregnew.aspx).
  - a. Click the buttons to generate both a **Client Id** as well as a **Client Secret**.
  - b. Select the **Client Id** value, copy the text and store the value somewhere safe to be used in later steps in this guide.
  - c. Select the **Client Secret** value, copy the text and store the value somewhere safe to be used in later steps in this guide.
3. Now you need to grant permissions to newly registered app (AKA principal). Navigate to [https://mytenant-admin.sharepoint.com/\\_layouts/15/appinv.aspx](https://mytenant-admin.sharepoint.com/_layouts/15/appinv.aspx). It’s important to note the addition of “-admin” to your site's normal name.

Office 365 Admin

SharePoint admin center

site collections

infopath

user profiles

bcs

term store

records management

search

secure store

apps

sharing

settings

configure hybrid

device access

**App Id and Title**  
The app's identity and its title.

App Id:

Title:

App Domain:   
Example: "www.contoso.com"

Redirect URL:   
Example: "https://www.contoso.com/default.aspx"

**App's Permission Request XML**  
The permission required by the app.

Permission Request XML:

- a. Add your Client Id as **App Id**.
  - b. Add the XML as shown, reproduced here to aid in copy and paste. Note, there are other, more restrictive options that can be considered listed in Table 1 at Microsoft's documentation topic, [Add-in permissions in SharePoint](#). Be careful using other values as it may prevent Process Director from working correctly.

```
<AppPermissionRequests AllowAppOnlyPolicy="true">
  <AppPermissionRequest Scope=
e="http://sharepoint/content/sitecollection" Right="FullControl" />
</AppPermissionRequests>
```
  - c. Set the **Title** to "Process Director".
  - d. Set **App Domain** to the fully qualified domain name of your Process Director deployment.
  - e. Set the **Redirect URL** to the URL of your Process Director deployment.
4. Click **Create**.
  5. Click **Trust It** in the follow-up prompt.

## 2. Configure the Datasource

1. In a Process Director **Content List** folder, select **Data Source** from the **Create New** menu.
2. Supply a **Name** and click **OK** to open the new Datasource definition.
3. Set the **Datasource Type** drop-down to "SharePoint OAuth (Site)".
4. Add the **SharePoint Site URL** for your SharePoint Online installation.
5. Add the **Client ID** (AKA Application Id) and **Client Secret** from SharePoint that you set aside in the steps for **Configure SharePoint Site Permissions** above.
6. Click **OK** then select the **Update** item from the **OK** menu at the top right corner of the page to save the configuration.
7. Click the **Test Connection** button to test your connection to the SharePoint site.

A successful test means that your Datasource is correctly configured and is connecting to the SharePoint site correctly.

## Conclusion

Congratulations! Assuming that you've correctly followed the instructions above, you've now configured both SharePoint Online and Process Director. You can now use this Datasource and the SharePoint Custom Tasks in Process Director to integrate your SharePoint sites and data with Process Director.

## Sharepoint Legacy Datasource #

For connections to pre-OAuth versions of SharePoint, the SharePoint Legacy datasource type enables you to create a datasource connection to the SharePoint server.

The screenshot displays the configuration window for a new datasource. On the left is a vertical toolbar with icons for settings, linking, cloning, saving, and confirmation. The main area is titled 'PROPERTIES' and contains the following fields:

- Datasource Connection Name:** A text box containing 'SP Test' with an 'Icon' button (cylinder icon) to its right.
- Description:** A large text area with the placeholder text 'Enter a brief description of this Object'.
- Datasource Type:** A dropdown menu currently set to 'SharePoint Legacy'.
- Sharepoint Site URL:** An empty text box.
- User ID:** An empty text box.
- Password:** An empty text box.
- Domain:** An empty text box.

At the bottom of the form is a button labeled 'Test Connection' with a document icon.

There are four properties to configure to create this datasource.

The **Sharepoint Site URL** property enables you to enter the fully qualified URL of the Sharepoint server to which you wish to connect.

The **User ID** must be the user ID for a valid SharePoint User, while the **Password** property will be the password for the specified user. The **Domain** property is the SharePoint domain that contains the specified user.

Once you've configured the datasource, you can click the **Test Connection** button and a message banner will appear, notifying you whether the connection was successful.

## Other Datasource Types

To see more information about different [Datasource Types](#) and their configuration, please refer to the following topics:

- [Common Datasources](#)
- [Excel Datasources](#)
- [File Datasources](#)
- [Social Datasources](#)

## Microsoft OAuth for SMTP

To configure integration between Azure and Process Director, you'll first need to create and register an Azure Active Directory (AAD) Application, if you do not have one. Please see the [Configuring Azure for Process Director Integration](#) topic for instructions on how to create and register an AAD Application.

Once the AAD Application has been registered, you'll need to perform some additional configuration to the AAD Application's settings in Azure.

First, in the **Authentication** area, you'll need to set the **Allow public client flows** property to: **Yes (On)**

Unfortunately, there are many factors that might impact the remaining AAD Application settings you'll need to use. Since that is so, you may wish to reference Microsoft's explanation of [SMTP OAuth implementation](#).

Depending on your Azure installation, as well as your organization's policies, there are different configuration settings that you might need to implement, in order to enable your AAD application to enable Process Director to use OAuth to send mail messages. **BP Logix cannot, therefore, definitively describe what settings might be required to make your Azure installation accept OAuth authentication, as we have no knowledge of, or access to, your Azure configuration.**




We strongly recommend that you refer to the Microsoft documentation topic on this subject: [How to set up a multifunction device or application to send emails using Microsoft 365 or Office 365](#).

We can provide some common configuration suggestions that have worked for our customers in the past, though ***we cannot guarantee that these settings will work with your specific Azure configuration.***

1. If it's available for your Azure installation, in the [Office 365 Exchange Online](#) section of the [API Permissions](#) area, you can set the permissions `SMTP.AccessAsUser.All`. This setting is not available for all installations. This setting seems to have been deprecated for recent installations of Azure, in lieu of #2, below.
2. In the [Office 365 Exchange Online](#) area, enable the `SMTP.SendAsApp` property. You may also need to enable `IMAP.SendAsUser.All` to true.
3. In the [Microsoft Graph](#) section of the [API Permissions](#) area, you can enable the following permissions: `Microsoft.Graph.DelegatedSMTP.Send` and `DelegatedUser.Read`.
4. For more comprehensive email access, you can set `Microsoft.Graph.DelegatedIMAP.AccessAsUser.All`.

If no combination of the settings above work for you, you may need to contact your Microsoft Azure technical support representative to assist you with configuring the correct AAD App permissions for your installation.

 For more information on authentication permissions, please refer to the [Microsoft Graph Permissions Reference](#) from Microsoft. Please be aware that BP Logix has an extremely limited ability to assist you with troubleshooting your Azure installation or settings.

Once configured, you'll need to get the following properties from the AAD Application's settings to transfer to the corresponding OAuth settings for the "Office365/Microsoft OAuth" **SMTP Authentication Type**, which is found on the [Properties page](#) of the [Installation Settings](#) section of the [IT Admin](#) area.:

<b>SMTP Authentication Type</b>	Office365/Microsoft OAuth ▼
<b>SMTP Tenant ID</b>	<input type="text"/>
<b>SMTP Client ID</b>	<input type="text"/>
<b>SMTP Secret</b>	<input type="text"/>

1. **SMTP Tenant ID**
  - a. The ID of the Azure Tenant in which the AAD Registered App resides (Creation of an AAD Registered App requires the existence of a Tenant)
  - b. The Tenant ID is displayed as the **Directory (tenant) ID** property on the **Overview** page of your AAD Application in Azure, but this value will also be displayed following `login.microsoft.com/...` in the Endpoint URLs that the App references
2. **SMTP Client ID**
  - a. The ID of the AAD Registered App
  - b. This value is displayed as the **Application (client) ID** property on the **Overview** page of your AAD Application.
3. **SMTP Secret**
  - a. The client secret or application password the administrator created to use with the AAD Registered App.
4. **UserID/Password**
  - a. Some installations may require that you provide a valid **UserID** and **Password** to connect to an email account on your system for sending mail messages, as part of the authentication.

Some Azure configurations may also be configured to require a specific email address be used to send *all* mails as the "From" email address. In that case, you will *at least* need to go to the [Global Variables page](#) and set the **Workflow From Email Address** property to the email address you've specified in Azure. You

may also wish to set that email address for the [Registered Email](#) property on this page ([Properties](#)), as a backup to the [Global Variables](#) setting.



Be advised that, with this configuration, ALL email addresses sent from the system MUST use the specified email address as the From address. This means that any custom email addresses you configure elsewhere, such as the "From Email" property of a Email Data control in an email template, *will not send email messages.*

# Index

---

## A

Accelerator Library 767  
Accessibility 339, 372, 691, 699, 701, 710, 714, 720, 723  
Accessibility Canada 699, 701, 710, 714, 720, 723  
Accessible 699, 701, 710, 714, 720, 723  
Actions Controls 322  
Activity Check 814  
Activity Comments 303  
Activity Configuration 567, 572, 582-583, 596-598, 600-604, 608, 612  
Activity emails 359  
Activity Instance 612  
Activity Log 320  
Activity Participants 583  
Activity Results 583  
Activity Settings 572, 582, 596, 600-604, 612  
Activity Tabs 572, 582, 596, 600-604, 612  
Activity Types 572, 582, 596, 600-604, 612  
Add Documents 386  
Add Icons 281  
Add images 350  
Add Pictures 350  
Add Report 625, 636  
Add Row 378  
Add URL 349  
AddRow 418  
Administer 841  
Administer timeline 612  
Administration 841  
Advanced Reporting 625, 636  
Advanced Reports 625, 636  
AI 528  
Americans With Disabilities Act 699, 701, 710, 714, 720, 723

Annotatons 237  
Annotations 227  
Anonymous Users 823  
App Stubs 767  
App Template 767  
Application Samples 16  
Application Stubs 767  
Application Templates 767  
Approval Comments 303  
Approver Comments 303  
Array 376, 420  
Array Controls 376  
Array End 376  
Array Rows 379  
Array Start 376  
ArrayColumn 418  
ArrayEnd 376  
ArrayMoveDown 420  
ArrayMoveUp 419  
ArrayRemoveRow 419  
Arrays 376  
ArrayStart 376  
Artificial Intelligence 528  
Assistive Devices 699, 701, 710, 714, 720, 723  
Asynchronous Operations 566  
Attach 421  
Attach KView 396  
Attach Objects 386  
Attach Objects with KView 396  
AttachDrop 422  
AttachKView 396, 422  
Attachment Groups 400  
Attachments 227, 237, 386, 400  
AttachObject 386  
AttachPolling 424



Attribute 545, 549, 554  
Attribute Picker 317  
AttributePicker 317  
Attributes 317  
Audit Button 356  
Audit Control 356  
Audit Viewer 741  
AuditButton 356  
Auditing 356  
Automate Process 642  
Automated Processes 642  
Automatic Form Creation 642

## B

Basic Controls 289  
Basic Form Controls 289  
Basics 12  
Best Practices 691, 734  
Bootstrap 372  
bpEmailImport 806  
bpImport 797  
bpImportEmail 806  
BR 82  
Branch Activity 603  
Branch Tab 603  
Branch Type 603  
Building 691  
Business Rules 82  
Business Value 89-90, 103, 174, 178, 182, 186, 195, 203, 871, 880  
Business Values 89-90, 103, 174, 178, 182, 186, 195, 203, 871, 880  
Button 332, 424  
Button Area 323  
ButtonArea 425  
BV 89-90, 103

## C

Cache 176, 220  
Caching 176, 220  
Calculate 426  
Calculate Dates 353  
Calculate Field 352  
Calculate Form values 352  
Calculation Control 352  
Calculations 352  
Cancel 427  
Cancel Form 326  
Cancel Loop 604  
Cancel Timeline 326  
Cancel Workflow 326  
Captcha 335  
Case 113, 118  
Case Management 108, 113, 118  
Categories 316  
Category 316, 545, 549, 554  
Category Picker 316  
CategoryPicker 316  
Cell Properties 455  
Change Forms 601  
Changes 841  
Changing 841  
Charts 128  
Check Box 292  
Checkbox 292  
CheckBox 428  
Child 13  
Children 13  
Choose Groups 313  
Choose Icons 58  
Choose Users 311  
ClientSection 428

Collaborative Document Authoring 237  
Collaborative Document Markup 227  
Collaborative Editing 237  
Color 280  
Color Picker 280  
Color Properties 280  
Color Settings 280  
Comment End 370  
Comment Log 306  
Comment Start 370  
CommentEnd 370  
CommentLog 306, 428  
Comments 227, 237, 303, 320, 370  
CommentStart 370  
Condition Builder 62  
Conditions 62  
Configure Activity 572, 582, 596, 600-604, 612  
Configure Permissions 829-830, 834, 841  
Configure Report 625, 636  
Configure Timeline 560, 567, 572, 582-583, 596-598, 600-604, 608, 612  
Configuring Forms 210, 213, 217-218, 238-239, 247-248, 257, 260, 264, 266, 269, 271  
Connection 174, 178, 182, 186, 191, 195, 203, 871, 880  
Connector 174, 178, 182, 186, 191, 195, 203, 871, 880  
Container 365  
Content List 21, 37-38, 81, 723, 726, 734, 741, 748  
Content List Objects 21, 37-38, 81, 741  
Content List Organization 748  
Content List Structure 723  
Content Locking 733  
Content Object 726  
Content Picker 315  
ContentPicker 315, 429  
Control Colors 280  
Control Icons 281  
Control Labels 339

Control Picker 318  
ControlPicker 318, 430  
Controls 274, 280, 303, 322, 376, 386, 414, 455  
Create Report 625, 636  
Create Template 767  
Create Timeline 560, 567, 572, 582-583, 596-598, 600-604, 608  
Creating Forms 210, 213, 217-218, 238-239, 247-248, 257, 260, 264, 266, 269, 271  
Creating Objects 726  
Creating Tables 455  
CSV 522  
Custom Task 600  
Custom Task Activity 600  
Custom Task Tab 600  
Custom Utilities 796

## D

Dashboards 147, 156  
Data 174, 178, 182, 186, 191, 195, 203, 220, 317, 843, 871, 880  
Data Flow Analyzer 691, 843  
Data import 775  
Data List 401  
Data List Controls 401  
Data Source 174, 178, 182, 186, 191, 195, 203, 871, 880  
Database 174, 178, 182, 186, 191, 195, 203, 269, 775, 871, 880  
Datalist 401  
Datasource Picker 317  
DataSourcePicker 317  
Datasources 174, 176, 178, 182, 186, 191, 195, 203, 317, 871, 880  
Date 293, 430  
Date Control 353  
Date Difference 353  
Date Field 293  
Date Math 353  
Date Picker 293  
DateDiff 353, 432

DatePicker 293  
DateTime 431  
DB 174, 178, 182, 186, 191, 195, 203, 871, 880  
DBConnectorPicker 433  
Debug 74  
Debug Mode 74  
Default Form 601  
Delete 829-830, 834, 841  
Delete Children 829-830, 834, 841  
Designing Tables 455  
Desktop 76  
Desktop Interface 76  
Desktop UI 76  
Desktop Workspace 76  
DFA 843  
Different Forms 601  
Disability 699, 701, 710, 714, 720, 723  
Disable fields 223  
Document Attachments 386  
Document editing 237  
Documentation 8  
Documents 227, 237, 386  
DropDown 204, 294, 433  
Dropdown Control 294  
Dropdown Items 204  
Dropdown Object 204  
Dropdowns 204

## E

ECN 301 549 699, 701, 710, 714, 720, 723  
Edit Attachments 237  
Edit Instance 612  
Edit Tab 247  
Editing 237  
Email Complete Link 334, 468

Email Complete URL 468  
Email Data 359, 458, 466  
Email Import 806  
Email Messages 806  
Email Notifications 359, 596  
Email Offline Completion 468  
Email Result Link 468  
Email Results 458, 466, 468  
Email Task Completion 468  
Email Templates 359, 458, 466, 468  
EMAIL\_COMPLETE\_URL 468  
EMAIL\_RESULT\_LINKS 468  
EmailCompleteLink 334  
EmailData 359, 458, 460, 466  
Emails 359, 458, 466, 468  
Embedded Section 366  
Embedded Section End 366  
EmbeddedSection 366  
EmbeddedSectionEnd 366  
Enable Fields 223  
Enabling 223  
End Process 602  
End Process Activity 602  
Equations 352  
Evaluate 62  
Event Field 218  
Events 218  
Example 553, 620  
Examples 16, 553, 620  
Excel 522, 524  
Existing Processes 691  
Export application 748  
Export objects 748  
Exporting 522, 748  
External Data 89-90, 103, 174, 178, 182, 186, 191, 195, 203, 642, 775, 871, 880

External Users 823

## F

Facebook 203

File Import 797

File Path 191

Files 191

Fill Dropdowns Tab 266

Find Icons 58

Folder Structure 734

Form 417

Form Actions 601

Form Actions Activity 601

Form Auditing 356

Form Cache 220

Form Calculations 352

Form Colors 280

Form Comments 320, 370

Form Configuration 210, 213, 217-218, 238-239, 247-248, 257, 260, 264, 266, 269, 271

Form Controls 274, 280, 289, 303, 315, 322, 338, 364, 376, 386, 401, 414, 455

Form Controls Tab 248

Form Creation 210, 213, 217-218, 238-239, 247-248, 257, 260, 264, 266, 269, 271

Form Data 220

Form Data Caching 220

Form Data SQL View Tab 269

Form Definition 209-210, 213, 217-218, 223, 238-239, 247-248, 257, 260, 264, 266, 269, 271

Form Error String 370

Form Error String Location 370

Form Errors 370

Form Events 218

Form HTML 344

Form Icons 281

Form images 350

Form Info String 370

Form Info String Location 370

Form Information 370  
Form Instance 741  
Form Javascript 344  
Form Knowledge View 347  
Form KView 347  
Form Labels 339  
Form Locking 217, 354  
Form Math 352  
Form Pictures 350  
Form Properties 210, 213, 217-218, 238-239, 247-248, 257, 260, 264, 266, 269, 271  
Form Reports 357  
Form Samples 16  
Form Tabs 367  
Form URL 213  
Form Views 269  
FormErrorString 370  
FormErrorStrings 434  
FormInfoString 370  
FormInfoStrings 434  
Forms 209-210, 213, 217-218, 220, 227, 237-239, 247-248, 257, 260, 264, 266, 269, 271, 280, 289,  
303, 315, 338, 364, 386, 401  
Formulas 352  
Free HTML 344  
Full-text search 66  
full text 66

## G

Goal Object 474, 476  
Goals 474, 476  
Graphs 128  
Group Name 400  
Group Permissions 829-830, 834, 841  
Group Picker 313  
GroupPicker 313, 434  
Groups 400  
Guidelines 691



## H

Handicapped 699, 701, 710, 714, 720, 723  
Hide Fields 223  
Hierarchy 13  
Hold Timeline 608  
Horizontal Tabs 367  
Hotlink 349  
HotLink 435  
HTML Code 344  
HTML Comments 370  
HTML Control 344  
HTML Snippet 344  
Hyperlink 349

## I

Icon 58, 362, 435  
Icon Chooser 58  
Icon Properties 281  
Icon Settings 281  
Icons 58, 281  
Image 436  
Image Control 350  
Images 350  
Implementation 12, 691  
Implementations 16  
Import Application 748  
Import Data 642  
Import documents 797  
Import Emails 806  
Import Mail 806  
Import objects 748  
Import Objects 797  
Import Utility 797  
Importing 748  
Importing Data 775

Include 437  
Infographics 128  
Input 289, 437  
Input Control 289  
Input Controls 303  
Instance Properties 612, 741  
Instance Status 741  
Instances 741  
Instructions 553, 620  
Introduction 12  
Invite 330  
iPad 849  
iPhone 849  
Iterating Loop 604

## J

JavaScript 344  
Journal 320  
Journal Control 320  
Jump 603

## K

Knowledge View Control 347  
Knowledge View Definition 482, 488, 508, 514, 522  
Knowledge View Filter 508  
Knowledge View Parameters 508, 514  
Knowledge View URL 508  
Knowledge Views 482, 488, 508, 514, 522  
KView 347, 438  
KView Control 347  
KView Definition 482, 488, 508, 514, 522  
KView Export 522  
KView Filter 508  
KView Parameters 508, 514  
KView URL 508  
KViews 482, 488, 508, 514, 522

## L

Label 339  
Label Control 339  
Layout 364, 372  
Layout Controls 364  
List Box 307  
ListBox 307, 439  
Lock Form 217  
Lock Form Control 354  
LockForm 354  
Locking 733  
Loop 604  
Loop Conditions 604  
Loop Results 604  
Loop Settings 604  
Looping 604  
Looping Conditions 604

## M

Machine Learning 528  
Make a Timeline Wait 608  
Make Activity Wait 608  
Make editable 223  
Make Report 625, 636  
Make Templates 767  
Manage Instance 612  
Manage Process Users 328  
ManageUsers 440  
Managing Content 691  
Managing Objects 723  
Managing Users 691, 823  
Meta Data 316-317, 545, 549, 554  
Meta Data Admin 545, 549, 554  
Meta Data Attributes 545, 549, 554  
Meta Data Categories 545, 549, 554

metadata 550  
Metadata 316-317, 545, 549, 554  
Metadata Attribute 545, 549, 554  
Metadata Category 316, 545, 549, 554  
Microsoft Mail 806  
Microsoft SharePoint 778  
Microsoft Word 456  
Milestones 320  
ML 528  
Mobile 849  
Mobile App 849  
Mobile Application 691, 849  
Mobile Device 849  
Mobile Forms 849  
Mobile Users 849  
Modify 829-830, 834, 841  
Modify Children 829-830, 834, 841  
Move Row Down 379  
Move Row Up 379  
Multi-line 289  
Multiple Forms 601

## N

Navigation 21, 37-38  
Navigation Bar 21, 37-38  
New Template 767  
Non-Authenticated Users 823  
Notifications 359, 458, 466, 468, 596  
Notifications Tab 596  
Notify Activity 596

## O

OAuth 203  
OAuth Mail 806  
Object Instance 741  
Object Locking 733

Object Organization 723  
Object Permissions 829-830, 834, 841  
Object URL 776  
Objects 741  
Online Form Designer 209, 217, 227, 237, 274, 280, 289, 303, 315, 322, 338, 364, 376, 386, 401, 414, 455  
Online Review Tool 227, 237  
OnlyOffice 237  
Operator 64  
Operators 66  
Organization 734  
Organizational Data 775  
Organizing 734  
Organizing Objects 723  
Other Controls 338  
Other Input Controls 315

**P**

Parameter 89-90, 103  
Parameter Strings 514  
Parameters 89-90, 103  
Parent 13  
Parent Activity 604  
Parent Results 604  
Parents 13  
Participants Tab 583  
Pause Process 608  
Pause Timeline 608  
Permission Exceptions 841  
Permission Groups 829-830, 834, 841  
Permissions 691, 829-830, 834, 841  
Permissions Methodology 829-830, 834, 841  
Permissive Access 834  
Phone 849  
Pick Groups 313  
Pick Icons 58

Pick Users 311  
Picture Control 350  
Pictures 350  
Pre-built Apps 767  
Pre-Made Apps 767  
Prevent Form Editing 354  
Print 441  
Print Form 325  
Process Activity 597  
Process Changes 691  
Process Director Objects 81  
Process In Error 612  
Process Instance 612, 741  
Process Status 741  
Process Tab 597  
Process Timeline Activities 572, 582, 596, 600-604, 612  
Process Timelines 560, 567, 572, 582-583, 596-598, 600-604, 608, 612  
Process Wait 608  
Processes 510, 560, 567, 572, 582-583, 596-598, 600-604, 608, 612, 649, 652, 664, 680, 778, 841  
Properties Tab 239

## Q

Queries 843  
Query Strings 514  
Quick Apps 767

## R

Radio 441  
Radio Button 298  
Radio Button List 308  
Radio Buttons 308  
RadioButton 298  
RadioButtonList 308  
RadioList 442  
Rating 311  
Re-Authenticate 334

Reauth 334, 442  
Reauthenticate 334  
Remove Row 378  
RemoveRow 442  
Report Configuration 625, 636  
Report Designer 625, 636  
Report Forms 357  
Report Object 625, 636  
Report Writer 625, 636  
Reporting Component 625, 636  
Reports 357, 625, 636  
Responsive 372, 699, 701, 710, 714, 720, 723  
Responsive Layout Controls 372  
Responsiveness 372, 699, 701, 710, 714, 720, 723  
Restart Loop 604  
Results 843  
Results Tab 583  
RichText 443  
Rollback 603  
Routing Slip 341  
RoutingSlip 341, 443  
Row Number 383  
ROW\_NUM 383  
Rule 82  
Rules 82  
Running Timeline 612

## S

Sample 553, 620  
Sample Applications 16, 767  
Sample Apps 767  
Samples 16, 553, 620  
Save 446  
Save Button 325  
Schedule Knowledge View 474, 476

Schedule KView 474, 476  
Schedule Process 474, 476  
Schedule Timeline 474, 476  
Scheduled Actions 642  
Scheduled Import 642, 797, 806  
Scheduled Tasks 814  
Scheduler 355, 474, 476  
Schema 545, 549, 554  
Script Activity 598  
Script Tab 598  
Scripting 598  
Scripts 598  
Section 447, 466  
Section 508 699, 701, 710, 714, 720, 723  
Select Groups 313  
Select Users 311  
Send Emails 596  
Set Form Data 567  
Set Form Data Tab 264  
Set Icons 58  
Set Permissions 829-830, 834, 841  
SharePoint 195, 778, 871, 880  
SharePoint Export 778  
SharePoint Import 778  
Show Attach Kview 398  
Show Attached Objects 391  
Show Attached Objects with KView 398  
Show Fields 223  
Show Kview 347  
Show Report 357  
ShowAttach 391, 448  
ShowAttachKView 398, 450  
ShowReport 357  
Signature 304  
Signature Comments 303



Signature Control 304  
Signature Devices 319  
SignatureComments 303, 451  
Signatures 304, 319  
Sim 553, 620  
Simulation 553, 620  
Simulator 553, 620  
Slider 309  
Social 203  
Social Media 203  
Software Simulation 553, 620  
Sort Array 380  
SortArray 380  
Specified Pause 608  
Stop Process 602  
Stop Timeline 602  
Stream Actions 642  
Structure 734  
Subprocesses 597  
Sum 379, 452  
Sum Control 379  
Switch 300, 452  
Switch Forms 601  
Synchronous Subprocess 597  
System Administration 776, 888  
System Variable Control 338  
Sysvar Control 338

## T

Tab Controls 367  
Tab Strip 367  
Tab Strip Content 367  
Tab Strip Content End 367  
Tab Strip End 367  
Tabbed Form 367

TabContent 453  
Table Properties 455  
Tables 455  
Tablet 849  
TabStrip 367, 452  
TabStripContent 367  
TabStripEnd 367  
TanStripContentEnd 367  
Task Comments 303  
Task Emails 359, 596  
Task Notification 458, 466, 468  
Task Notifications 359  
Taxonomy 545, 549, 554  
Template 767  
Template Apps 767  
Template Library 767  
Templates 767  
Text Box 289  
Textbox 289  
Time 453  
Timeline 560  
Timeline Activities 572, 582, 596, 600-604, 612  
Timeline Activity 572, 582, 596, 600-604, 612  
Timeline Configuration 560, 567, 572, 582-583, 596-598, 600-604, 608, 612  
Timeline Definition 560  
Timeline Instance 612, 741  
Timeline Options 560, 567, 583, 597-598, 608  
Timeline Properties 567  
Timeline Settings 567  
Timeline Status 741  
Timelines 841  
Tooltip 362  
Topaz 319  
Topaz Device 319  
Topaz Signature Control 319

Twitter 203  
Two Dates 353  
Two Forms 601

## U

UI 21, 37-38, 76  
URL 213, 776  
URL Access 776  
URL Filters 508  
URL Link 349  
URLs 213, 776  
User Activity 583  
User Interface 21, 37-38, 76  
User Permissions 829-830, 834, 841  
User Picker 311  
User Profile 823  
User Task 583  
UserPicker 311, 453  
Users 823, 830  
Utilities 797, 806, 814

## V

Validation Rules tab 260  
Variables 458  
Vertical Tabs 367  
View 829-830, 834, 841  
View Children 829-830, 834, 841  
Views 269  
Visibility 223, 466

## W

Wait Activity 608  
Wait Until Time 608  
WCAG 699, 701, 710, 714, 720, 723  
Widgets 147, 156  
Windows Scheduler 814

Word Builder 456  
Word Designer 456  
Word Form Builder 456  
Word Forms 456  
Workflow Steps 664  
Workflows 649, 652, 664, 680, 841  
Workspace 76, 147, 156  
Workspace Tabs 21, 37-38