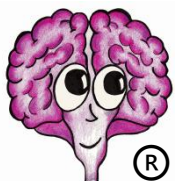


# DocuBrain® TechDoc Workflow Guide



A DocuBrain® Product

<https://docubrain.com/>

By Prevo Technologies, Inc.

<https://prevo.com/>



## **DocuBrain® TechDoc Workflow Guide**

By Prevo Technologies, Inc.

Copyright © 2020, Prevo Technologies, Inc. All rights reserved.

Published by Prevo Technologies, Inc., 1111 Keener Rd, Seymour, TN, 37865.

This guide is distributed with software that includes an end user license agreement (EULA). This guide, as well as the software described in the EULA, is furnished under license and may be used or copied only in accordance with the terms of the EULA. Except as permitted by the EULA, no part of this guide may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of Prevo Technologies, Inc. Please note that the content in this guide is protected under copyright law even if it is not distributed with software that includes a EULA.

The authoritative end user license agreement (EULA) can be found at:

<https://docubrain.com/licenses>

The content of this guide is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Prevo Technologies, Inc (PTI). PTI accepts no responsibility or liability for loss or damage occasioned to any person or property through use of the material, instructions, methods, or ideas contained herein, or acting or refraining from acting as a result of such use. PTI disclaims all implied warranties, including merchantability or fitness for any particular purpose.

DocuBrain and the DocuBrain logo are registered trademarks of Prevo Technologies, Inc.

## Table of Contents

1. Introduction .....	1
2. Getting Started.....	2
3. Overview of Features.....	3
3.1. DocuBrain Workflow Editor .....	3
3.2. Workflow Engine.....	4
3.3. Electronic Notifications.....	5
3.4. User Task Management .....	5
3.5. Monitoring and Administration .....	5
4. DocuBrain Workflow Editor Tutorial.....	7
4.1. Layout of the User Interface .....	7
4.1.1. Toolbar .....	7
4.1.2. Left Panel .....	9
4.1.3. Right Panel .....	10
4.1.4. Status Bar .....	12
4.2. Creating a Workflow Process.....	12
4.3. Connecting to a Workflow Engine Repository .....	16
4.4. Deploying to a Workflow Engine Repository .....	18
4.5. Downloading from a Workflow Engine Repository .....	20
4.6. Multi-Process Deployments.....	24
4.7. Importing and Exporting Diagrams.....	24
4.8. Printing a Diagram .....	25
4.9. Working with Fragments.....	26
4.10. Calling TechDoc/BPMN Service Task Operations .....	27
5. TechDoc Service Task Operations.....	30
5.1. Add Access Association .....	30
5.2. Add Commenter Association .....	32
5.3. Add Distribution Association .....	33
5.4. Add Keyword.....	34
5.5. Add Notification Association .....	35
5.6. Release Document .....	36
5.7. Remove Access Association .....	37
5.8. Remove Commenter Association .....	38
5.9. Remove Distribution Association.....	40
5.10. Remove Keyword .....	41
5.11. Remove Notification Association.....	41
5.12. Replace Keyword .....	43
5.13. Reserve Document.....	43
5.14. Simple Web Request.....	44
5.15. Select Document.....	44
5.16. Send Email.....	45

- 5.17. Unrelease Document ..... 46
- 5.18. Unreserve Document..... 46
- 6. Script Task Helper Objects ..... 47
  - 6.1. DBBase64 ..... 47
  - 6.2. DBHtmlParser..... 47
  - 6.3. DBJsonBuilder ..... 48
  - 6.4. DBJsonParser..... 49
  - 6.5. DBNow ..... 50
  - 6.6. DBSoapUtils..... 50
  - 6.7. DBXmlBuilder ..... 51
  - 6.8. DBXmlParser ..... 52
    - 6.8.1. containsAttribute ..... 52
    - 6.8.2. elementToString ..... 52
    - 6.8.3. findElementById ..... 52
    - 6.8.4. findElementByName ..... 53
    - 6.8.5. findElementsByAttributeValue ..... 53
    - 6.8.6. getRoot..... 53
    - 6.8.7. hideNodes ..... 53
    - 6.8.8. parseAttributeValueAsBoolean ..... 54
    - 6.8.9. parseAttributeValueAsDouble ..... 54
    - 6.8.10. parseAttributeValueAsInteger ..... 54
    - 6.8.11. parseAttributeValueAsString ..... 54
    - 6.8.12. parseAttributesByPrefix..... 55
    - 6.8.13. parseChildNode..... 55
    - 6.8.14. parseChildNodes ..... 55
    - 6.8.15. parseTextContent ..... 55
    - 6.8.16. stripNamespace ..... 56
- 7. The TechDoc Server-Side Workflow Engine Reference ..... 57
  - 7.1. Workflows Management Menu..... 57
    - 7.1.1. Creating a Workflow Deployment ..... 57
    - 7.1.2. Deleting a Workflow Deployment ..... 58
    - 7.1.3. Modifying a Workflow Deployment ..... 59
    - 7.1.4. Showing a Workflow Deployment ..... 60
    - 7.1.5. Showing a Workflow Process Definition..... 61
    - 7.1.6. Showing a Workflow Process Instance ..... 63
    - 7.1.7. Showing Workflow Activity..... 65
    - 7.1.8. Starting a Workflow Process Instance ..... 65
    - 7.1.9. Activating a Workflow Process Instance..... 66
    - 7.1.10. Suspending a Workflow Process Instance ..... 66
    - 7.1.11. Deleting a Workflow Process Instance ..... 67
    - 7.1.12. Modifying Workflow Process Instance Variables ..... 67
    - 7.1.13. Creating a Workflow Process Trigger ..... 69
    - 7.1.14. Modifying a Workflow Process Trigger..... 70
    - 7.1.15. Deleting a Workflow Process Trigger..... 72

7.1.16. Showing a Workflow Process Trigger .....	72
7.1.17. Showing a Workflow Task.....	74
7.1.18. Completing a Workflow Task .....	75
7.1.19. Claiming a Workflow Task.....	76
7.1.20. Assigning a Workflow Task .....	77
7.1.21. Unassigning a Workflow Task .....	77
8. Suggestions and Feedback .....	78



## 1. Introduction

Workflow is a term that is used to describe all of the steps, tasks, or events that occur in a business process and the participants or organizations involved. For example, think of an automobile manufacturer (the organization) and all of the steps involved in the manufacturing process of a car: receiving raw materials, casting and machining parts, assembly of the engine and transmission, construction of the chassis, etc. Though complex, every process can be broken down to individual steps.

In order to notate a process, each step must be identified along with the requirements and participants involved. Once a process has been identified, it can be modeled. The most widely used and accepted method of describing a workflow process is Business Process Model and Notation (commonly referred to as BPMN). Using BPMN (usually through a graphical design tool such as the DocuBrain® Workflow Editor), an individual can map out each step of a process from start to end. Once the process has been modeled using BPMN, it can be uploaded to a workflow engine and executed.

TechDoc features a vast set of capabilities, with one of the main features being a workflow engine. The workflow engine included in TechDoc is BPMN 2.0 compliant and has been tightly integrated with TechDoc to provide a seamless workflow experience. While you can design processes to be completely standalone and independent from TechDoc, you can also perform a large array of TechDoc operations through BPMN Service Tasks as described in the [TechDoc Activities](#) section below.

Before you dive in and start trying to build and design workflow processes, it is necessary to become familiar with the BPMN 2.0 specification. Throughout this guide, we will touch on small pieces of the specification and how they pertain to workflow in TechDoc however, we will not be going into depth reiterating the BPMN 2.0 specification.

## 2. Getting Started

This guide contains the information you need to understand to design, build, and manage workflow processes within DocuBrain products. Starting from the ground up, we will cover from the fundamentals of process design to deployment and execution on the TechDoc workflow engine.

- [Overview of Features](#) – Provides a high-level overview of the features and capabilities.
- [DocuBrain Workflow Editor Tutorial](#) – A tutorial detailing the use of the DocuBrain Workflow Editor.
- [TechDoc Service Task Operations](#) – Covers the TechDoc specific operations that are available for use via Service Tasks in a workflow process.
- [The TechDoc Workflow Engine](#) – Overview of the TechDoc workflow engine and how to interact with it.



### 3. Overview of Features

DocuBrain TechDoc features a complete workflow management system. From process design to execution and management, we have you covered. In today's every changing business world, business process automation is invaluable. Using DocuBrain TechDoc, you can leverage all the benefits of electronic document and records management while streamlining all of your business processes. Integration of electronic document management and business process automation has never been easier.

#### 3.1. *DocuBrain Workflow Editor*

The DocuBrain Workflow Editor is an easy to use and intuitive workflow process editor. It allows users the ability to graphically model business processes following the BPMN standard and deploy them directly to a TechDoc workflow engine. The DocuBrain Workflow Editor can be used in conjunction with TechDoc or used on its own to create generic BPMN 2.0 workflow processes. The DocuBrain Workflow Editor features:

- Drag and drop modeling
- BPMN 2.0 compliant
- Code free process design
- Customizable look & feel
- Connects to TechDoc Document Managers
- Multi-diagram deployments
- Collaborative process modeling
- Process repository
- Process versioning
- BPMN fragments/snippets
- Form editing
- Process deployment
- Model validation

- Diagram printing

### 3.2. *Workflow Engine*

The TechDoc workflow engine is an extremely fast and lightweight Business Process Management (BPM) Platform. The process engine is BPMN 2.0 compliant and fully integrated with TechDoc. Alongside all of the abilities of BPMN 2.0, an extensive set of TechDoc operations are also available for use via BPMN Service Tasks. In addition, workflow processes can be triggered to execute based on different events in a TechDoc Document Manager such as the creation, modification, or deletion of a document. Here are just a few features of the TechDoc workflow engine:

- Completely web-based, accessible from anywhere
- Process administration and management dashboard
- Real time monitoring with graphical view of workflow process and process variable inspector
- User task integration with TechDoc Users and Groups
- Email-based notification and alerts
- Direct launch of user task completion from email notification
- Process versioning
- Centralized exception handling
- Inter-process communications
- Parallel activities (forking and joining)
- Long running workflows (months or more)
- Sub-process and task looping
- Expression evaluation using Unified Expression Language (UEL)
- Escalation
- Forms support
- History logging

- Reporting

### ***3.3. Electronic Notifications***

The workflow engine is fully integrated with TechDoc's email and notification system. The engine sends email notification to users when: a process requires them to complete a task, a process of theirs encounters a business exception or escalates, etc. Additionally, the email notification sent contains action links to the TechDoc Document Manager that will take the user straight to the item needing their attention. This means you can stay connected even while on the go with your smart phone or tablet.

### ***3.4. User Task Management***

User Tasks generated by workflow processes are completely integrated with TechDoc. User Tasks appear under the TechDoc Document Manager My Work area just like all other TechDoc items that require a user's attention. Additionally, as mentioned in the previous section, User Tasks also notify the user via email when a task needs to be completed.

What happens when a User Task is encountered within a process, and the user assigned is gone on vacation? No worries here, you can easily locate and re-assign a User Task under the Workflows Management section in the TechDoc Document Manager. Furthermore, User Tasks can also be un-assigned so they are made available to any other potential owners, or can just be completed directly without the need to be re-assigned. Flexibility is everything with User Task management.

### ***3.5. Monitoring and Administration***

Monitoring the TechDoc workflow engine could not be easier. Under the Workflows Management section of a TechDoc Document Manager, a user with the Workflows Management privilege can easily view all of the deployments, process definitions, and process instances in the system as well as all User Tasks awaiting completion.

Have you ever wondered exactly what step your process instance is currently on? Many systems simply tell you the name or ID of the element being executed and then require you to pull up the diagram and try to locate the element yourself. In TechDoc, you can simply click a running process instance. TechDoc features a real-time interactive process diagram so you can see the process just as it was designed as well as the activity being executed. However, we did not stop there. You can also click elements on the process and view all of their metadata in a popup window. Have you ever forgotten how long of a period you set on a timer task or the assignee of a User Task? Just click the activity and

take a quick peek; no more firing up an editor and loading a diagram just to figure it out. Additionally, when a process instance is focused on a User Task, you can just click the User Task and you will be taken straight to that task to complete.

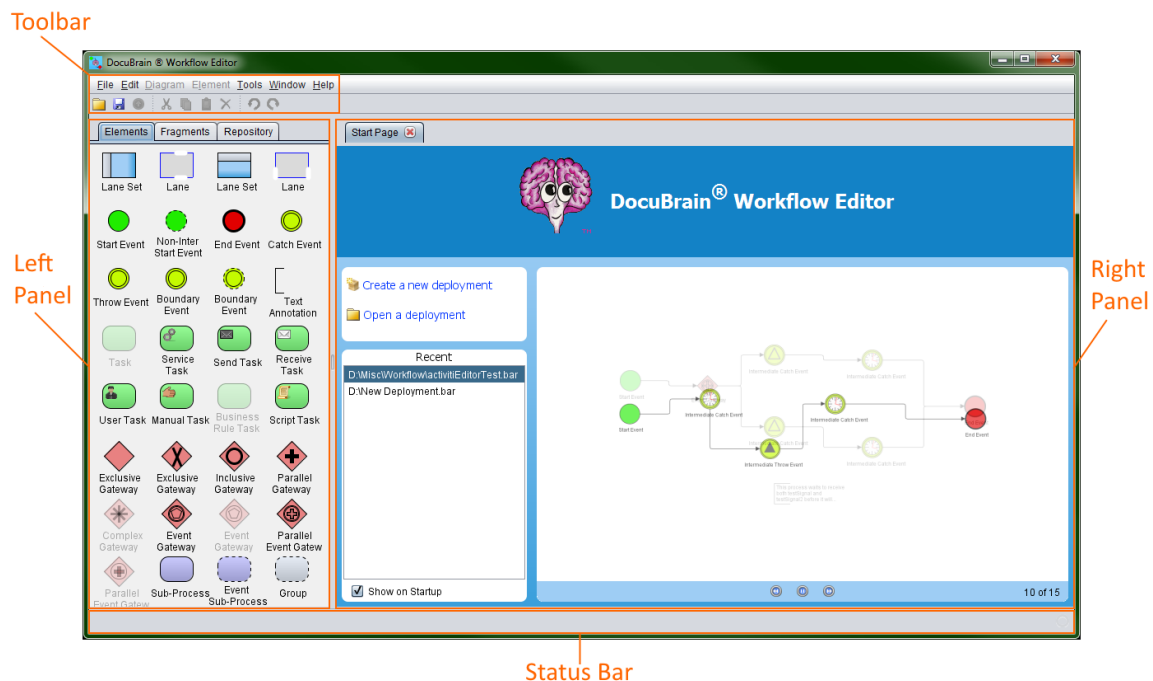
All of this is great for administrators, but what about normal users? All users can make use of the real-time monitoring and interaction with the only restriction being they can only view processes and assign/complete tasks they own.

## 4. DocuBrain Workflow Editor Tutorial

In this section, we will go over all the basics of using the DocuBrain Workflow Editor. We will not touch much on the BPMN specification, but there will be portions that go over interaction with the BPMN elements. As the BPMN elements do differ a great deal, it is good idea to at least be familiar with the basic elements: Start/End Events, Intermediate Events, Boundary Events, Tasks, and Gateways.

### 4.1. Layout of the User Interface

The editor is split into four main areas: the [Toolbar](#), [Left Panel](#), [Right Panel](#), and [Status Bar](#).












#### 4.1.1. Toolbar

On the toolbar, you will find all of the menus for the editor and the short cut buttons. The toolbar contains the following menus:

- File – Used to open, close, and save workflow deployments as well as add, remove, import or export diagrams from an open deployment. Additionally, this menu can be used to print an open deployment or upload it to a workflow repository.
- Edit – This menu is only used when a deployment is open. It contains the undo/redo options to undo or redo an action to a diagram such as un-delete an element. This menu also contains all of the clipboard operations: to cut, copy, paste and delete elements from or to a diagram.
- Diagram – This menu is only used when a deployment is open. It contains all of the properties for the currently selected diagram. This menu is also accessible by simply right clicking on the diagram (grid paper). On this menu, you will find options like setting the file name of the diagram, the BPMN process ID and name, editing the XML namespaces for the diagram, etc.
- Element – This menu is only used when a deployment is open and a BPMN element is selected on a diagram. It contains all of the properties for the currently selected BPMN element. This menu is also accessible by simply right clicking on an element on your diagram. On this menu you will find the properties for the element, the ability to cut or copy the element (or selection if more than one element is selected), and other options that vary from element to element depending on its type.
- Tools – The Tools menu contains all of the tools and global settings for the editor. This is where you will find the Options dialog where you can change the target workflow engine and edit presets. You can also stylize the Diagram Editor and XML Editor to suit your needs. Additionally, the XML Editor's theme can be changed if needed. Note that there is a high contrast theme for those who need it.
- Window – The Window menu contains all of the options to display the various windows of the editor if they have been closed such as the Start Page.
- Help – The Help menu contains the About dialog that shows the editor's version and your license info, as well as the options to check for updates to the editor and submit feedback on the editor to the DocuBrain website.

The icons of the toolbar from left to right are shortcuts to toolbar menu items. There are as follows:

-  File-> Open Deployment
-  File-> Save Deployment

-  File -> Upload Deployment
-  Edit -> Cut
-  Edit -> Copy
-  Edit -> Paste
-  Edit -> Delete
-  Edit -> Undo
-  Edit -> Redo




#### 4.1.2. Left Panel

The left panel contains four tabs: Deployment, Elements, Fragments, and Repository tab.

The Deployment tab displays the deployment that is currently open in the editor. Each process contained in the deployment is listed here and may be opened by double clicking it. Additionally, you can right click a process and delete it from the deployment or right click the deployment and add a new process.

On the Elements tab, you will find all of the BPMN elements that the editor currently supports. To drag an element onto a diagram, press and hold the left mouse button over an element, drag it over to the diagram and release the left mouse button. This will place the element onto the diagram; you can then proceed to make connections, size and position the element, etc.

On the Fragments tab, you will find all of the fragments (or snippets as they are sometimes referred to) that you have created. Fragments are not usually a full diagram, but just a few elements connected in a particular way that you find yourself re-creating repeatedly. We will dig deeper into fragments later on.

The Repository tab is used to connect to a TechDoc workflow engine. Once a connection has been configured, you can select it from the dropdown box and click the connect button  to connect to the workflow engine and retrieve the list of all the workflow deployments and processes it contains. Once the list of deployments has been retrieved, you can select a deployment and click the download button  to download the deployment from the workflow engine (repository) and load it in the editor to view or edit. A deployment can be deleted from the repository by selecting it from the list and clicking the delete button . When a deployment is selected from the list, you can

view the Definitions, Resources, and Images it contains using the tabs below the deployment list. We will go into more detail later on about repository connections.

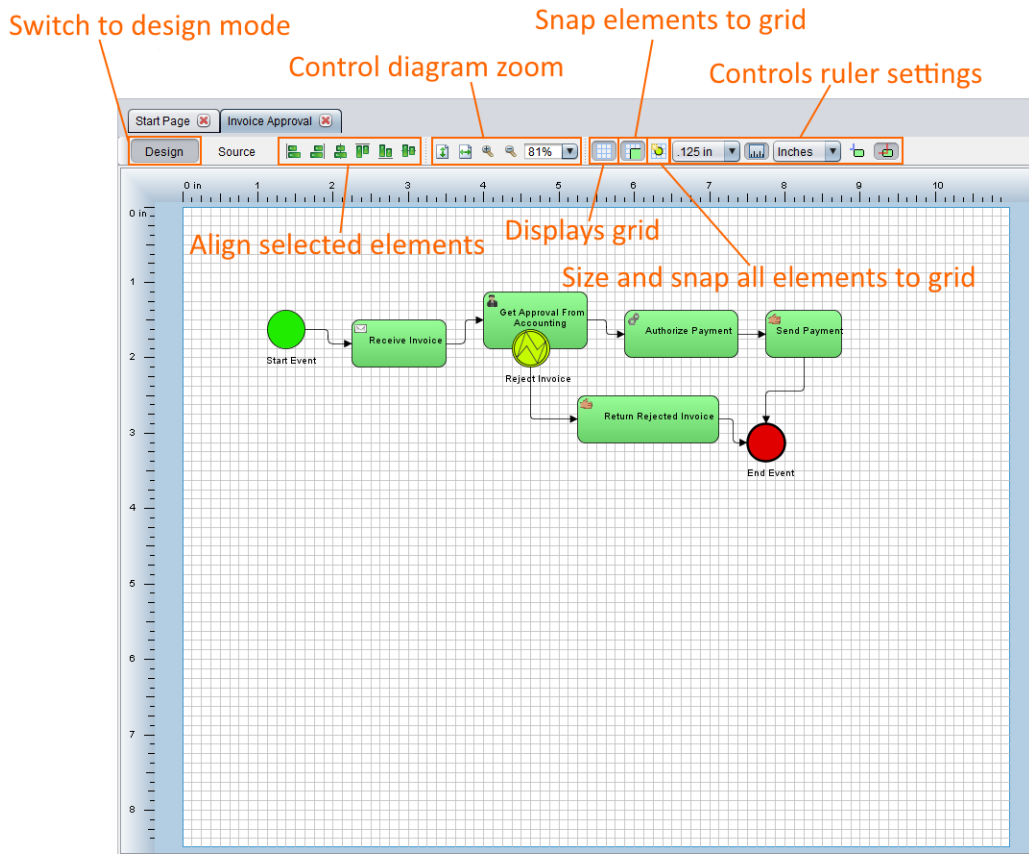
### 4.1.3. **Right Panel**

The right panel is the workspace of the editor. When you create or load a deployment, all of the diagrams are loaded into the workspace. Each diagram will have a tab, and on each tab, there are two main sections the Design view and Source view.

The design view is used to graphically layout a workflow process. Elements can be dragged from the elements panel on the left panel to the diagram. Once an element has been placed on the diagram, it becomes part of the workflow process. To view the properties of the diagram, right click anywhere on the graph paper. To view the properties of an element, right click on an element.

The design view features a both a ruler and tool bar. The ruler can be set to use inches, metric, or pixel-based measurements. The graph paper's grid size synchronizes to the ruler's settings and is very helpful when placing and aligning elements. The toolbar contains various buttons and settings, see the image below for a description of each:





The source view is used to examine the XML BPMN code generated by building the diagram in design mode. The source view is used purely for reference and cannot be used to alter or manually code BPMN. This view also contains a toolbar; see the image below for a description of each control:

Switch to source mode

Highlight matching tags

Enable code folding and line number

Collapse tag

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <definitions exporter="DocuBrain Workflow Editor" exporterVersion="8"
3   expressionLanguage="http://www.w3.org/1999/XPath"
4   targetNamespace="http://docubrain.com/namespaces/bpmn"
5   typeLanguage="http://www.w3.org/2001/XMLSchema"
6   xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
7   xmlns:activiti="http://activiti.org/bpmn"
8   xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
9   xmlns:dc="http://www.omg.org/spec/DD/20100524/DC"
10  xmlns:di="http://www.omg.org/spec/DD/20100524/DI"
11  xmlns:docubrain="http://docubrain.com/namespaces/bpmn"
12  xmlns:tns="http://sourceforge.net/bpmn/definitions/_1377117918629"
13  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
14  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:yaoliang="http://bpmn.sourceforge.net">
15  <error errorCode="123" id="ERR_1" name="ERR_TEST"/>
16  <message id="MSG_1" name="MSG_TST"/>
17  <process id="InvoiceApproval" isExecutable="true"
18    name="Invoice Approval" processType="None">
19    <manualTask completionQuantity="1" id="_18"
20      isForCompensations="false" name="Return Rejected Invoice" startQuantity="1">
21      <incoming_19</incoming>
22      <outgoing_20</outgoing>
23    </manualTask>
24    <manualTask completionQuantity="1" id="_15"
25      isForCompensations="false" name="Send Payment" startQuantity="1">
26      <incoming_16</incoming>
27      <outgoing_17</outgoing>
28    </manualTask>
29    <userTask activiti:priority="50" completionQuantity="1" id="_6"
30      implementation="##Unspecified" isForCompensations="false"
31      name="Get Approval From Accounting" startQuantity="1">
32      <incoming_11</incoming>
33      <outgoing_13</outgoing>
34      <humanPerformer id="_6_RES_1" name="">
35        <resourceAssignmentExpression>
36          <formalExpression><![CDATA[jeprevola]]</formalExpression>
37        </resourceAssignmentExpression>
38      </humanPerformer>
39    </userTask>
40    <receiveTask completionQuantity="1" id="_8"
41      implementation="##WebService" instantiate="false"
42      isForCompensations="false" messageRef="MSG_1"
43      name="Receive Invoice" startQuantity="1">
44      <incoming_10</incoming>
45      <outgoing_11</outgoing>

```

#### 4.1.4. Status Bar

The status bar is the bottom bar of the editor. It is used to display status messages when the editor is busy doing things such as saving or loading a deployment, retrieving a list of deployments from a repository, etc. The status bar can be very useful to watch the progress of action that can take just a bit to complete. For example, if you are connecting to a workflow repository over a slower internet connection that contains many deployments, it could take several seconds to a minute.

## 4.2. Creating a Workflow Process

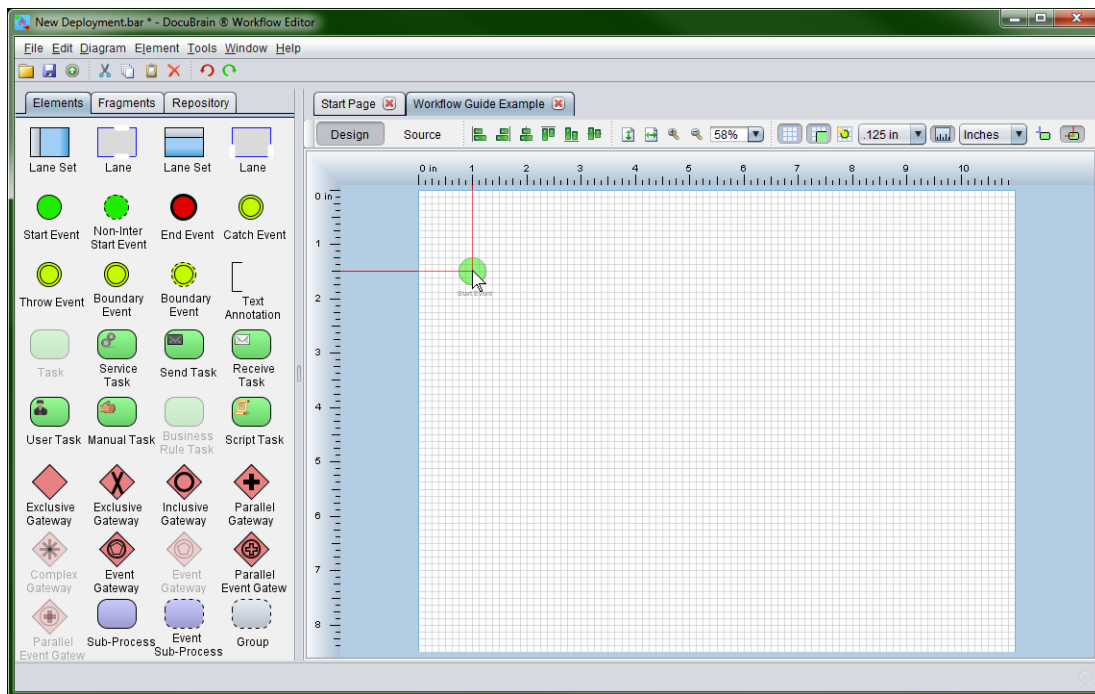
In this section, we will cover what you need to know to construct a basic workflow process. Start by launching the editor and selecting "New Deployment" from the File menu, or by clicking "Create a new deployment" on the Start Page. You will then be presented with a new deployment containing just one diagram.

When creating a new process, you must always start by giving the diagram a file name, and then name the process. Right click on the diagram (the graph paper), and then click

Diagram Properties under Diagram. On the Diagram Properties dialog, enter the filename to use for the diagram and then click OK.

Now right click on the diagram, and then click Process Properties under Diagram. On the Process Properties dialog, you will want to give the process its name and ID; this is a very important step. Later on, when you deploy this process to a workflow engine, it will be very hard to identify your process if it does not have a unique name. Additionally, if you upload a process with a name or ID that is already being used by another process, many workflow engines will assume this is a new version of the original process and may begin using it for unintended purposes. After you have given the process a name and ID, click the OK button.

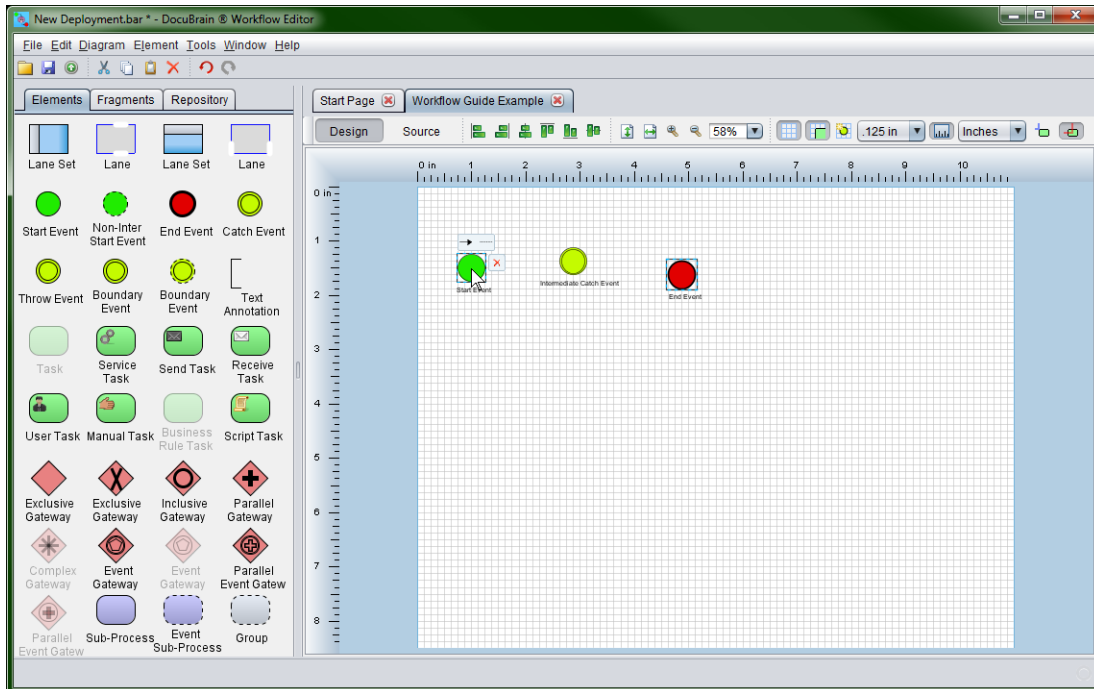
Now we have a named diagram file and process. Let us begin by dragging a Start Event from the elements panel on the Left Panel onto the diagram.



Notice when we drag the element around on the diagram, there is a red alignment line extending from the center of the element towards the ruler. These alignment lines can be used to make sure we place the elements exactly where we want it. If you prefer, you can click the corner align button (blue alignment line) from the upper toolbar so that you may align the element using its upper left corner.

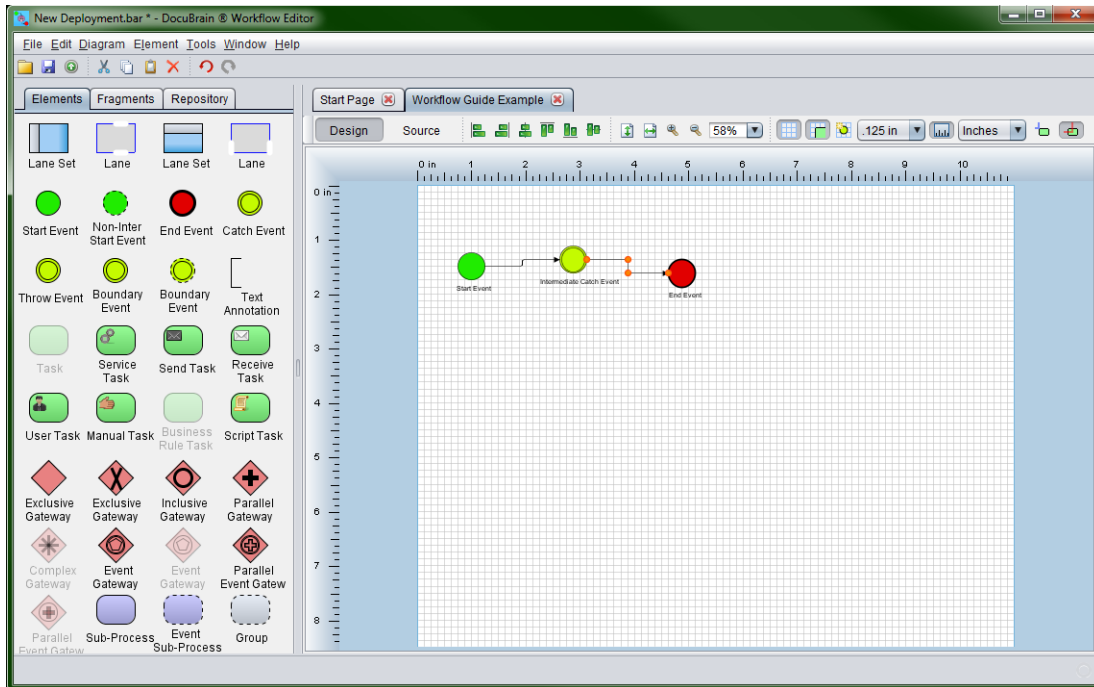
Now that we have placed a Start Event, go ahead and drag a Catch Event and End Event onto the diagram as well. In order for a workflow engine to know how to execute your process, the elements need to be connected in the order in which they should execute

beginning with the Start Event. Let us begin by connecting the Start Event to the Catch Event. To do this, click on the center Start Event to show its actions:




Once the actions appear, click and hold the left mouse button over the connection arrow. Now drag the red connection line over the center of the Catch Event and release the left mouse button to make the connection. This will make a Sequence Flow connection from the Start Event to the Catch Event. The order in which you make connections is important. Notice how the arrow on the line points to the Catch Event. This means the connection is sourcing from the Start Event and targeting the Catch Event. If we had done this backwards, the connection would be starting from the Catch Event and ending on the Start Event, which is not the direction that we intend for the engine to execute our process; in fact, this is also illegal, as a Start Event cannot have an incoming connection by the BPMN 2.0 specification.


Now that you have connected the Start Event and Catch Event, go ahead and connect the Catch Event to the End Event. Your diagram should now look like this:



Notice the orange dots along the connection. If desired, you could left click and hold on top of an orange dot, drag it to a new location and release the mouse button to change the connection's path. Additionally, if you drag the first or last dot you can change the point at which the connection leaves or enters the source or target element.

Now we have a complete BPMN process. By the BPMN specification, to have a complete process, you must have a Start Event and an End Event. While this process would execute on a workflow engine, it would be near impossible to catch while it is running because as soon as you start it, it would complete because there are no operations to perform. Let us address this by giving the Catch Event a timer definition. Right click on the Catch Event Task and then click Event Definitions.


On the Element Event Definitions dialog, click the create button  and then select Timer under element defined. Select Duration as the type and enter PT1M as the expression to say that we want a duration period of 1 minute. Click the OK button on the Create Timer Event Definition dialog. Now on the Element Event Definitions dialog, you can see the timer event definitions that you just created. Click the OK button to close the Element Event Definitions dialog.

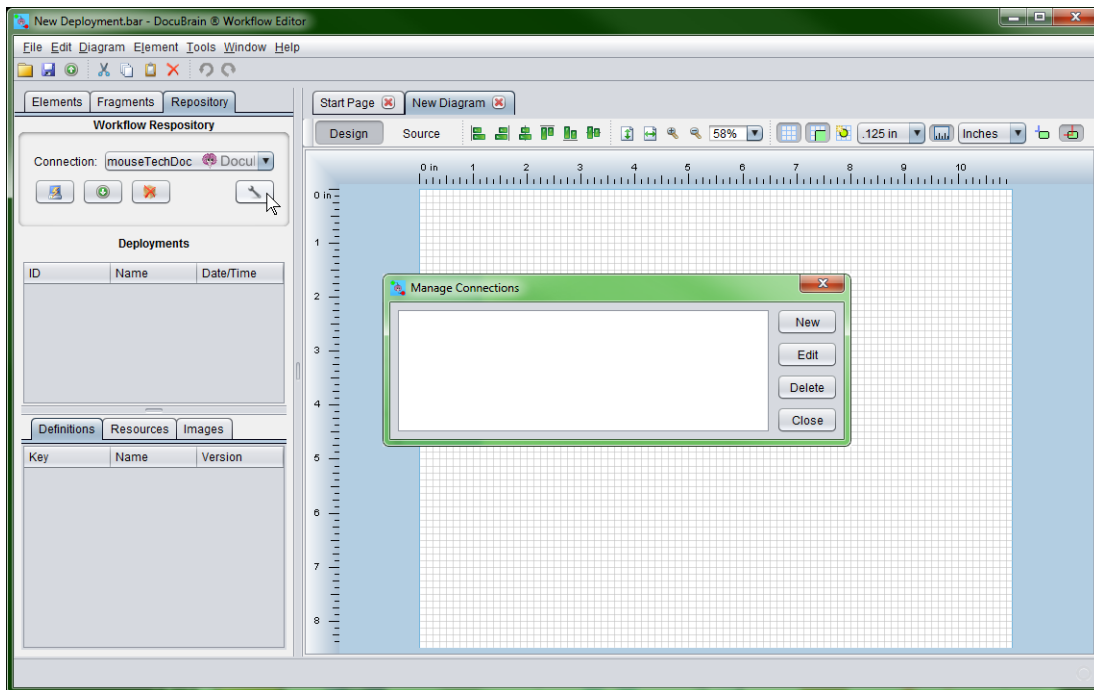
You now have a complete process that will execute and take one minute to complete. Go ahead and save this deployment; we will need it in our next exercise. On the File menu, click Save Deployment or click the save button  on the toolbar. Find a location on your computer to save the deployment, give it a name and click the Save button.

You can use this deployment in the next section when we connect to a workflow repository and upload it.

### 4.3. Connecting to a Workflow Engine Repository

The DocuBrain Workflow Editor has the ability to connect directly to a TechDoc workflow engine to deploy workflow processes or download and edit existing processes.

Let's start by clicking the Repository tab on the Left Panel and then clicking the manage connections button .



On the Manage Connections dialog, click the New button. You will now be looking at the Create Connection dialog.

- Enter a name for the connection in the Connection Name box.
- Select the TechDoc version this connection will target.
- Enter the host name of the TechDoc instance; i.e. example.com making sure to leave off any trailing context path.
- Enter the base URL for a TechDoc connection is always */service/workflow/* and cannot be changed.

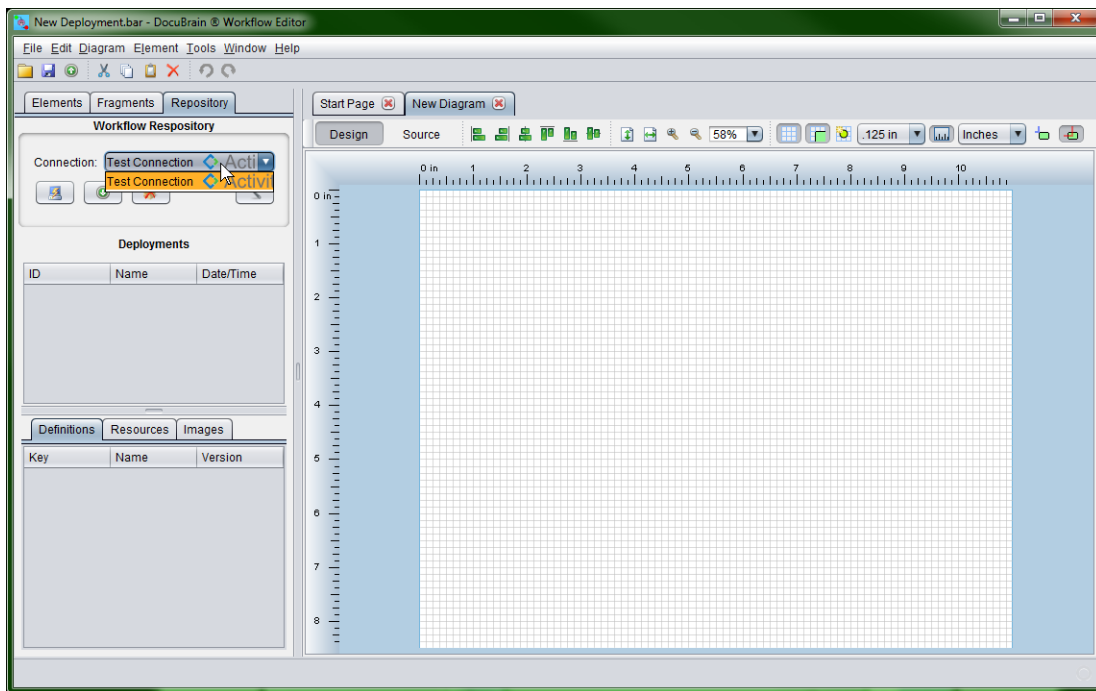
- Now enter the username and password of your TechDoc user account. If you are connecting via Single Sign On, leave both fields blank and check the "Connect using Single Sign On" checkbox. Please remember that the editor does NOT save your password and does not have the ability to do so. It is used here just to verify that the editor can make a connection to the workflow engine repository.
- Next, you'll need to decide whether or not the workflow repository is using HTTPS. If you visit the workflow engine in a web browser, you should be able to look at the beginning of the URL to see if it starts with "http://" or "https://".
- If you are required to connect through a Proxy server, you may enter the proxy server and port number (if required) in the Proxy field. You should enter something like example.com or example.com:1234. If you do not use a proxy server, this field MUST be left blank.
- You may optionally adjust the timeout in the Timeout field. The timeout is specified in seconds and is used for every call made to the repository. For instance, when you go to click the OK button on this dialog the editor will call the repository to make sure it can connect to it and retrieve its version. The version check call will be made and then the editor will wait to 30 seconds (the default) for the workflow repository to respond. You may adjust this field if you find the repository is commonly timing out due to a slow internet connection or high level of usage.


Once you have entered the settings for the connection to your repository, click the OK button. The editor will then test your connection settings to make sure it can connect to the repository. If your connection information is correct, the Create Connection dialog will close and you will be looking at the Manage Connections dialog again. If you instead encounter an error message, please read the message and verify your connection information is correct. Click the Close button on the Manage Connections dialog.

#### 4.4. Deploying to a Workflow Engine Repository

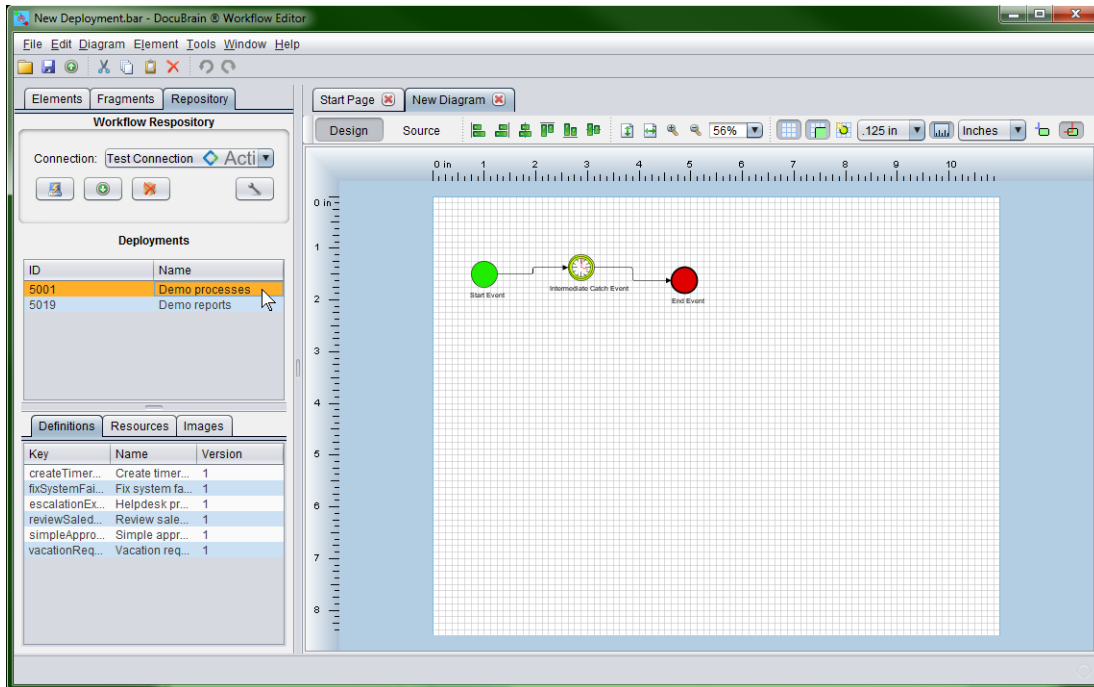
In this section, we will cover uploading a deployment to a workflow engine repository. Locate a test deployment to upload or use the deployment you created in the [previous section](#). To open the deployment in the editor, click Open Deployment on the File menu. On the Open Deployment dialog, navigate to the deployment, select it and then click the Open button. Once a deployment is open in the editor it may be uploaded to workflow engine repository.

On the main editor window, click the Repository tab on the Left Panel and select your connection from the Connection drop down box.

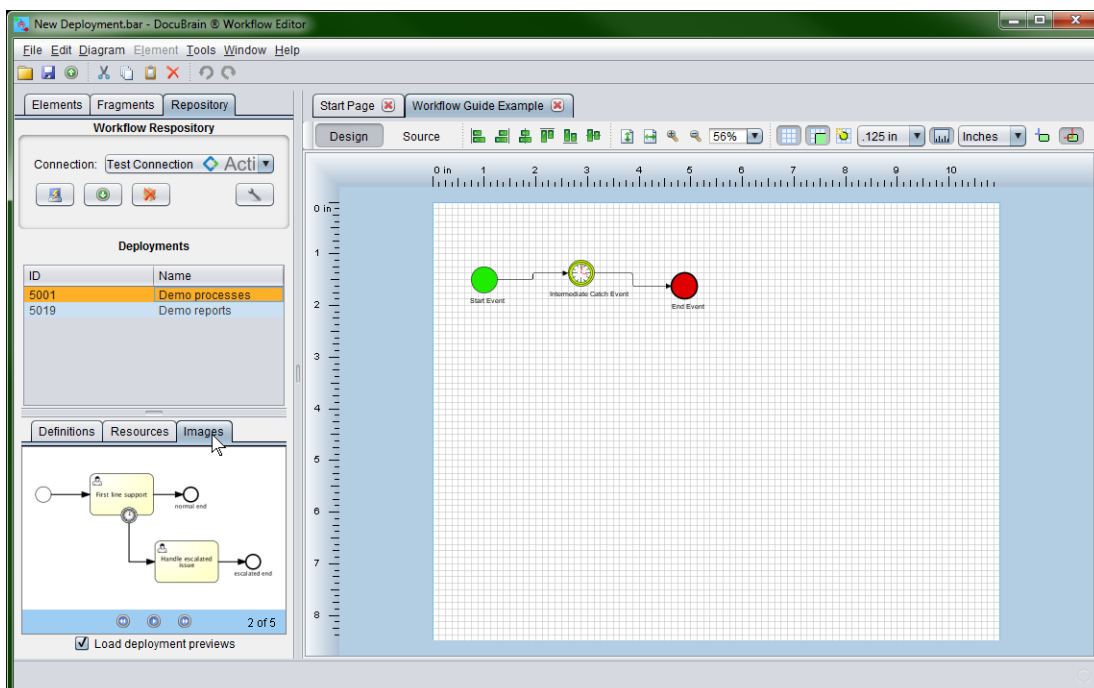



On the Repository tab, click the connect button  to connect to the workflow engine repository. When you connect, the editor will download the information for all of the workflow deployments and processes that you have access to see. If the workflow engine does not yet contain a deployment, the list will be empty. Below is default content you will see if you connect to a TechDoc workflow repository.







Notice when you click a deployment listed in the Deployments section, you can view the process definitions it contains on the Definitions tab below it. You may also view all of the resources in the deployment; typically, this list will contain all of the bpmn20.xml files, all of the thumbnails for each process (usually PNG files) and any other miscellaneous resource files the deployment might contain. Additionally, you can click the Images tab and view the thumbnails of the processes if the "Load deployment previews" checkbox is selected.

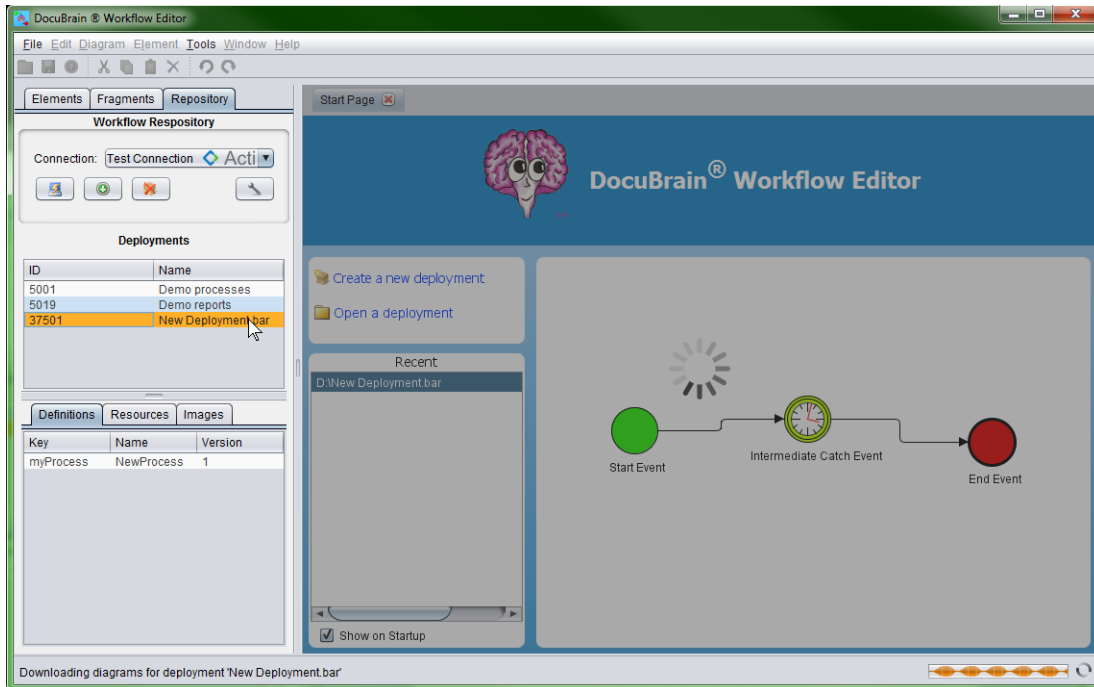


Now that we have connected to the TechDoc repository, let's upload our deployment. To do this, simply make sure your deployment is loaded in the editor and then click the upload button  on the main toolbar or select Upload Deployment from the File menu. The editor will then ask you a few questions to make sure you are uploading to the repository that you mean to and ask you for a name for the deployment (the package as a whole; the name of the BAR file). Once you have answered these questions, the editor will upload your deployment on the workflow engine. You will notice that the editor then calls the workflow repository and refreshes the list of Deployments. If everything went smoothly and your deployment contained one or more legal BPMN processes, you should see your deployment in the list. If you encounter an error while uploading your deployment, you likely have an error with one of your processes. You will need to read the error message and correct the issue before the deployment can be completed.

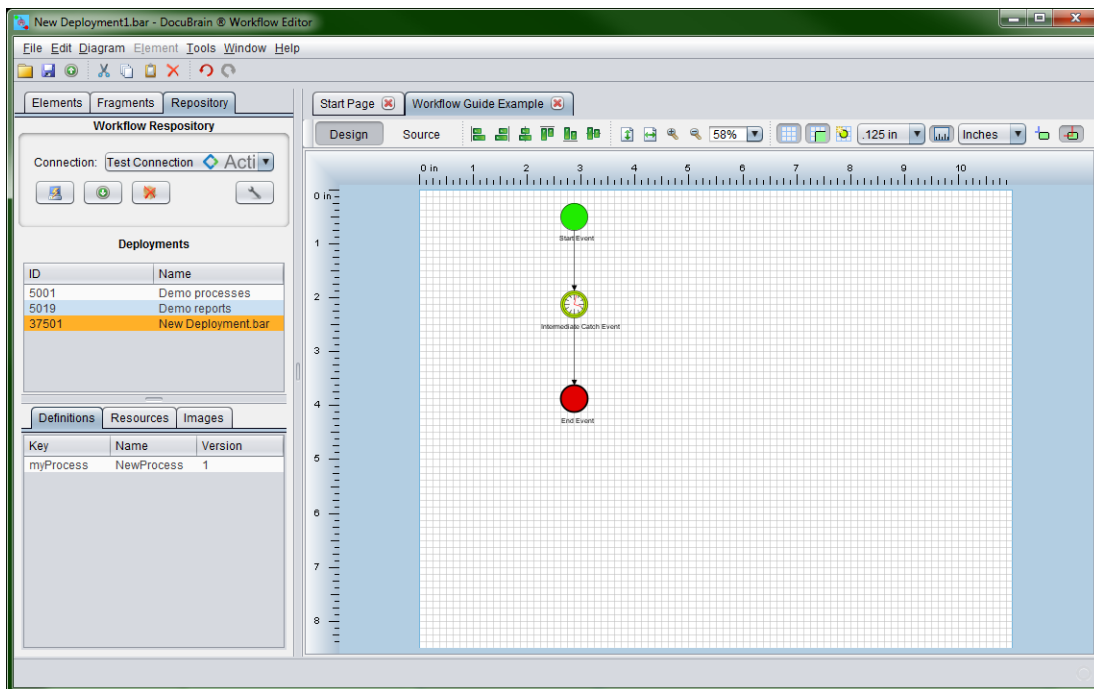
#### ***4.5. Downloading from a Workflow Engine Repository***

The DocuBrain Workflow Editor allows you not just to upload to a workflow engine repository but also download a deployment, edit it and then upload it back. In order to download from a repository, you must have first created a connection to that workflow repository. If you have not done this, refer to the previous section [Connecting to a Workflow Engine Repository](#).

First, we need to connect to the repository that we wish to download from. Click the Repository tab on the Left Panel, select a connection and then click the connect button . After the editor connects to the repository, it will return a list of the deployments in the repository that you have access to. On the Deployments list, select the deployment that you wish to download and then double click it or click the download button .



Once the deployment download has completed, it will be loaded into the editor where you can then edit any of the diagrams it contains. For this example, we will use our example deployment that we created in a [previous section](#). We've downloaded the deployment back from TechDoc and we've moved the elements around so that they are now oriented vertically.

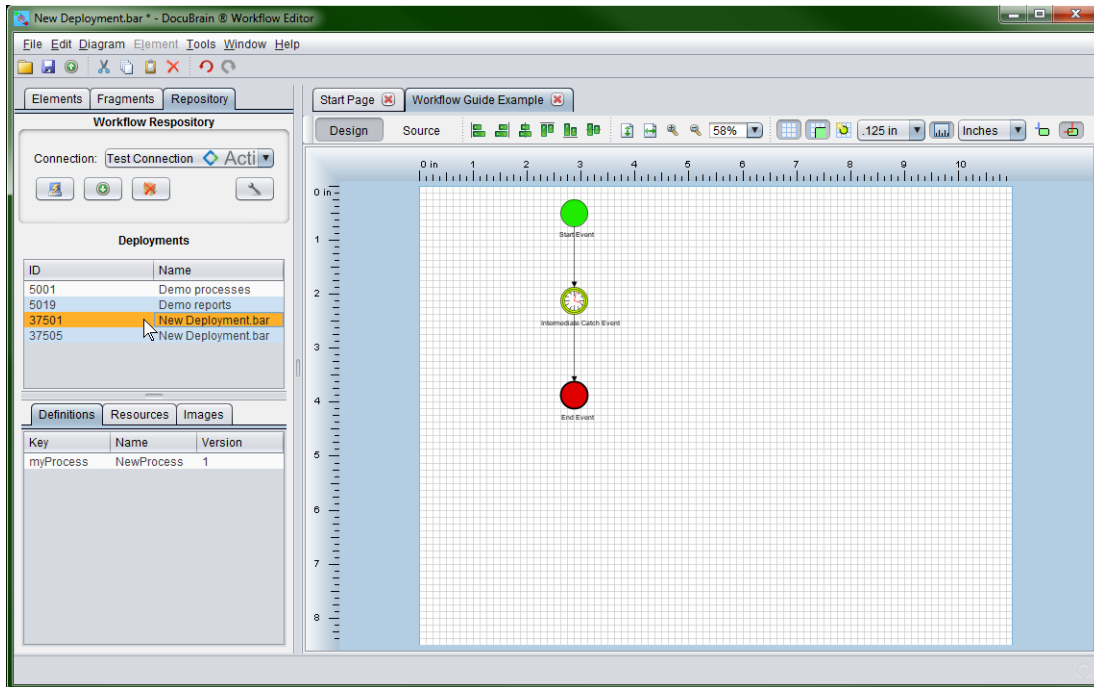


We have completed the changes we wanted to make, so we have gone ahead and saved it locally. You do not have to save a deployment locally that you have downloaded and modified from a repository. You are free to pull down deployments, make changes, and then send them right back. You will notice that the original name of the deployment was "*New Deployment.bar*" and the second modified version that we saved locally is named "*New Deployment1.bar*". We have done this so that we can maintain both the original and the second version locally. Now we will upload this deployment to TechDoc. We have a couple options:

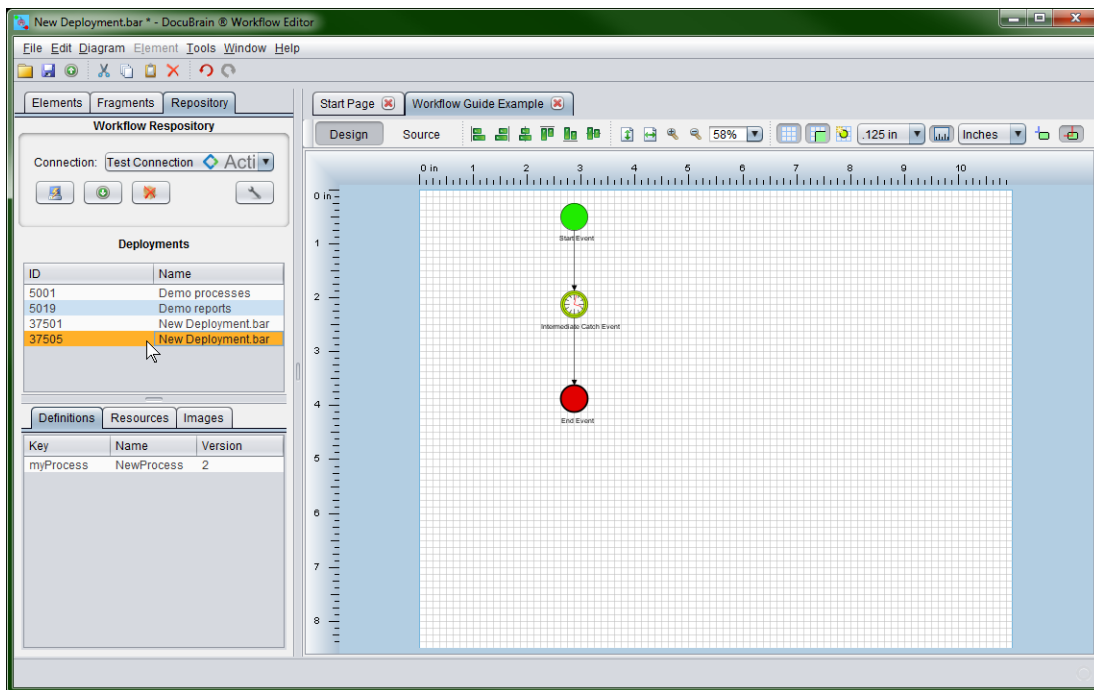
- We can delete the original deployment from the Deployments list on the Left Panel and upload this deployment keeping the deployment name the same. This will get rid of the old deployment and this deployment will become the new version 1 of the deployment.
- For the second option, we can leave the original deployment untouched in the repository and upload this deployment leaving the original version 1 and creating this modified version as version 2.

You will have to decide what is right in your case. Some people choose to wait until all running instances of a deployment have completed and then delete the original and simply replace it with the new one. However, many environments do not allow for this because they have such a large usage of workflow and cannot interrupt or have down time. In a case like this, it is most appropriate to upload a new "version 2" of your deployment. The workflow engine should then begin using the new version 2 deployment automatically. Over time, all of the instances of the original (version 1) deployment should complete. After there are no more running processes, you may delete the original so that you are left with only the new modified deployment that you wish to keep.

For this example, we will go ahead and upload the modified example deployment leaving the original deployment in TechDoc. Because we have not changed the name on the process, we will end up with a version 1 (the original) and a version 2 (the modified process). This is what you will see:




If you look at the Deployments list on the Left Panel, you will see that we have selected the first "New Deployment.bar". If you look at the Definitions list below it, you will see our process name "NewProcess" with a Version number of 1.



If you click the new deployment in the list, you'll notice that because the process contains had the same name "*NewProcess*", the process has been deemed version 2.

#### ***4.6. Multi-Process Deployments***

Another important feature of the DocuBrain Workflow Editor is its support for deployments that contain more than one workflow process (diagram). While there are few editors out there that have this ability, most workflow engines do support it. Multi-diagram deployments are very nice when you want to package a bunch of processes that all have something in common. For example, we package all of our TechDoc workflow tutorial processes in a single deployment. This way a user can deploy the single tutorial package and have access to all of the processes. Later on, when the tutorial processes are no longer needed the single deployment can be removed.

Additional processes can be added to a deployment, by simply opening a deployment and then selecting Add Diagram from the File menu. This will add a new empty diagram to the currently open deployment. If needed, a diagram can be removed from a deployment by clicking its tab, and then clicking the  button on the tab or selecting Remove Diagram from the File menu.

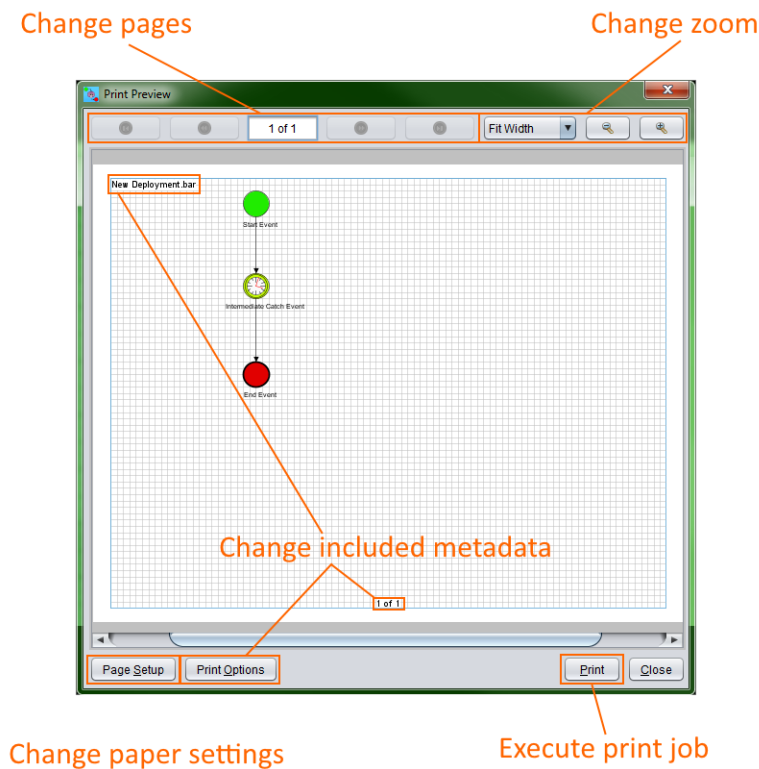
#### ***4.7. Importing and Exporting Diagrams***

In addition to multi-process deployments, diagram can be imported or exported to and from a deployment. Currently, the DocuBrain Workflow Editor supports importing: *.bpmn*, *.bpmn20.xml*, and *XML* files. This feature is particularly handy when you want to import individual diagrams from another deployment or import a diagram from another editor. Please keep in mind, when importing from another editor, you may have to make correction to a diagram once it has been imported if it contains customizations specific to that editor or contains illegal code. Though many BPMN process editors follow the BPMN 2.0 specification, not all editors build processes in the same manner. Additionally, not all workflow engines support the same BPMN elements and most have customized attributes they use that are not defined in the BPMN 2.0 specification.

With all of this in mind, you can import a diagram in your deployment by simply clicking Import Diagram from the File menu, selecting a BPMN file, and then clicking Open. The editor will load the BPMN file and add it as an additional diagram to your deployment. In addition, you may export a diagram from your deployment by opening the deployment, selecting the diagram that you wish to export and then selecting Export Diagram from the File menu. If you encounter any issues importing or exporting diagrams or have a need to support importing from another editor or workflow editor, please voice this on our website using the Submit Feedback action on the Help menu.

## 4.8. Printing a Diagram

The DocuBrain Workflow Editor supports printing the diagrams of a deployment. To print a diagram, open/download a deployment and then select Print from the File menu. Once Print Preview dialog displays, you can preview how the print out will look using the various controls on the dialog.

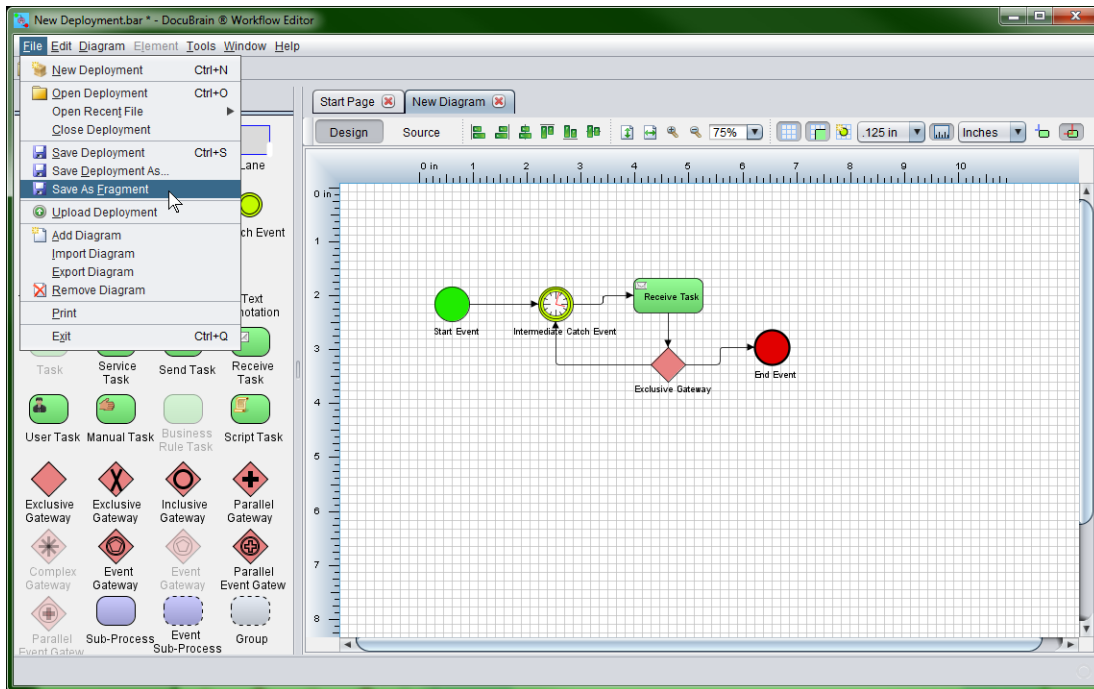


Each diagram of a deployment will be shown as a separate page in the print job. You can navigate through the pages using the buttons on the top left of the dialog. You may also zoom the diagram in and out if needed. If you need to change the print margins or paper size, click the Page Setup button. Additionally, the editor supports printing different metadata about the deployment on six places on the page, if you wish to change the title, add the date and time, etc. click the Print Options button to do so. When you are satisfied with the preview of the print job, click the Print button to begin the print job.

After clicking the Print button, you will see the print window that you usually see when printing from any other application. This print window is not a part of the DocuBrain Workflow Editor, but it generated by your operating system (Windows, Linux, Mac, etc.). Typically, you can just click the Print/OK button to begin the print job however, if you need to change the target printer, print quality, etc., you can do it there.

## 4.9. Working with Fragments

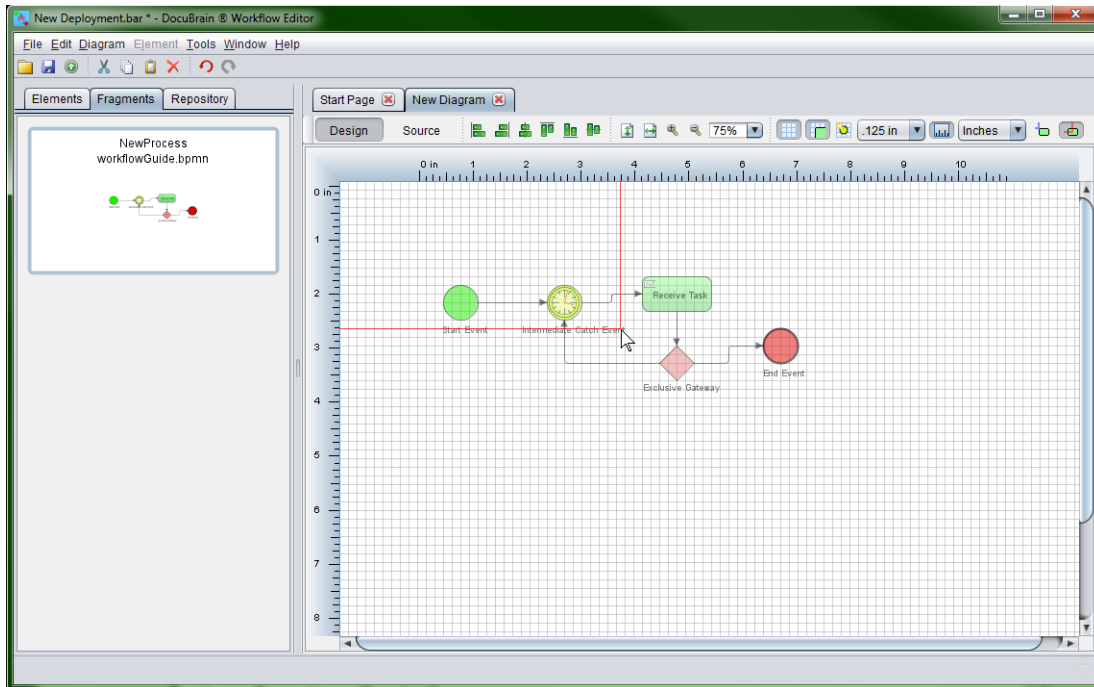
The DocuBrain Workflow Editor also supports working with fragments (or snippets as called by some). When you find yourself repeatedly re-creating a portion of a process, you may want to consider creating a fragment. To create a fragment, start by creating a new deployment. Now drag the elements on to the screen that you will need for your fragment. Now connect everything as needed. Once you are finished building your fragment, select "Save As Fragment" from the File menu.



Enter a name for the fragment and click Save. Consider the example above. I have a process that runs a timer once a minute, waits for a message to be received, and then decides whether to complete or loop back around. If I found myself repeatedly needing this chunk to use in many workflow processes that I create, I could save quite a bit of time creating it as a fragment so it can be reused.

A fragment can be used any time a deployment is open. Simply click the Fragments tab on the Left Panel, then drag a fragment from the list on to your diagram like so:

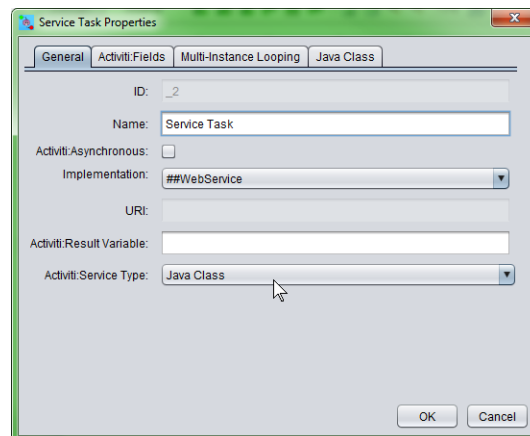




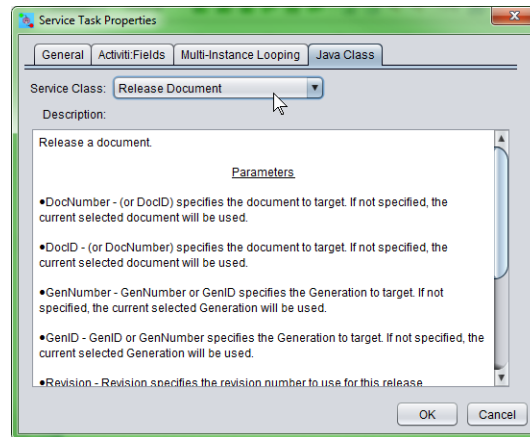
#### 4.10. *Calling TechDoc/BPMN Service Task Operations*

The DocuBrain Workflow Editor supports calling Java class operations using a BPMN Service Task. This is the most common way to perform operations that have been specifically designed for your workflow engine. In this section, we will cover how to call TechDoc operations (also called Activities) from a workflow process.

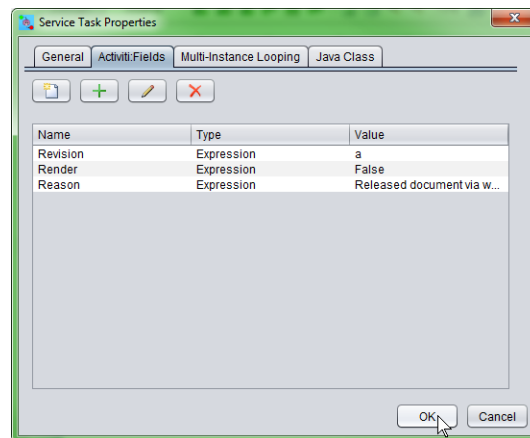
Let us build a simple process that releases a document in TechDoc when it executes. Start by dragging on to the diagram and connecting a Start Event, Service Task and End Event in that specific order. Now right click on the Service Task and then click Properties from the popup menu. You first need to make sure that your Service Type is set to Java Class.



After selecting Java Class as the Service Type, click the Java Class tab. In the Service Class drop down select Release Document. Notice the Description box displays the parameters this Java Class uses, what they expect and whether or not they are optional.



Now click the *Activiti:Fields* tab. Notice that a few fields have already been filled in for us. Go ahead and double click Revision and specify a revision. Double click Render and specify False. After you have done this, click the OK button.



You may now save this process and upload it to the workflow engine.

Did you notice that we did not enter a *DocNumber* or *DocID* variable and we also did not enter a *GenNumber* or *GenID* variable? If you do not specify a document identifier and a generation identifier (when a generation is needed), the workflow engine will try to assume the document and generation to use. For instance, say you created a trigger in the TechDoc Document Manager that ran this workflow process every time you created

a document with the Document Type Invoice. You then created an Invoice document in the TechDoc Document Manager. The workflow engine would then start this workflow process and use the document ID and generation ID of the document you just created.

Most of the time it does not make sense to explicitly set the document and generation in a workflow process since the process would only be good for a single document and generation. You could however use workflow variables to set them from other data in the process. For example, you could specify *DocID* to be *\${myVariable}* to tell the workflow engine that it should find the current value for *myVariable* and set *DocID* using that variable.

## 5. TechDoc Service Task Operations

In this section, we will discuss the TechDoc operations that are available for use from a Service Task within a workflow Process. If you are not familiar with using Service Tasks to call TechDoc Activities, please first read the [previous section](#).

### 5.1. *Add Access Association*

Add Access Association allows you to add access associations to a Document for multiple Groups, Users, and Remote Users in a single call.

Service Class: *wm.activity.AddAccessAssociation*

Fields:

- *Groups* – (Optional) – The *Groups* field is used to specify access to be added for one or more Groups separated by semicolons. The value of this field should be specified as:

*groupName:accessSetting, accessSetting;*

For example, to associate a Group named *TestGroup* with Read and Modify access, you would specify:

*TestGroup:Read,Modify;*

If you wanted to add an additional Group named *AnotherGroup* to the previous example with Owner access you would specify:

*TestGroup:Read,Modify;AnotherGroup:Owner;*

- *Users* – (Optional) – The *Users* field is used to specify access to be added for one or more Users separated by semicolons. The value of this field should be specified just like the *Groups* field:

*username:accessSetting, accessSetting;*

For Example, to associate a User named Joe with Delete access, you would specify:

*Joe>Delete;*

If you wanted to add an additional User named Bob to the previous example with Reserve/Replace and Delete access you would specify:

*Joe>Delete;Bob:Reserve/Replace,Delete;*

- *RemoteUsers* – (Optional) – The *RemoteUsers* field is used to specify access to be added for one or more Remote Users separated by semicolons. Remote Users do not have access values when associating. Any associated Remote user is given Read access only. The value of this field should be specified as:

*authenticatorName\username;*

For example, to associate a Remote User named Joe that authenticates using the Authenticator named *TestAuthenticator* you would specify:

*TestAuthenticator\Joe;*

Remote Users can also be specified as Remote User name @ Authenticator name. For example:

*joe@TestAuthenticator;*

- *Reason* – (Mandatory) – The *Reason* field must be specified when calling this operation. The value entered should be treated just as a Reason field in TechDoc explaining why you are performing this operation.

At the time of the writing of this guide, the valid Document Access values are

- None – A User has no access to the Document.
- Read – A User can view the attributes of this Document, view the attributes of this Document's Generations and Renditions and fetch this Document's Generations and Renditions.
- Modify – A User can modify the attributes of this Document and this Document's Generations and Renditions. The User must also have the Modify Document privilege.
- Delete – A User can delete this Document. The User must also have the Delete Document privilege.
- Reserve/Replace – A User can reserve and replace the Document. The User must also have the Create Generation privilege.

- Owner – A User can act as the owner of this Document. The User will be able to do anything to the Document that the owner can, as long as the User's privileges allow it.

## 5.2. *Add Commenter Association*

Add Commenter Association allows you to add commenter associations to a Document for multiple Groups and Users in a single call.

Service Class: *wm.activity.AddCommenterAssociation*

Fields:

- *Groups* – (Optional) – The *Groups* field is used to specify one or more Groups to be added separated by semicolons. The value of this field should be specified as:

*groupName;groupName;*

For example, to associate a Group named TestGroup you would specify:

*TestGroup;*

If you wanted to add an additional Group named AnotherGroup to the previous example, you would specify:

*TestGroup;AnotherGroup;*

- *Users* – (Optional) – The *Users* field is used to specify one or more Users to be added separated by semicolons. The value of this field should be specified just like the Groups field:

*username;username;*

For Example, to associate a User named Joe, you would specify:

*Joe;*

If you wanted to add an additional User named Bob to the previous example, you would specify:

*Joe;Bob;*

- *Reason* – (Mandatory) – The *Reason* field must be specified when calling this operation. The value entered should be treated just as a Reason field in TechDoc explaining why you are performing this operation.

### 5.3. *Add Distribution Association*

Add Distribution Association allows you to add distribution associations to a Document for multiple Groups, Users, and Remote Emails in a single call.

Service Class: *wm.activity.AddDistributionAssociation*

Fields:

- *Groups* – (Optional) – The *Groups* field is used to specify one or more Groups to be added separated by semicolons. The value of this field should be specified as:

*groupName;groupName;*

For example, to associate a Group named TestGroup you would specify:

*TestGroup;*

If you wanted to add an additional Group named AnotherGroup to the previous example, you would specify:

*TestGroup;AnotherGroup;*

- *Users* – (Optional) – The *Users* field is used to specify one or more Users to be added separated by semicolons. The value of this field should be specified just like the Groups field:

*username;username;*

For Example, to associate a User named Joe, you would specify:

*Joe;*

If you wanted to add an additional User named Bob to the previous example, you would specify:

*Joe;Bob;*

- *RemoteEmails* – (Optional) – The *RemoteEmails* field is used to specify one or more Remote Emails to be added separated by semicolons. The value of this field should be specified just like Groups and Users:

*remoteEmail;remoteEmail;*

For example, to associate the Remote Email joe@somewhere.com you would specify:

*joe@somewhere.com;*

If you wanted to add an additional Remote bob@somewhere.com to the previous example, you would specify:

*joe@somewhere.com;bob@somewhere.com;*

- *Reason* – (Mandatory) – The *Reason* field must be specified when calling this operation. The value entered should be treated just as a Reason field in TechDoc explaining why you are performing this operation.

## 5.4. **Add Keyword**

Add Keyword allows you to add a Keyword to a Document.

Service Class: *wm.activity.AddKeyword*

Fields:

- *KeywordName* – (Mandatory) – The *KeywordName* field is used to specify the name of the Keyword to add.
- *KeywordValue* – (Mandatory) – The *KeywordValue* field is used to specify the value this Keyword should have.
- *Reason* – (Mandatory) – The *Reason* field must be specified when calling this operation. The value entered should be treated just as a Reason field in TechDoc explaining why you are performing this operation.



## 5.5. *Add Notification Association*

Add Notification Association allows you to add notification associations to a Document for multiple Groups, Users, and Remote Emails in a single call.

Service Class: *wm.activity.AddNotificationAssociation*

Fields:

- *Groups* – (Optional) – The *Groups* field is used to specify one or more Groups to be added separated by semicolons. The value of this field should be specified as:

*groupName;groupName;*

For example, to associate a Group named TestGroup you would specify:

*TestGroup;*

If you wanted to add an additional Group named AnotherGroup to the previous example, you would specify:

*TestGroup;AnotherGroup;*

- *Users* – (Optional) – The *Users* field is used to specify one or more Users to be added separated by semicolons. The value of this field should be specified just like the Groups field:

*username;username;*

For Example, to associate a User named Joe, you would specify:

*Joe;*

If you wanted to add an additional User named Bob to the previous example, you would specify:

*Joe;Bob;*

- *RemoteEmails* – (Optional) – The *RemoteEmails* field is used to specify one or more Remote Emails to be added separated by semicolons. The value of this field should be specified just like Groups and Users:

*remoteEmail;remoteEmail;*

For example, to associate the Remote Email `joe@somewhere.com` you would specify:

`joe@somewhere.com;`

If you wanted to add an additional Remote `bob@somewhere.com` to the previous example, you would specify:

`joe@somewhere.com;bob@somewhere.com;`

- *Reason* – (Mandatory) – The *Reason* field must be specified when calling this operation. The value entered should be treated just as a Reason field in TechDoc explaining why you are performing this operation.

## 5.6. Release Document

Release Document allows you to release a Document.

Service Class: `wm.activity.ReleaseDocument`

Fields:

- *Revision* – (Mandatory) – The *Revision* field is used to specify the revision number to use for the released Document.
- *Render* – (Mandatory) – The *Render* field is used to specify whether or not the Document should be rendered. Use True or False.
- *Reason* – (Mandatory) – The *Reason* field must be specified when calling this operation. The value entered should be treated just as a Reason field in TechDoc explaining why you are performing this operation.

If the Document being released is a Metric Document, the following fields must also be specified:

- *MetricDate* – (Mandatory) – The *MetricDate* field is usually used to specify the reporting end date for a Metric. A Date in this field should be specified as:

`07/31/2004`

- *MetricStatus* – (Mandatory) – The *MetricStatus* field is used to specify the metric status of the Document at the time of releasing. A *MetricStatus* field should be specified as:

*Green - Improving*

At the time of the writing of this guide, the valid Metric Status values are:

-  Inactive
-  Green - Improving
-  Green - Staying Constant
-  Green - Worsening
-  Yellow - Improving
-  Yellow - Staying Constant
-  Yellow - Worsening
-  Red - Improving
-  Red - Staying Constant
-  Red - Worsening

## 5.7. ***Remove Access Association***

Remove Access Association allows you to remove access associations from a Document for multiple Groups, Users, and Remote Users in a single call. When removing access for a Group, User, or Remote User, only the name can be specified. When a Group, User or Remote User is specified for removal, all access associations for that Group, User, or Remote User are removed from the Document.

Service Class: *wm.activity.RemoveAccessAssociation*

Fields:

- *Groups* – (Optional) – The *Groups* field is used to specify access to be removed for one or more Groups separated by semicolons. The value of this field should be specified as:

*groupName;*

For example, to remove a Group named *TestGroup*, you would specify:

*TestGroup;*

If you wanted to add an additional Group to remove named *AnotherGroup* to the previous example, you would specify:

*TestGroup;AnotherGroup;*

- *Users* – (Optional) – The *Users* field is used to specify access to be removed for one or more Users separated by semicolons. The value of this field should be specified just like the *Groups* field:

*username;*

For Example, to remove a User named Joe, you would specify:

*Joe;*

If you wanted to add an additional User to remove named Bob to the previous example, you would specify:

*Joe;Bob;*

- *RemoteUsers* – (Optional) – The *RemoteUsers* field is used to specify access to be removed for one or more Remote Users separated by semicolons. The value of this field should be specified as:

*authenticatorName\username;*

For example, to remove a Remote User named Joe that authenticates using the Authenticator named *TestAuthenticator* you would specify:

*TestAuthenticator\Joe;*

Remote Users can also be specified as Remote User name @ Authenticator name. For example:

*joe@TestAuthenticator;*

- *Reason* – (Mandatory) – The *Reason* field must be specified when calling this operation. The value entered should be treated just as a *Reason* field in TechDoc explaining why you are performing this operation.

## 5.8. ***Remove Commenter Association***

Remove Commenter Association allows you to remove commenter associations from a Document for multiple Groups and Users in a single call.

Service Class: *wm.activity.RemoveCommenterAssociation*

Fields:

- *Groups* – (Optional) – The *Groups* field is used to specify one or more Groups to be removed separated by semicolons. The value of this field should be specified as:

*groupName;groupName;*

For example, to remove a Group named *TestGroup* you would specify:

*TestGroup;*

If you wanted to add an additional Group to remove named *AnotherGroup* to the previous example, you would specify:

*TestGroup;AnotherGroup;*

- *Users* – (Optional) – The *Users* field is used to specify one or more Users to be removed separated by semicolons. The value of this field should be specified just like the Groups field:

*username;username;*

For Example, to remove a User named Joe, you would specify:

*Joe;*

If you wanted to add an additional User to remove named Bob to the previous example, you would specify:

*Joe;Bob;*

- *Reason* – (Mandatory) – The *Reason* field must be specified when calling this operation. The value entered should be treated just as a Reason field in TechDoc explaining why you are performing this operation.

## 5.9. *Remove Distribution Association*

Remove Distribution Association allows you to remove distribution associations from a Document for multiple Groups, Users, and Remote Emails in a single call.

Service Class: *wm.activity.RemoveDistributionAssociation*

Fields:

- *Groups* – (Optional) – The *Groups* field is used to specify one or more Groups to be removed separated by semicolons. The value of this field should be specified as:

*groupName;groupName;*

For example, to remove a Group named *TestGroup* you would specify:

*TestGroup;*

If you wanted to add an additional Group to remove named *AnotherGroup* to the previous example, you would specify:

*TestGroup;AnotherGroup;*

- *Users* – (Optional) – The *Users* field is used to specify one or more Users to be removed separated by semicolons. The value of this field should be specified just like the Groups field:

*username;username;*

For Example, to remove a User named Joe, you would specify:

*Joe;*

If you wanted to add an additional User to remove named Bob to the previous example, you would specify:

*Joe;Bob;*

- *RemoteEmails* – (Optional) – The *RemoteEmails* field is used to specify one or more Remote Emails to be removed separated by semicolons. The value of this field should be specified just like Groups and Users:

*remoteEmail;remoteEmail;*

For example, to remove the Remote Email `joe@somewhere.com` you would specify:

```
joe@somewhere.com;
```

If you wanted to add the additional Remote Email to remove `bob@somewhere.com` to the previous example, you would specify:

```
joe@somewhere.com;bob@somewhere.com;
```

- *Reason* – (Mandatory) – The *Reason* field must be specified when calling this operation. The value entered should be treated just as a Reason field in TechDoc explaining why you are performing this operation.

## 5.10. **Remove Keyword**

Remove Keyword allows you to remove a Keyword from a Document.

Service Class: *wm.activity.RemoveKeyword*

Fields:

- *KeywordName* – (Mandatory) – The *KeywordName* field is used to specify the name of the Keyword to remove.
- *KeywordValue* – (Optional) – The *KeywordValue* field is used to specify the value of the Keyword to remove. If the *KeywordValue* is not specified, all Keywords with the *KeywordName* specified will be removed from the Document.
- *Reason* – (Mandatory) – The *Reason* field must be specified when calling this operation. The value entered should be treated just as a Reason field in TechDoc explaining why you are performing this operation.

## 5.11. **Remove Notification Association**

Remove Notification Association allows you to remove notification associations from a Document for multiple Groups, Users, and Remote Emails in a single call.

Service Class: *wm.activity.RemoveNotificationAssociation*

Fields:

- *Groups* – (Optional) – The *Groups* field is used to specify one or more Groups to be removed separated by semicolons. The value of this field should be specified as:

*groupName;groupName;*

For example, to remove a Group named *TestGroup* you would specify:

*TestGroup;*

If you wanted to add an additional Group to remove named *AnotherGroup* to the previous example, you would specify:

*TestGroup;AnotherGroup;*

- *Users* – (Optional) – The *Users* field is used to specify one or more Users to be removed separated by semicolons. The value of this field should be specified just like the Groups field:

*username;username;*

For Example, to remove a User named Joe, you would specify:

*Joe;*

If you wanted to add an additional User to remove named Bob to the previous example, you would specify:

*Joe;Bob;*

- *RemoteEmails* – (Optional) – The *RemoteEmails* field is used to specify one or more Remote Emails to be removed separated by semicolons. The value of this field should be specified just like Groups and Users:

*remoteEmail;remoteEmail;*

For example, to remove the Remote Email *joe@somewhere.com* you would specify:

*joe@somewhere.com;*



If you wanted to add the additional Remote Email to remove bob@somewhere.com to the previous example, you would specify:

*joe@somewhere.com;bob@somewhere.com;*

- *Reason* – (Mandatory) – The *Reason* field must be specified when calling this operation. The value entered should be treated just as a Reason field in TechDoc explaining why you are performing this operation.

## 5.12. **Replace Keyword**

Replace Keyword allows you to replace the value of an existing Keyword on a Document.

Service Class: *wm.activity.ReplaceKeyword*

### Fields:

- *KeywordName* – (Mandatory) – The *KeywordName* field is used to specify the name of the Keyword whose value should be replaced.
- *KeywordValue* – (Mandatory) – The *KeywordValue* field is used to specify the new value for the Keyword.
- *Reason* – (Mandatory) – The *Reason* field must be specified when calling this operation. The value entered should be treated just as a Reason field in TechDoc explaining why you are performing this operation.

## 5.13. **Reserve Document**

Reserve Document allows you to reserve a Document.

Service Class: *wm.activity.ReserveDocument*

### Fields:

- *Reason* – (Mandatory) – The *Reason* field must be specified when calling this operation. The value entered should be treated just as a Reason field in TechDoc explaining why you are performing this operation.

### 5.14. *Simple Web Request*

This operation gives the ability to be able to craft and send any kind of web request. This operation can be used to hand build anything from a simple HTTP get or basic a REST request to a large and complex SOAP operation call. Simple Web Request is also a great when unique web services need to be called that are propriety/one off and cannot be used by a standard Service Task because they do not follow a publicly accepted standard.

Service Class: *wm.activity.SimpleWebRequest*

Fields:

- *RequestUrl* – (Mandatory) – The request URL field must specify the web address where the request should be sent.
- *RequestMethod* – (Optional) – The HTTP request method i.e. GET, POST, PUT, etc. When a *RequestMethod* isn't specified, the request defaults to GET.
- *RequestHeaders* – (Optional) – One or more HTTP headers may be specified to be included in the request. Each header and value should be separated by a colon and if more than one header and header value is specified, they should be separated using a vertical bar. For example:

*Accept: text/plain|Accept-Charset: utf-8*

- *RequestBody* – (Optional) – The body text to include in the request if a body is needed.

### 5.15. *Select Document*

Select Document is used to “Select” the Document that all subsequent operations will target when called. For example, if you select a Document with the Document number *testDocument*, all operations called afterwards would target *testDocument*. If you were to call Reserve Document, *testDocument* would be reserved.

When calling Select Document, you must either specify the Document number or Document ID of the Document to select.

Service Class: *wm.activity.SelectDocument*

Fields:

- *DocNumber* – The *DocNumber* field is used to select a Document by its Document number.

or

- *DocID* – The *DocID* field is used to select a Document but its Document ID.
- *Reason* – (Mandatory) – The *Reason* field must be specified when calling this operation. The value entered should be treated just as a Reason field in TechDoc explaining why you are performing this operation.

## 5.16. **Send Email**

Send Email allows you to send an email from within a workflow Process.

Service Class: *wm.activity.SendEmail*

Fields:

- *To* – (Mandatory) – The *To* field is used to specify the email address where the email should be sent.
- *Subject* – (Optional) – The *Subject* field is used to specify the subject line of the email.
- *Body* – (Optional) – The *Body* field is used to specify the body of the email.
- *Cc* – (Optional) – The *Cc* field is used to specify the carbon copy recipients that should also receive this email.
- *Bcc* – (Optional) – The *Bcc* field is used to specify the blind carbon copy recipients that should also receive this email but not be visible to the other recipients.
- *Reason* – (Mandatory) – The *Reason* field must be specified when calling this operation. The value entered should be treated just as a Reason field in TechDoc explaining why you are performing this operation.

It is important to note that Send Email does allow additional body sections (paragraphs) to be added to the email. The first paragraph is specified using the *Body* field. If more paragraphs are desired after just this first one, you may use the fields:

*Body2, Body3, Body4, Body5, Body6, Body7, Body8, Body9, and Body10*

to specify up to 9 additional paragraphs for the email.

### 5.17. ***Unrelease Document***

Unrelease Document allows you to unrelease a Document.

Service Class: *wm.activity.UnreleaseDocument*

Fields:

- *Revision* – (Mandatory) – The *Revision* field is used to specify the revision number of the Document to unrelease.
- *Reason* – (Mandatory) – The *Reason* field must be specified when calling this operation. The value entered should be treated just as a Reason field in TechDoc explaining why you are performing this operation.

### 5.18. ***Unreserve Document***

Unreserve Document allows you to unreserve a Document.

Service Class: *wm.activity.UnreserveDocument*

Fields:

- *Reason* – (Mandatory) – The *Reason* field must be specified when calling this operation. The value entered should be treated just as a Reason field in TechDoc explaining why you are performing this operation.

## 6. Script Task Helper Objects

The TechDoc BPMN Script Task can be used to get and set process variables as well as build and parse complex objects and blocks of data using JavaScript. Along with all of the basic JavaScript functions, TechDoc has many additional helper objects that are available to aid in building and parsing different types of data. In the sections below, we'll cover each of the helper objects and the functionalities they expose. Each of the helper objects are prefixed with DB (short for DocuBrain) to help with any ambiguity. It is advised that any process variable naming or JavaScript variable/object naming done on the user's part try and avoid anything starting with DB.

### 6.1. *DBBase64*

The *DBBase64* object can be used to encode and decode base 64 encoded data. The data can be sourced/saved directly from/to process variables. This object contains the following two functions:

- *decodeBytes* – This function takes a base 64 encoded string and return a byte array of decoded data.
- *encodeBytes* – This function takes a byte array of data and returns a base 64 encoded string.

As an example, one may base 64 encode a process variable using the following:

```
var encodedVal = DBBase64.encodeBytes(YOUR_PROCESS_VAR);
```

### 6.2. *DBHtmlParser*

The *DBHtmlParser* object is available to help with parsing HTML. Typically, this function is used after a Simple Web Request service task call to parse the response received. For example, a simple HTTP GET can be performed and then this function can be used to locate and save various pieces of information to process variables. To get started, simply create a new *DBHtmlParser* object passing the HTML to parse:

```
var htmlParser = new DBHtmlParser(YOUR_PROCESS_VAR);
```

Then you may get the root document element:

```
var docRoot = htmlParser.getRoot();
```

At this point, you now have a *JSoup* document root object, you can begin call any of the standard functions available in the *JSoup* library. For more information on *JSoup* and all that it has to offer, please visit <https://jsoup.org>.

### 6.3. *DBJsonBuilder*

The *DBJsonBuilder* object allows you to create a full JSON object without the need to construct the formatted string by hand. *DBJsonBuilder* takes a single parameter; a boolean that specifies whether or not the base JSON object is an array. If the base object should be created as an array, true should be specified. If a plain JSON object is needed, just specify false. For example, to create a plain JSON object one should enter the following:

```
var jsonBuilder = new DBJsonBuilder(false);
```

After this, one should begin by first getting the object root:

```
var root = jsonBuilder.getRoot();
```

Once the root is obtained, one can begin adding child fields and objects. For example, to add a few string keys and values field one could do the following:

```
root.add("title", "Title Here");
root.add("body", "Body Here");
root.add("userId", "42");
```

To modify the title field:

```
root.set("title", "new title here");
```

To remove the title field all together:

```
root.remove("title");
```

Finally, child objects and arrays can be added using the same methods as listed above. For example, one could create a new child object called person and add it to the root.

```
// Create a new person object.
var personBuilder = new DBJsonBuilder(false);
var person = personBuilder.getRoot();
person.set("name", "John");
person.set("age", 30);
```

```

// Add the person object to the root object.
root.add("person", person);

// To create an array of people, first create the individuals.
var person1Builder = new DBJsonBuilder(false);
var person1 = person1Builder.getRoot();
person1.set("name", "mike");
person1.set("age", 31);

var person2Builder = new DBJsonBuilder(false);
var person2 = person2Builder.getRoot();
person2.set("name", "Jim");
person2.set("age", 32);

// Now create an array to hold the people.
var peopleBuilder = new DBJsonBuilder(true);
var people = peopleBuilder.getRoot();
people.add(person1);
people.add(person2);

// Finally add the people array to the root object.
root.add("people", people);

// When it's time to save the JSON object to a string based process variable...
YOUR_PROCESS_VAR = root.toString();

```

For more information on the operators available, please visit:

<https://github.com/ralfstx/minimal-json>

## 6.4. *DBJsonParser*

The *DBJsonParser* object allows you to parse a JSON object without the need to walk a big string variable by hand. *DBJsonParser* takes a single parameter, the JSON string to parse. For example:

```

// Create a new instance of the JSON parser.
var jsonParser = new DBJsonParser(YOUR_PROCESS_VAR);

// Get the root JSON object to start working with.
var root.JsonObjOrArray = jsonParser.getRoot();

// Getting a string value.

```

```
var stringVal = rootJsonObjOrArray.get("myString");

// Getting child object.
var childObj = rootJsonObjOrArray.get("myChildObject");
```

For more information on the operators available, please visit:

<https://github.com/ralfstx/minimal-json>

## 6.5. *DBNow*

The *DBNow* object can be used to get the date and time at the moment it is called. This is handy as a workflow process might be built right now and not executed for months and you need the date and time at the very moment the process executes. *DBNow* can be used as follows:

```
// Get the current date and time in the ISO 8601 format.
var currentDateTime = DBNow.iso8601DateTime();

// Get the current date in day-month-year format.
var currentDate = DBNow.dateTime('dd-MM-yyyy');
```

The *dateTime* function accepts anything that Java allows using Java's *SimpleDateFormat*. For more information on the format, please visit:

<https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html>

## 6.6. *DBSoapUtils*

The *DBSoapUtils* object can be when building a SOAP request to generate some of the various pieces needed to complete the request envelope. The object features the following functions:

```
// Generate a random message ID.
var msgID = DBSoapUtils.generateRandomMessageID();

// Generate random nonce bytes.
var nonceBytes = DBSoapUtils.generateRandomNonceBytes();

// Generate random nonce bytes in a base 64 encoded string.
var nonceBytes64 = DBSoapUtils.generateRandomNonceBase64();

// To generate a password digest using your nonce value (you may pass either
// nonceBytes or nonceBytes64), created (an ISO8601 date time), and a
```



```
// password.
var digest = DBSoapUtils.generatePasswordDigest(nonce, created, password);
```

This is all that exists now in the *DBSoapUtils* object but if there are any other core type functionalities you need to generate a SOAP message, let us know and we'll get them added.

## 6.7. *DBXmlBuilder*

The *DBXmlBuilder* object allows you to create a full XML object without the need to construct the formatted string by hand. For example, to create an XML object one should enter the following:

```
var xmlBuilder = new DBXmlBuilder();
```

After this, one should begin by first getting the object root:

```
var root = xmlBuilder.getRoot();
```

Once the root is obtained, one can begin constructing the document using all of the standard methods available in the Java XML API. For example, to create an element:

```
var elm = root.createElement('MyTagName');
```

To set an attribute on that element:

```
elm.setAttribute('attributeNameHere', 'attributeValueHere');
```

To remove an attribute:

```
elm.removeAttribute('attributeNameHere');
```

To remove a child element:

```
root.removeChild(elm);
```

For more information on the operators available, please visit:

<http://docs.oracle.com/javase/6/docs/api/org/w3c/dom/Document.html>

## 6.8. *DBXmlParser*

The *DBXmlParser* object allows you to parse an XML object without the need to walk a big string variable by hand. *DBXmlParser* takes a single parameter, the XML string to parse. For example:

```
// Create a new instance of the XML parser.  
var xmlParser = new DBXmlParser(YOUR_PROCESS_VAR);  
  
// Get the root XML object to start working with.  
var root = xmlParser.getRoot();
```

There are many different functions available on the *DBXmlParser* instance.

### 6.8.1. **containsAttribute**

To check if an attribute exists on an element, the following function can be used:

```
if (xmlParser.containsAttribute(elementToCheck, attributeName))
```

Pass the element to check and the name of the attribute to look for. The function will return true if the attribute exists or false if it does not.

### 6.8.2. **elementToString**

To write an element to a string, the following function can be used. Specify the element to write out, true/false to skip the XML declaration, true/false to prettily format the text.

```
var str = xmlParser.elementToString(element, false, true);
```

### 6.8.3. **findElementByID**

This function can be used to traverse a single node, many nodes, or the entire document to locate and retrieve a single node by its ID. For example, to search the whole document:

```
var elementFound = xmlParser.findElementByID(root, 'ID_HERE');
```

#### 6.8.4. **findElementByName**

This function can be used to traverse a node, node tree or document to locate and retrieve an element by its name. This function returns the first element it encounters. For example, to find and return the first element with the name 'color':

```
var element = xmlParser.findElementByName(nodeToSearch, 'color', true);
```

The last parameter specifies whether or not to ignore the namespace of the element. If true is specified, the namespace will be ignored and just 'color' should be passed. If the namespace is important, pass false and specify 'yourNameSpace:color'.

#### 6.8.5. **findElementsByAttributeValue**

This function can be used to find and return all of the child elements under the element specified that have the attribute name and value specified. Additionally, the type of element should be specified i.e. 'div', 'input', etc. For example:

```
var divList = xmlParser.findElementsByAttributeValue(root, 'div', attributeName, attributeValue);
```

The return value of this function is an array.

#### 6.8.6. **getRoot**

This function returns the root node of the document. For example:

```
var documentRoot = xmlParser.getRoot();
```

#### 6.8.7. **hideNodes**

This function can be used to hide specific nodes from showing up in subsequent parse calls. For example, if you wanted to hide all of the div nodes that had a specific attribute name and value present you could:

```
xmlParser.hideNodes(root, 'div', 'favoriteColor', 'blue');
```

In this example, any *div* in the entire document that had an attribute named *favoriteColor* with a value of *blue* would be hidden.

### 6.8.8. **parseAttributeValueAsBoolean**

This function can be called to parse and return the specified attribute on an element as a boolean. For example:

```
var boolVal = xmlParser.parseAttributeValueAsBoolean(element, attributeName, defaultValue);
```

The element and attribute name should be specified as well as the default value to return should the attribute not exist.

### 6.8.9. **parseAttributeValueAsDouble**

This function can be called to parse and return the specified attribute on an element as a double. For example:

```
var dblVal = xmlParser.parseAttributeValueAsDouble(element, attributeName, defaultValue);
```

The element and attribute name should be specified as well as the default value to return should the attribute not exist.

### 6.8.10. **parseAttributeValueAsInteger**

This function can be called to parse and return the specified attribute on an element as an integer. For example:

```
var intVal = xmlParser.parseAttributeValueAsInteger(element, attributeName, defaultValue);
```

The element and attribute name should be specified as well as the default value to return should the attribute not exist.

### 6.8.11. **parseAttributeValueAsString**

This function can be called to parse and return the specified attribute on an element as a string. For example:

```
var stringVal = xmlParser.parseAttributeValueAsString(element, attributeName, defaultValue);
```

The element and attribute name should be specified as well as the default value to return should the attribute not exist.

### 6.8.12. **parseAttributesByPrefix**

This function can be called to parse and return all of the attributes of an element whose attribute name starts with the prefix specified. For example:

```
var attributeList = xmlParser.parseAttributesByPrefix(element, 'TechDoc');
```

In the example above, a list of attributes is returned that contains only attributes with names that start with TechDoc.

### 6.8.13. **parseChildNode**

This function can be called to parse and return a single child node of an element. For example:

```
var childNode = xmlParser.parseChildNode(parentNode, 'myTag', true);
```

In the example above, the first child node with a tag name of 'myTag' is returned. The third parameter specifies whether or not to ignore the namespace. If true is specified, just 'myTag' should be specified. If false is specified, 'myNamespace:myTag' should be specified.

### 6.8.14. **parseChildNodes**

This function can be called to parse and return multiple child nodes of an element. For example:

```
var childNodes = xmlParser.parseChildNode(parentNode, 'myTag', true);
```

In the example above, any child node with a tag name of 'myTag' is returned. The third parameter specifies whether or not to ignore the namespace. If true is specified, just 'myTag' should be specified. If false is specified, 'myNamespace:myTag' should be specified.

### 6.8.15. **parseTextContent**

This function can be called to parse and return the text content of the node specified. For example:

```
var textContentString = xmlParser.parseTextContent(node);
```

### 6.8.16. **stripNamespace**

This function can be used to strip the namespace off of the string specified. For example, if you performed a *parseChildNode* to select a particular element and wanted its tag name without the namespace you could:

```
var tagNameOnly = xmlParser.stripNamespace(myElement);
```

## 7. The TechDoc Server-Side Workflow Engine Reference

TechDoc features a complete BPMN 2.0 workflow engine and management system. The engine is tightly integrated with TechDoc and provides a large set of TechDoc operations in addition to all the generic features of BPMN. The management system provides both users and administrators with a management dashboard, real-time process monitoring tools, user task delegation and much more.

### 7.1. Workflows Management Menu

The Workflows Management menu contains all of the functioning for creating and managing Workflow Deployments, Definitions, Processes, and Process Triggers. This menu also displays Workflow Tasks that need attention, Tasks that are open to claim, and process instances started by you. Currently, this menu is only available to Users with the Workflows privilege or Administrators.

**Navigation:** *[DocMgr > Workflows]*

A User with the Workflows Manager or Admin privilege will have access to all of the Workflows Management functions. A User with the Workflows privilege will only have access to their Workflow Processes.

#### 7.1.1. Creating a Workflow Deployment

Create Workflow Deployment creates a new Workflow Deployment in the Document Manager. A Workflow Deployment is a package (a .bar file) containing one or more Workflow Process Definitions their resources and/or images. In order to manually create a Workflow Deployment, you must save your process/processes from your BPMN editor as a .bar file (Activiti Business Archive File). Using the BPMN Editor, after creating a Workflow Process and saving it, select Save As from the File menu, and choose "Activiti business archive File (.bar)" from the Files of Type drop down when saving.

- The user must have the Workflows, Workflows Manager, or Admin privilege.

**Navigation:** *[DocMgr > Workflows > Side Menu > Create Deployment]*

##### **Step 1:**

1. Enter the name of the Deployment in the Name box. This is a required field. The maximum length of this field is 255 characters.
2. Select the Deployment owner in the Owner box by clicking on the down arrow and selecting a name from the list.

Note: Only an Admin can select the owner of a Deployment.

3. Enter reason for creating the Workflow Deployment in the Reason box. This is a required field. The maximum length of this field is 255 characters.
4. Click the Cancel button to cancel the command, or click the Next button to continue.

Step 2:

A Workflow Deployment file is an archive (.bar) file containing one or more Workflow Processes.

1. At the Deployment box click on the Browse... button to locate the Workflow Deployment to be load into the Document Manager.

Note: The File Upload box will be displayed.

2. In the Look in box select the drive/folder where the Deployment to be stored in the Document Manager is located. To display all of the files in the folder, click on the down arrow and select All Files (\*.\*). Click on the Deployment to be stored in the Document Manager. This will automatically insert the filename in the File name box. Click the Open button.
3. Click the Cancel button to cancel the command, click the Previous button to go back to the previous screen, or click the OK button to create the Workflow Deployment.

Notes:

- A new Workflow Deployment Mapping record will be created.
- The Workflow Deployment will be inserted into the Workflow Engine.
- A history record will be generated for creation of the Workflow Deployment.

### 7.1.2. Deleting a Workflow Deployment

Delete Workflow Deployment deletes an existing Workflow Deployment in the Document Manager. Multiple steps are required during the process in order to minimize the chances of an accidental deletion.

- The User must have Owner access to the Workflow Deployment.

**Navigation:** [*DocMgr > Workflows > Side Menu > Deployments > Select Desired Deployment > Side Menu > Delete*]

**Step 1:**



The Deployment to be deleted and the Deployment attributes are displayed.

1. Click the Cancel button to cancel the command, or click the Next button to continue.

Step 2:

The Deployment to be deleted and the Deployment attributes are displayed.

1. Enter the reason for deleting the Deployment in the Reason box. This is a required field. The maximum length of this field is 255 characters.
2. Click the Cancel button to cancel the command, click the Previous button to return to the previous screen, or click the OK button to delete the Workflow Deployment.

Notes:

- The Workflow Deployment Mapping record will be deleted.
- The Workflow Deployment will be retracted from the Workflow engine.
- A history record will be generated for deletion of the Workflow Deployment.

### 7.1.3. Modifying a Workflow Deployment

Modify Workflow Deployment modifies an existing Deployment in the Document Manager. A Deployment is a bar file package that contains one or more workflow process definitions and various resource files needed by the deployment. Typically, Workflow Deployments are created using the DocuBrain Workflow Editor and are uploaded directly to the TechDoc Document Manager where they will execute. If they need to be modified, they should be downloaded using the Workflow Editor, modified, and then re-uploaded to the Document Manager. The Modify Workflow Deployment servlet only allows an Admin to re-assign the owner of the deployment.

- The user must have the Admin privilege.

**Navigation:** [*DocMgr > Workflows > Side Menu > Deployments > Select Desired Deployment > Side Menu > Modify*]

**Step 1:**

1. If applicable, modify the owner of the Workflow Deployment by clicking on the down arrow and selecting a new owner from the list.
2. Enter reason for modifying the Workflow Deployment in the Reason box. This is a required field. The maximum length of this field is 255 characters.
3. Click the Cancel button to cancel the command, or click the OK button to modify the Workflow Deployment.

Notes:

- The existing Workflow Deployment record will be modified.
- A history record will be generated for the modification of the Workflow Deployment.

#### 7.1.4. Showing a Workflow Deployment

Displays Workflow Deployments created in the Document Manager

##### ***A Specific Workflow Deployment***


**Navigation:** [*DocMgr > Workflows > Side Menu > Deployments > Select Desired Deployment*]

The User must have read access to the Workflow Deployment.

Field Name	Description
<b>Deployment ID</b>	The ID used by the Workflow engine to uniquely identify this Deployment.
<b>Name</b>	The name of this Workflow Deployment.
<b>Created</b>	The date and time the Deployment was created.
<b>Deployment Time</b>	The date and time the Deployment was deployed in the Workflow engine.
<b>Owner</b>	The User that owns this Deployment. Click the owner's name link to display the User Info screen.

##### Process Definitions

This section contains all of the Process Definitions contained in this Deployment. The Name, Process Definition ID, Key, and Version will be displayed for each Process Definition in the Deployment.

- Click on  to Show Info of the specific Process Definition.

##### Deployment Resources


This section contains all of the resources contained in this Deployment. The filename will be displayed for each resource in the Deployment.

- Click on  to Fetch the specific Deployment resource.

### All Workflow Deployments

**Navigation:** [DocMgr > Workflows > Side Menu > Deployments > Side Menu > All Deployments]


All Workflow Deployments displays all the Deployments that have been created in the Document Manager. Note: Only a User with the Workflows Manager or Admin privilege can show All Deployments.

- The Deployment ID, Name, and Deployment Time are displayed for each Deployment.
- The number of Deployments is shown.
- Click on  to Show Info for the specific Deployment.

### My Workflow Deployments

**Navigation:** [DocMgr > Workflows > Side Menu > Deployments > Side Menu > My Deployments]

My Workflow Deployments displays all the Deployments that you currently own in the Document Manager.

- The Deployment ID, Name, and Deployment Time are displayed for each Deployment.
- The number of Deployments is shown.
- Click on  to Show Info for the specific Deployment.

## 7.1.5. Showing a Workflow Process Definition

Displays Workflow Process Definitions created in the Document Manager

### ***A Specific Workflow Process Definition***

**Navigation:** [DocMgr > Workflows > Side Menu > Process Definitions > Select Desired Process Definition]

The User must have read access to the Workflow Process Definition.

Field Name	Description
<b>Process Definition ID</b>	The ID used by the Workflow engine to uniquely identify this Process Definition. This ID is a combination of the Process Definition's Key and Version number.
<b>Name</b>	The name of this Process Definition.

<b>Version</b>	The version of number of this Process Definition. "1" represents an original deployment of a Process Definition while subsequent numbers represent each revision.
<b>Deployment ID</b>	The ID used by the Workflow engine to uniquely identify the Deployment this Process Definition belongs to. Click the Deployment ID link to display the Deployment Info screen.
<b>Deployment Name</b>	The name of the Deployment this Process Definition belongs to.
<b>Key</b>	The Key of this Process Definition. The Key is the original id given to the process in the BPMN editor it was created in.
<b>Created</b>	The date and time the Process Definition was created.
<b>Owner</b>	The User that owns this Process Definition. Click the owner's name link to display the User Info screen.
<b>Diagram Resource Name</b>	The Deployment resource name of the thumbnail for this Process Definition.
<b>Resource Name</b>	The Deployment resource name of the BPMN XML file for this Process Definition.


## Process Diagram

This is a thumbnail of the diagram representing the BPMN logic for this Process Definition.

### All Workflow Process Definitions

**Navigation:** [*DocMgr > Workflows > Side Menu > Process Definitions > Side Menu > All Process Definitions*]


All Workflow Process Definitions displays all the Process Definitions that have been created in the Document Manager. Note: Only a User with the Workflows Manager or Admin privilege can show All Process Definitions.

- The Name, Process Definition ID, Key, and Version are displayed for each Process Definition.
- The number of Process Definitions is shown.
- Click on  to Show Info for the specific Process Definition.

### My Workflow Process Definitions

**Navigation:** [*DocMgr > Workflows > Side Menu > Process Definitions > Side Menu > My Process Definitions*]


My Workflow Process Definitions displays all the Process Definitions that you currently own in the Document Manager.

- The Name, Process Definition ID, Key, and Version are displayed for each Process Definition.
- The number of Process Definitions is shown.
- Click on  to Show Info for the specific Process Definition.

Process Definitions in Deployment

**Navigation:** [*DocMgr > Workflows > Side Menu > Deployments > Select Desired Deployment > Side Menu > Process Definitions*]

Process Definitions in Deployment displays all the Process Definitions in the selected Deployment.

- The Name, Process Definition ID, Key, and Version are displayed for each Process Definition.
- The number of Process Definitions is shown.
- Click on  to Show Info for the specific Process Definition.

### 7.1.6. Showing a Workflow Process Instance

Displays Workflow Process Instances in the Document Manager

#### ***A Specific Workflow Process Instance***

**Navigation:** [*DocMgr > Workflows > Side Menu > Process Instances > Select Desired Process Instance*]

The User must have read access to the Workflow Process Instance.

Field Name	Description
<b>Name</b>	The Name of the Process Definition this Process Instance started from.
<b>Process Instance ID</b>	The ID used by the Workflow engine to uniquely identify this Process Instance.
<b>Process Definition ID</b>	The ID used by the Workflow engine to uniquely identify the Process Definition this Instance belongs to.
<b>Ended</b>	The date and time the Process Instance ended.
<b>Suspended</b>	Whether or not Process Instance is suspended.


## Process Diagram

This is a thumbnail of the diagram representing the BPMN logic for this Process Instance. The activity circled in red indicates the currently running activity.

### All Workflow Process Instances

**Navigation:** [*DocMgr > Workflows > Side Menu > Process Instances > Side Menu > All Process Instances*]


All Workflow Process Instances displays all the Process Instances in the Document Manager. Note: Only a User with the Workflows Manager or Admin privilege can show All Process Instances.

- The Name, Process Instance ID, Process Definition ID, Suspended, and Ended, values are displayed for each Process Instance.
- The number of Process Instances is shown.
- Click on  to Show Info for the specific Process Instance.

### All Workflow Process Instances that I Started

**Navigation:** [*DocMgr > Workflows > Side Menu > Process Instances > Side Menu > My Process Instances*]


All Workflow Process Instances that I Started displays all the Process Instances in the Document Manager that you started.

- The Name, Process Instance ID, Process Definition ID, Suspended, and Ended, values are displayed for each Process Instance.
- The number of Process Instances is shown.
- Click on  to Show Info for the specific Process Instance.

### Process Instances of Definition

**Navigation:** [*DocMgr > Workflows > Side Menu > Process Definitions > Select Desired Process Definition > Side Menu > Process Instances*]

Process Instances of Definition displays all the Process Instances in the Document Manager of the selected Process Definition.

- The Name, Process Instance ID, Process Definition ID, Suspended, and Ended, values are displayed for each Process Instance.
- The number of Process Instances is shown.
- Click on  to Show Info for the specific Process Instance.

### 7.1.7. Showing Workflow Activity

Show Workflow Activity provides a way to view the BPMN specific information for an individual Workflow activity that is part of a Workflow process. The information displayed will vary based on the type of activity being displayed.

- The User must have the Workflows privilege with Read access to the Workflow process.

**Navigation:** [*DocMgr > Workflows > Side Menu > Show Process Instances > Select Desired Process > Click on the Activity to Display*]

#### **Step 1:**

The information shown for a Workflow activity will vary depending on the type of activity being displayed. All activities do have an ID, Name, Type, and Documentation field, and most activities will have one or more incoming and outgoing sequence flows. The information displayed will directly correlate to the information displayed when viewing the activity in the BPMN Editor. The ID shown is the ID that uniquely identifies an activity in a Workflow process. Any reference to this activity by other activities or connecting sequence flows will refer to this activity using its ID.

### 7.1.8. Starting a Workflow Process Instance

Start Workflow Process Instance is used to start a Process in the Document Manager. A Process can be started from a Process Definition or started manually on a Document or Generation.

- The User must have read access to the Process Definition.

#### **Starting a Process from Process Definition**

**Navigation:** [*DocMgr > Workflows > Side Menu > Process Definitions > Select Desired Process Definition > Side Menu > Start Instance*]

When starting a Process Instance directly from a Process Definition, you will be redirected back to the Process Definition unless the Process Definition requires you to complete a starting form. Starting forms are used in a Process to take in and set initial data for the Process. If a form is displayed, follow the steps below.

1. Follow any instructions displayed for this starting form.
2. Complete each field of the starting form that is required.
3. Click the Cancel button to cancel the command, or click the OK button to start the Process.

Notes:

- An instance of the Workflow Process will be started on the Workflow Engine.

Starting a Process on a Document or Generation

*Navigation: [DocMgr > Explorer > Select Desired Document or Generation > Side Menu > Start Process]*

When starting a Process on a Document or Generation, the Document and/or Generation will be passed into the Process. All subsequent commands in the Process will target that Document and/or Generation.

- The User must have read access to the Document or Generation.

All of the available Processes to start are displayed.

1. Select a Process to start in the Process Trigger box by clicking the down arrow and choosing a Process Trigger from the list. The Process set in the Process Trigger selected will be the Process that is started. Only Process Triggers that have their Command set to "Start Process" can be started manually.
2. Enter the reason for starting the Process in the Reason box. Reason is a required field. The maximum length of this field is 255 characters.
3. Click the Cancel button to cancel the command, or click the OK button to start the Process.

Notes:

- An instance of the Workflow Process will be started on the Workflow Engine.
- A history record will be generated for starting the Workflow Process.

### 7.1.9. Activating a Workflow Process Instance

Activate Workflow Process Instance is used to wake up a Process Instance that has been previously suspended by a User. This is sometimes useful because it allows you to stop and start a process. For example, if a Process Instance looks as though it might time out because of a timer task waiting on another condition (like a Document to be created) and you know the condition will be met shortly, you can suspend the Process Instance and then re-activate it afterwards so that you don't have to restart the entire Process over again.

### 7.1.10. Suspending a Workflow Process Instance

Suspend Workflow Process Instance is used to pause the execution of a Process Instance. This is sometimes useful because it allows you to stop and start a process. For example, if a Process Instance looks as though it might time out because of a timer task waiting on



another condition (like a Document to be created) and you know the condition will be met shortly, you can suspend the Process Instance and then re-activate it afterwards so that you don't have to restart the entire Process over again.

### 7.1.11. Deleting a Workflow Process Instance

Delete Workflow Process Instance deletes an existing Workflow Process Instance in the Document Manager. Multiple steps are required during the process in order to minimize the chances of an accidental deletion.

- The User must have Owner access to the Workflow Process Instance.

**Navigation:** [*DocMgr > Workflows > Side Menu > Process Instances > Select Desired Process Instance > Side Menu > Delete*]

#### **Step 1:**

The Process Instance to be deleted and the Process Instance attributes are displayed.

1. Click the Cancel button to cancel the command, or click the Next button to continue.

#### Step 2:

The Process Instance to be deleted and the Process Instance attributes are displayed.

1. Enter the reason for deleting the Process Instance in the Reason box. This is a required field. The maximum length of this field is 255 characters.
2. Click the Cancel button to cancel the command, click the Previous button to return to the previous screen, or click the OK button to delete the Workflow Process Instance.

#### Notes:

- The Workflow Process Instance will be deleted from the Workflow engine.
- A history record will be generated for deletion of the Workflow Process Instance.

### 7.1.12. Modifying Workflow Process Instance Variables

Modify Workflow Process Instance Variables allows the variables of a workflow process instance to be modified. Existing variables can be edited or removed and new variables can be added. This ability can be very handy for a process instance that has become stuck because of a business process change, design issue, etc.

#### Notes:

Before the variables of a workflow process instance can be modified, the instance must first be suspended.

After modifying the variables of a process instance, the instance will be activated again just before the variable changes are saved.

- The user must have the Workflows, Workflows Manager or Admin privilege.
- The user must have Owner access to the process.

**Navigation:** [*DocMgr > Workflows > Side Menu > Show... > Process Instances > Select Desired Process Instance > Side Menu > Modify Variables*]

### **Step 1:**

A message is displayed that details the importance of understanding the process instance's design before any attempt at modification is made. Due to the complex nature of workflow, a single error in the adjustment, creation or removal of a variable can cause the entire process to terminate on error and become unrecoverable.

1. Click the Cancel button to cancel the command, click the Next button to continue.

### **Step 2:**

The current process instance variables and their values are displayed. All the variables can be modified or removed with the exception of current user (if applicable). Like all other areas in TechDoc the owner or current user of an object can only be modified by an Admin and if a current user variable does exist on a process instance, it cannot be removed by anyone including an Admin. Additionally, if a current user variable does not exist on a process instance, one can be added. However, keep in mind once added it cannot be removed. New variables can be added by selecting a type from the drop down, entering a name for the variable next to the drop down and finally clicking the Add button.

1. Edit the values of any variables needed.
2. Remove any variables if needed.
3. Add any new variables needed.
4. Click the Cancel button to cancel the command, click the Previous button to return to the previous screen, or click the Next button to continue.

### **Step 3:**

1. Click the Cancel button to cancel the command, click the Previous button to go back to the previous screen, or click the Next button to continue.

**Step 4:**

1. Enter the reason for the modification of workflow process instance variables in the Reason box. This is a required field. The maximum length of this field is 255 characters.
2. Click the Cancel button to cancel the command, click the Previous button to go back to the previous screen, or click the OK button to perform the modification.

**Notes:**

- The workflow process instance will be activated.
- The variables changes will be made to the workflow process instance.
- Two history records will be generated. The first record will be created for the activation of the workflow process instance. The second record will detail all of the variable changes made on the workflow process instance.

**7.1.13. Creating a Workflow Process Trigger**

Create Workflow Process Trigger creates a new Process Trigger in the Document Manager. A Process Trigger is used to trigger the start of a selected Workflow Process after a command is executed and a preset condition is met by the specified Document attributes or Keywords. For example, if you were to create a Process Trigger that specified a Workflow Process Definition named "MyFirstWorkflowProcess", the command "Create Document", and the Document Category "NS - Non-Sensitive Information", anytime a Document is created with the Document Category "NS - Non-Sensitive Information" a new instance of the "MyFirstWorkflowProcess" Process is started. When starting the instance, the Document that triggered the process becomes the "selected" document that all subsequent actions in the process will target (i.e. reserve document, release document, add keyword etc.). Therefore, when designing a Process that will be triggered by a Document, it is unnecessary to "select" the Document to work with in the Process as this will be done for you when the Process is triggered.

- The user must have the Workflows Manager or Admin privilege.

**Navigation:** *[DocMgr > Workflows > Side Menu > Create Process Trigger]*

**Step 1:**

1. Enter the Name for the Process Trigger in the Name box. The Process Trigger Name must be unique within the same Document Manager. The maximum length of this field is 255 characters.
2. Select the Process Trigger owner in the Owner box by clicking on the down arrow and selecting a name from the list.

Note: Only an Admin can select the owner of a Process Trigger.

3. Select the command this Process Trigger should act upon in the Command box by clicking on the down arrow and selecting a command from the list.
4. Select a Process Definition in either the Process Definition ID or Process Definition Key drop down. You may select either one or the other, not both. A Process Definition has both an ID and Key. The ID is unique and can belong to only one version of one Process Definition whereas the Key is not unique and is used to identify all versions of a Process Definition. Each time a Process Definition is updated, a newer version is created. If you select a Process Definition by ID, it is guaranteed that that specific Process Definition will always be used even if a newer version is uploaded. If you select a Process Definition by Key, it is guaranteed that the latest version of a Process Definition will be used.
5. Enter reason for creating the Workflow Process Trigger in the Reason box. This is a required field. The maximum length of this field is 255 characters.
6. If the command is not "Start Process", additional criteria may be added to the trigger to further limit which documents will cause the trigger to fire. If an additional criterion is desired, select a Document attribute or Keyword from either the New Doc Attribute or New Doc Keyword box, click the Add button, and enter or select the value for the new criterion that was just added.

"Start Process" does not allow any additional criteria. All other commands can optionally have one or more additional criteria.

7. Click the Cancel button to cancel the command, or click the OK button to create the Workflow Process Trigger.

Notes:

- A new Workflow Process Trigger record will be created.
- A history record will be generated for the creation of the Workflow Process Trigger.

#### 7.1.14. Modifying a Workflow Process Trigger

Modify Workflow Process Trigger modifies an existing Process Trigger in the Document Manager. A Process Trigger is used to trigger the start of a selected Workflow Process after a command is executed and a preset condition is met by the specified Document attributes or Keywords. For example, if you were to create a Process Trigger that specified a Workflow Process Definition named "MyFirstWorkflowProcess", the command "Create Document", and the Document Category "NS - Non-Sensitive Information", anytime a Document is created with the Document Category "NS - Non-Sensitive Information" a new instance of the "MyFirstWorkflowProcess" Process is started. When starting the instance, the Document that triggered the process becomes the "selected" document that all subsequent actions in the process will target (i.e. reserve document, release document, add keyword etc.). Therefore, when designing a Process that will be triggered by a Document, it is unnecessary to "select" the Document to work with in the Process as this will be done for you when the Process is triggered.

- The user must have the Workflows Manager, or Admin privilege.

**Navigation:** [*DocMgr > Workflows > Side Menu > Process Triggers > Select Desired Process Trigger > Side Menu > Modify*]

**Step 1:**

1. If applicable, modify the Name for the Process Trigger in the Name box. The Process Trigger Name must be unique within the same Document Manager. The maximum length of this field is 255 characters.
2. Select the Process Trigger owner in the Owner box by clicking on the down arrow and selecting a name from the list.

Note: Only an Admin can select the owner of a Process Trigger.

3. If applicable, modify the command this Process Trigger should act upon in the Command box by clicking on the down arrow and selecting a command from the list.
4. If applicable, modify the Process this Process Trigger should start by selecting a Process Definition in either the Process Definition ID or Process Definition Key drop down. You may select either one or the other, not both. A Process Definition has both an ID and Key. The ID is unique and can belong to only one version of one Process Definition whereas the Key is not unique and is used to identify all versions of a Process Definition. Each time a Process Definition is updated, a newer version is created. If you select a Process Definition by ID, it is guaranteed that that specific Process Definition will always be used even if a newer version is uploaded. If you select a Process Definition by Key, it is guaranteed that the latest version of a Process Definition will be used.
5. Enter reason for modifying the Workflow Process Trigger in the Reason box. This is a required field. The maximum length of this field is 255 characters.
6. If applicable, modify the Document attributes and/or Keywords used as additional criteria to determine which documents will cause this Process Trigger to fire.

"Start Process" does not allow any additional criteria. All other commands can optionally have one or more additional criteria.

7. Click the Cancel button to cancel the command, or click the OK button to modify the Workflow Process Trigger.

Notes:

- The existing Workflow Process Trigger record will be modified.
- The existing Workflow Process Trigger attribute records will be modified.

- A history record will be generated for the modification of the Workflow Process Trigger.

### 7.1.15. Deleting a Workflow Process Trigger

Delete Workflow Process Trigger deletes an existing Workflow Process Trigger in the Document Manager. Multiple steps are required during the process in order to minimize the chances of an accidental deletion.

- The User must have Owner access to the Workflow Process Trigger.

**Navigation:** [*DocMgr > Workflows > Side Menu > Process Triggers > Select Desired Process Trigger > Side Menu > Delete*]

#### **Step 1:**

The Process Trigger to be deleted and the Process Trigger attributes are displayed.

1. Click the Cancel button to cancel the command, or click the Next button to continue.

#### Step 2:

The Process Trigger to be deleted and any Process Trigger attributes are displayed.

1. Enter the reason for deleting the Process Trigger in the Reason box. This is a required field. The maximum length of this field is 255 characters.
2. Click the Cancel button to cancel the command, click the Previous button to return to the previous screen, or click the OK button to delete the Workflow Process Trigger.

#### Notes:

- The Workflow Process Trigger record will be deleted.
- Any additional Workflow Process Trigger attribute records will be deleted.
- A history record will be generated for deletion of the Workflow Process Trigger.

### 7.1.16. Showing a Workflow Process Trigger

Displays Workflow Process Triggers created in the Document Manager

#### **A Specific Workflow Process Trigger**

**Navigation:** [*DocMgr > Workflows > Side Menu > Process Triggers > Select Desired Process Trigger*]

The User must have read access to the Workflow Process Trigger.

Field Name	Description
<b>Name</b>	The name of this Process Trigger.
<b>Process Definition ID</b>	The ID used by the Workflow engine to uniquely identify the Process Definition of the Process this Trigger will start. When using an ID, it is guaranteed that that exact Process Definition will always be used. If a newer version is uploaded, this trigger will continue to use the older version you specified. This value will be empty if a Process Definition Key is used instead.
<b>Process Definition Key</b>	The Key used by the Workflow engine to identify the Process Definition of the Process this Trigger will start. When using a Key, the newest version of a Process Definition will be used. This means, after a trigger is created, every time a Process Definition is updated and a new version is created, this trigger will automatically begin use the new version in the future. This value will be empty if a Process Definition ID is used instead.
<b>Owner</b>	The User that owns this Process Trigger. Click the owner's name link to display the User Info screen.
<b>Created</b>	The date and time the Process Trigger was created.
<b>Command</b>	The command that this Process Trigger will act on.


### Workflow Process Trigger Attributes

This section contains all of the attributes contained in this Process Trigger. The attribute Type, Name, and Value will be displayed for each attribute in the Process Trigger.

#### All Workflow Process Triggers

**Navigation:** [[DocMgr](#) > [Workflows](#) > [Side Menu](#) > [Process Triggers](#) > [Side Menu](#) > [All Process Triggers](#)]


All Workflow Process Triggers displays all the Process Triggers that have been created in the Document Manager. Note: Only a User with the Workflows Manager or Admin privilege can show All Process Triggers.

- The Name and Process Definition ID are displayed for each Process Trigger.
- The number of Process Triggers is shown.
- Click on  to Show Info for the specific Process Trigger.

## My Workflow Process Triggers

**Navigation:** [*DocMgr > Workflows > Side Menu > Process Triggers > Side Menu > My Process Triggers*]

My Workflow Process Triggers displays all the Process Triggers that you currently own in the Document Manager.

- The Name and Process Definition ID are displayed for each Process Trigger.
- The number of Process Triggers is shown.
- Click on  to Show Info for the specific Process Trigger.

### 7.1.17. Showing a Workflow Task

Displays Workflow Tasks in the Document Manager

#### ***A Specific Workflow Task***

**Navigation:** [*DocMgr > Workflows > Side Menu > Tasks > Select Desired Task*]

The User must have read access to the Workflow Task.


Field Name	Description
<b>Task ID</b>	The ID used by the Workflow engine to uniquely identify this Task.
<b>Name</b>	The name of this Task.
<b>Description</b>	The description of this Task. Typically, the description is a set of instructions on how to complete the Task.
<b>Process Instance ID</b>	The ID used by the Workflow engine to uniquely identify the Process Instance this Task belongs to.
<b>Process Definition ID</b>	The ID used by the Workflow engine to uniquely identify the Process Definition this Task belongs to.
<b>Assignee</b>	The User that is assigned to completed this task.
<b>Owner</b>	The User that owns this Task. Click the owner's name link to display the User Info screen.
<b>Created</b>	The date and time the Task was created.
<b>Due Date</b>	The date and time this Task must be completed.



### All Workflow Tasks

**Navigation:** [*DocMgr > Workflows > Side Menu > Tasks > Side Menu > All Tasks*]


All Workflow Tasks displays all the Tasks in the Document Manager. Note: Only a User with the Workflows Manager or Admin privilege can show All Workflow Tasks.

- The Task ID, Task Name, and Description are displayed for each Task.
- The number of Tasks is shown.
- Click on  to Show Info for the specific Task.

### All Workflow Tasks that Need My Attention

**Navigation:** [*DocMgr > Workflows > Side Menu > Tasks > Side Menu > My Tasks*]


All Workflow Tasks that Need My Attention displays all Tasks you are assigned in the Document Manager that need your attention.

- The Task ID, Task Name, and Description are displayed for each Task.
- The number of Tasks is shown.
- Click on  to Show Info for the specific Task.

### All Workflows Tasks that are Open to Claim

**Navigation:** [*DocMgr > Workflows > Side Menu > Tasks > Side Menu > Open Tasks*]

All Workflows Tasks that are Open to Claim displays all Tasks in the Document Manager that are not yet assigned to a User and are open to claim.

- The Task ID, Task Name, and Description are displayed for each Task.
- The number of Tasks is shown.
- Click on  to Show Info for the specific Task.

## 7.1.18. Completing a Workflow Task

Complete Workflow Task is used to complete a Task that you are assigned. Tasks are open-ended and can require you do just about anything before clicking to "complete" the Task. A Task can be a questionnaire you must complete, or be a set of instructions you must follow. A Task's instructions may require you to perform a task in another application, or manually perform an operation such as writing and sending an email, making a phone call to notify someone of something, physically take an envelope or package and mail it. A Task can instruct you to do just about anything.

- The User must be the assignee on the Task.

**Navigation:** [*DocMgr > Workflows > Side Menu > Tasks > Side Menu > My Tasks > Select Desired Task > Side Menu > Complete*]

**Step 1:**

The Task to be completed is displayed. Read and follow any instructions displayed for the Task. Complete any form questions displayed. As soon as you are confident that you've completed the steps required to fulfill the request of the instructions, click the OK button.

1. Click the Cancel button to cancel the command, or click the OK button to complete this Task.

Notes:

- The Task will be completed and the Workflow Process will advance to the next activity.

**7.1.19. Claiming a Workflow Task**

Claim Workflow Task is used to claim an open Task that you are eligible to claim.

- The User must be a potential owner or be a member of a potential Group.

**Navigation:** [*DocMgr > Workflows > Side Menu > Tasks > Side Menu > Open Tasks > Select Desired Task > Side Menu > Claim*]

**Step 1:**

The Task to be claimed and the Task attributes are displayed.

1. Click the Cancel button to cancel the command, or click the Next button to continue.

Step 2:

The Task to be claimed and the Task attributes are displayed.

1. Enter the reason for claiming the Task in the Reason box. This is a required field. The maximum length of this field is 255 characters.
2. Click the Cancel button to cancel the command, click the Previous button to return to the previous screen, or click the OK button to claim the Workflow Task.

Notes:

- The Workflow Task will be assigned to you.

### 7.1.20. Assigning a Workflow Task

Assign Workflow Task is used to assign a Task to a User.

- The User must have the Workflows Manager or Admin privilege.

**Navigation:** [DocMgr > Workflows > Side Menu > Tasks > Side Menu > All Tasks > Select Desired Task > Side Menu > Assign]

#### **Step 1:**

The Task to be assigned is displayed.

1. Select an Assignee by clicking on the down arrow and selecting a name from the list.
2. Click the Cancel button to cancel the command, or click the OK button to assign this Task.

Notes:

- The Task will be assigned to the User selected.

### 7.1.21. Unassigning a Workflow Task

Unassign Workflow Task is used to unassign a Task.

- The User must have the Workflows Manager or Admin privilege.

**Navigation:** [DocMgr > Workflows > Side Menu > Tasks > Side Menu > All Tasks > Select Desired Task > Side Menu > Unassign]

#### **Step 1:**

The Task to be unassigned is displayed.

1. Click the Cancel button to cancel the command, or click the OK button to unassign this Task.

Notes:

- The Task will be unassigned.

## 8. Suggestions and Feedback

We hope you find the workflow engine to be very useful in automating processes in TechDoc. As the workflow engine is quite vast and very new, we have chosen to expose just the TechDoc operations outlined in this guide for this version. As we continue to build upon the workflow engine, we can begin to add more operations. If you have any suggestions about operations that should be added, how the workflow engine sections are laid out in TechDoc, etc., please let us know by sending us email at [suggestions@docubrain.com](mailto:suggestions@docubrain.com)

If you do happen to find a problem, please feel free to use the same email above. While we do prefer suggestions and compliments, we also accept criticism (preferably constructive :-)

At Prevo Technologies, Inc., we take the quality of our products very seriously. If you do find an issue or something that you feel can be improved upon in the TechDoc workflow engine, please let us know.