Process Director Documentation

Best Practices for Implementation

Last Updated: 2026-09-02, 16:24

Contents

Documentation Formatting Note

Text and Code Formatting Conventions

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

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

- `Code Variable`: A programming object whose value is usually determined from a command written in code.
  Example: `var formControls = BaseCurrentForm.FormControls;`

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

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

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

This is an Important item.

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

This is a note.

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

Icons

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

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

ICON	NAME	DESCRIPTION
		dropdown text in a topic.
−	Dropdown Open	An icon that, when clicked, will close the expanded dropdown text in a topic.

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

Other Conventions

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

Developing Applications with Process Director

Process Director is designed to enable Business users, BPM specialists, and other people without software development or coding experience to create complex, sophisticated, and fully functional BPM applications. As a no-code/low-code platform, Process Director doesn't require any knowledge of coding to create applications in most cases. This feature makes it much easier for non-programmers to create applications from a *technical* standpoint.

Technical proficiency is, of course, a good thing to have. But, in addition to technical proficiency with the product itself, it's also good to have a basic understanding of how software developers go about the process of designing applications. The technical process of creating objects or placing controls on a form is only a small part of the high-level process of designing applications efficiently and effectively.

In this course, we will describe that high-level process of application design, to give you a road map you can use to create any application you need. Since this course assumes that you are probably not a software developer or programmer by trade, we will not use a bunch of technical terms or processes that are common to software development. Instead, we aim to give you a simple and effective method for managing all the activities that go into building a new application.

The Customer

Every application you build will have at least two customers.

The primary customer is the person or department in your organization who asks you to build something in Process Director. This person will most likely already have a process in mind that they want to incorporate into a new application. You must ensure this customer receives an application that meets all their needs and automates the process correctly. This customer should be directly involved in the development of the application, right from the start.

The secondary customer, or customer group, is all the people who will use your application. The application must be as convenient as possible for them to use, implement any required accessibility components they need, and be usable across a variety of devices. No matter how effectively the application automates a given

process, this customer will be dissatisfied if the application is hard for them to use or understand.

The goal of the Application Design Process is to ensure that all customers are satisfied with the application you build.

Stages of Development

The Application Design Process can be broadly divided into four stages: Design, Development, Testing, and Release.

Design Stage

In the Design stage, you will gather requirements from your primary customer. In this stage, you will learn the purpose of the application, and the process you are being asked to automate. As part of this stage, you will need to ask many questions of the customer to ensure you understand the project completely. You need to learn the tasks and time constraints that will be required, what notifications must be sent, and what data you'll need to collect.

Development Stage

In the Development stage, you'll create all the required Process Director objects and configure their interactions. You'll also need to ensure the application objects are properly organized in the Content List so that they can be easily imported and exported between your development, staging, and production systems.

Testing Stage

Once the initial application is built, you move into the testing stage. Using a relatively small group of users, you will have them run the application, performing all the tasks it requires. As they do so, you will gather feedback from all customers to gather suggestions for improvement and revise the application to incorporate that feedback. This will usually be an iterative stage, as a change in one aspect of the application may prompt additional feedback in other places. Once the testing stage is complete, the primary customer will approve the application for release in your production environment.

Release Stage

In the release stage, the simplest of the four stages, you will export the application from your development environment into your production environment. Once

you've done so, you will use the Versioning feature of Process Director to create the initial production version.

Development Environments

In an ideal environment, there would be three different Process Director installations:

- A Development system that is relatively open and can be used for testing and development in a fairly unrestricted atmosphere. Development and development testing would be performed on this system.
- A Staging system that exactly mirrors the configuration, users, permissions, and content list structure of the production server. This server should replicate the production environment so that the Testing Stage can be completed in a copy of the production environment before an implementation is moved to production. Fresh replications of the Production system should be made on the Staging server before any pre-release testing is performed.
- A Production system where all the real-world operations are conducted. This is the system into which your completed application will be imported.

Conclusion

In the sections that follow we'll take a look at each stage of the Application Design Process and provide detailed instruction on how to complete each phase. By following this simple Application Design Process, you will ensure that the applications you create satisfy both your primary and secondary customers. This customer satisfaction will greatly improve the chances of success for your development projects.

Continue to the **Design Stage of the application development process**.

The Design Stage

Most development projects start when the primary customer requests an application to automate a specific process. As the developer, the first thing you'll need from the customer is a list of requirements in some sort of requirements document. This requirements document, in the best of all possible worlds, would specify what process you're trying to model, how it works, the information the customer needs, and other items, but...it probably won't.

Gathering Requirements

The first problem you need to confront in gathering requirements is a mental one. The customer knows what the process is and has probably been using it for a while. Most processes you automate are, after all, existing processes. The customer may know the process inside and out, which is good, but you probably aren't as familiar with it. Gathering requirements from subject matter experts can be tricky. The customer is usually so familiar with the process that they assume you understand some simple basics about the process that are obvious to them but aren't obvious to you at all.

So, your initial requirements document might look something like this:

Requirements

Objective: Approve student request to declare or change an academic major.

Process

1. Students ask to declare/change major by notifying their faculty advisor.
2. Faculty advisor approves or disapproves after consulting with student.
3. If Department approval is required, the department responsible for the new major must approve.
4. If the student has completed 150 or more semester hours, the Dean's office must approve.
5. Once approved, the Registrar's office will assign a new faculty advisor.
6. The new faculty advisor gets notified of the new student assignment.

This is not bad, and it does give you a lot of important information. But there's also a lot of detail missing that you'll need to get from the customer. Remember, to the customer, who is very familiar with the process, this short document tells them everything they need to know. But, that's because, at this point, the customer knows a **lot** more about the process than you.

Your job, at this point, is to expand the requirements with the details you need to build the application properly. As the requirements become more detailed, you need to document these details, and collaborate with your customer on building the detailed requirements.

As you create the additional documentation you need to build the detailed requirements, you need to share the expanded documentation with your customer, to ensure that both they and you agree on what the application will do, and how it will work. Both you and the customer should have a shared understanding of what the scope of the project entails before you begin building the application.

So, remember: *Share all of the documentation with the customer!*

With that in mind, let's look at how to go about expanding the specifications with the details you need to build an application that meets your customer's expectations.

Analyze and Question

You will need to analyze each step in the process and ask the customer questions about anything that is unclear. Let's look at the first step in the requirements and see what questions we can come up with.

Your first step may be as simple as clarifying some ambiguous language.

Questions to requestor

Question	Answer
Are steps 1 & 2 a loop?	Yes, there may need to be some back and forth between the advisor and student before the advisor approves?
Does the student pick their advisor, or does this come from some external system?	Student chooses.
Who are the approvers for each step?	We will provide a list of department approvers. Registrar's office is Diana Stuart, Barb Stanley, or Rick Rob. Dean's office is Eric Wintemute.
How do we choose whether to assign Diana, Barb, or Rick?	Let them accept the task.
Will the student and new faculty advisor need to interact in the process?	No. Their contact will be offline from the BPM system, probably email.
Etc...	

Students ask to declare/change major by notifying their faculty advisor.

Does this mean that students notify their faculty advisor and that the faculty advisor initiates the request in Process Director, or does it mean the student submits the request in Process Director, and the faculty advisor needs to be tasked with reviewing and approving it? The answer to these questions will make the application's starting activities fundamentally different.

While we won't take the time to do so, we can ask a *lot* of questions about every step in these simple requirements. Here's a little sample of what we might need to know, though:

- *Does step 2 mean that the conversation between the student and faculty advisor have to be in a timeline loop?*
- *Can steps 3 and 4 occur in parallel, or do they have to be in sequence?*

- *How do I know if the student has 150 semester hours or if the department requires approval?*

So, at this stage, you need to keep two things in mind:

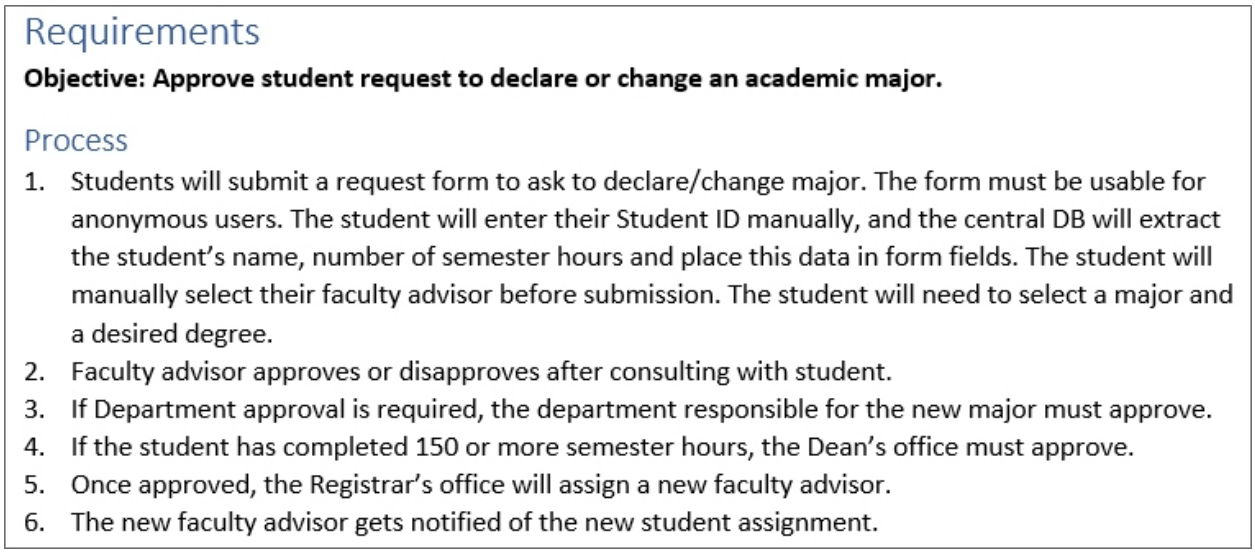
- Assume nothing. Ask questions about ***everything*** that isn't perfectly clear to you.
- As these questions are answered by the customer, document them in the requirements.

One useful thing you might do is make up a questionnaire that you can send to the customer to answer your questions.

Incorporating Answers into Your Design

As the customer answers questions, those answers must be incorporated into the specifications. The specifications document will, therefore, grow as you receive answers.

Let's say the answer to who submits the request in Step 1 is that the student will make the request in Process Director. If so, then other issues might arise that, while not immediately necessary to address, will eventually become part of the application, and should be documented.



Requirements

Objective: Approve student request to declare or change an academic major.

Process

1. Students will submit a request form to ask to declare/change major. The form must be usable for anonymous users. The student will enter their Student ID manually, and the central DB will extract the student's name, number of semester hours and place this data in form fields. The student will manually select their faculty advisor before submission. The student will need to select a major and a desired degree.
2. Faculty advisor approves or disapproves after consulting with student.
3. If Department approval is required, the department responsible for the new major must approve.
4. If the student has completed 150 or more semester hours, the Dean's office must approve.
5. Once approved, the Registrar's office will assign a new faculty advisor.
6. The new faculty advisor gets notified of the new student assignment.

- Based on how Process Director is set up in your organization, students may not be included as Process Director users, even though the faculty and staff are. That means the form will have to be set up for anonymous access.

- It also means that you won't be able to use a User Picker control to collect the student's name and other student information. That information will have to be supplied manually. If we have a student database from which we can extract information, maybe we can have the student enter their Student ID, and we can get the name and other information from that database based on the Student ID.
- We may have to use that database to find the student's faculty advisor, if it's available. If not, the student will have to supply it.

As you can see, just asking one simple question opens up a lot of different possibilities, and the requirements document will grow to incorporate the answers. Ultimately, the first step of the requirements document will become much longer.

Data to Collect

As you begin expanding the requirements to specify exactly what will happen at each step of the process, you'll need to begin documenting the data you'll need to collect as part of the process. You'll also begin mapping that data to form fields. For instance, we know that, based just on what we see here, that there are several data fields we need to collect.

Field	Field type	Options	Visible	Enabled	Required
ActionDesired	Radio	NEW PREMAJOR—to declare a new premajor CHANGE FROM PREMAJOR TO MAJOR—for approval to change from premajor to major NEW MAJOR—to declare upper division major		New form instance	X
ApprovalSection	Section		After initial submission		
DeanSection	Embedded Section		When Dean step is needed		
DeansOfficeRep	User Picker				
DeansOfficeRepComments	TextBox				
DegreeDesired	Radio	BA BFA BM BS		New form instance	X
DepartmentRep	User Picker				
DepartmentRepComments	TextBox				
DepartmentSection	Embedded Section		When Department step is needed		
FacultyAdvisor	User Picker			New form instance	X
FacultyAdvisorComments	TextBox				
MajorChangeApproval	CheckBox	Selected based on Business Value			
NewFacultyAdvisor	User Picker				
PrimaryMajor	DropDown	List from DB		New form instance	X
RegistrarSection	Embedded Section				
SectionStudentInfo	Embedded Section		When student is selected		
SemesterHours	TextBox			New form instance	
StudentID	TextBox			New form instance	
StudentName	Dropdown	List from DB		New form instance	X

- Student Name
- Student ID
- Semester Hours
- Faculty Advisor
- Desired Major
- Degree Type

Each of these pieces of data must be entered into some type of form field on the form we must create for the application. In addition, thinking about what data fields we need, we need to think about when the fields should be visible or enabled. Some fields may also be dropdowns or radio buttons with more than one option to select. You should document what these options will be, and whether you'll need to create them manually, or pull them from a data store.

One simple way to document all this is to create an Excel spreadsheet to list all the data fields we will need, and how the form will handle the data.

There are, of course, many ways to document these items, and this is just one suggestion. Feel free to document the data and field requirements in a way that makes sense to you.

Process Mapping

Just as you need to start mapping the data requirements, you also need to start mapping the process requirements at the same time. You may want to start by listing the activities that will have to be completed as part of the process, and add the following information to each task:

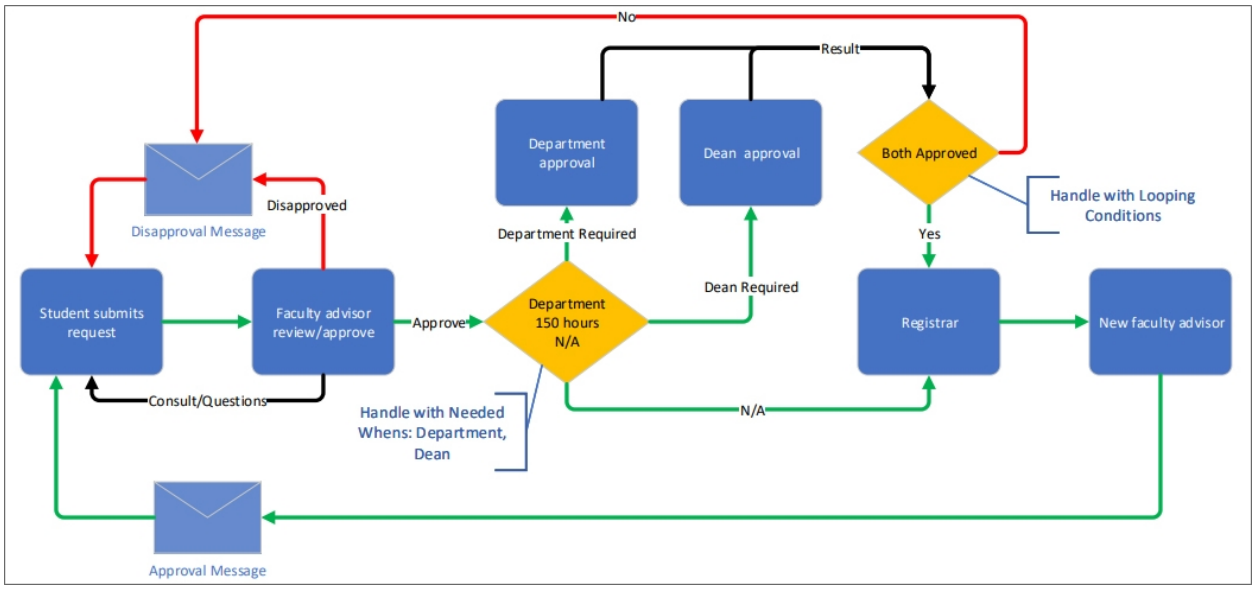
TASK	Duration	Participant	Results	Constraints: Needed When	Notes
Student Resubmission	2 days	Submitter	Resubmit	Faculty advisor requests resubmittal. Not needed otherwise.	These tasks need to loop until approved or disapproved.
Faculty advisor approval	2 days	Selected from Form field	Approve, Deny, Resubmit		
Department Approval	2 days	Dynamically built via Business Rule, Excel, or Academic Affairs (AA) DB.	Approve, Deny	Department Approval Required	Requirement is pulled from AA DB, an Excel list or a Business Rule.
Dean's Approval	3 days	Eric W.	Approve, Deny	Student has more than 150 Semester hours.	
Registrar Changes	2 days	Barb, Diana, or Rick. First to accept.	Assign Advisor		
New Faculty Advisor Acknowledgement	1 day	Selected from Form field	Acknowledge		Advisor will contact student offline, don't need a separate notification or step for student.

Notification	Sent when	To
Approval Message	When final approval is granted.	Submitter
Disapproval Message	When anyone disapproves	Faculty advisor, who will personally notify the student
Task Notification	All tasks start	All task participants

- The duration
- Participants
- Results that can be chosen by task assignees
- Start When, Needed When, or other constraints
- Any important notes to remember about each task.

Concurrently, you'll also want to list the different notifications that will be part of the application, when they'll be sent, and who will receive them.

Once again, you may find it easiest to document this in an Excel spreadsheet.



In addition, you may find that mapping the process in a diagram helps you to understand how the process will work. Diagrams like flowcharts or Gantt charts often help you visualize the main steps that have to be performed by a process. Any diagram, of course, is a simplification, and may leave out many details, but you may find them useful for organizing the process, decision steps, and notifications in a way that is easy to understand.

Application Objects

Finally, as you begin accumulating more details about the requirements, you can begin mapping out the various objects that will be needed to build the application. Obviously, we'll need a Form, a Process Timeline, and a Knowledge View or two. But there are other objects we need to consider as well.

Object	Name	Notes
Form	Change of Major Request	
Process Timeline	Change of Major Request	
Form	Email Template	Contains both required notification for approval and disapproval
Kview	Requests in Progress	Lists all active change of major request form instances
DataSource	AA DB	Source for external data
Business Value	Majors List	Fills the list of available majors
Business Value	Students List	Fills the list of students
Business Value	Major Change Approval	Sets the department approval required check box and department ID text box
Business Value	Selected Student Hours	Sets the Student ID and Semester hours field.
Business Rule	Department Approver	Assigns the department approver from a business value

For example, we know that we'll need to use some data from an external data source to find the student's name, faculty advisor, and other information. We will

need some Business Values to retrieve that data. That tells us that we need to list the Business Values as required objects for the application.

All these objects should be listed prior to developing the application. Again, a simple Excel spreadsheet can be used, though you can, of course, use whatever method of documenting these items that works best for you.

Why Do All This?

At this point, you may be wondering why all of this is necessary. Why not just start building objects in Process Director? Well, there are several good reasons for documenting all these details before you begin building the application.

Even a simple application, like this one, may have some unusual complexities. Documenting the requirements in detail prior to development ensures that you find these complexities, and plan for how to incorporate them. If you've already started building the application before you understand these complexities, you may have to rebuild a significant portion of your application, which will add to the time and cost of creating the application.

Your customer will probably want an estimate of when the application will be completed. Documenting the application in detail enables you to provide a reliable estimate of the time, and, if applicable, the cost, of building the application.

Knowing ahead of time what the scope of the application is will make building the application much easier for you. Prior to building anything in Process Director, documenting requirements in detail ensures that you already know what Form fields you'll need, what the timeline activities will be, and how they are constrained, and what objects need to be created. This foreknowledge will make the job of building the application much easier.

Finally, because Process Director objects work together so closely, specifications are a vital part of knowing how all the different objects work together, and how they depend on each other. We will discuss this in more detail in the next lesson.

Finally, documenting the application's requirements in detail, and sharing these details with your customer ensures that both you and the customer understand and agree.

Conclusion

Gathering specifications and documenting them in detail may seem like a lot of work, but it's vitally important that both you and the customer agree on what the

application entails and how it will work. Having this agreement with the customer before beginning ensures that you are both expecting the same result and working for the same goal. Once you and the customer have a shared set of expectations, you can be confident in moving forward with creating the application.

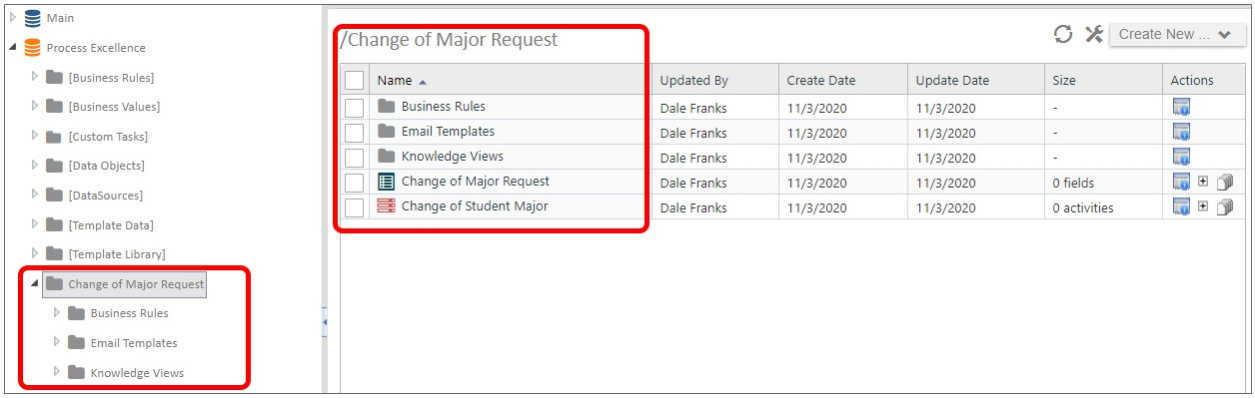
The Development Stage

The Development Stage is where you'll perform the technical work of building the application that conforms to the specifications you've gathered and documented.

Designing the Application Folders

Structuring your folders properly makes it easier to deploy individual projects or implementations. The best practice for structuring folders is to create a single parent folder to contain the entire application, and to include subfolders for various content list object types. For instance, look at the sample structure below.

Our Change of Major application is contained in a parent folder, named Change of Major Request. Inside this folder are subfolders for storing other application objects.



The image above is an example of structuring an implementation project logically. The top-level folder provides the name of the process you are modeling. All the individual Forms, Knowledge Views, etc. are organized into the lower-level folders. With this type of organization, any changes you make to the process are contained inside this folder structure, which allows you to simply export the Change of Major Request folder from your development server, and import it into production, with all of its content list objects intact, and without having to worry about interfering with any other project.

There are, of course, some exceptions to project/process-based organization. For instance, shared objects, like Datasources, Goals, and Business Values should probably each be grouped together in their own folder at the root level of the partition. You may have noticed that the Custom Tasks are organized in this fashion. These objects are considered "global" objects, which is to say that they can—and probably will be—used by many different applications.

Building Objects

Building an application usually requires that many different objects work together. For example, both a Form and Process Timeline might both require evaluation of data from a Business Value in a condition. This characteristic of many different objects working together is termed **interoperability**. Because each Process Director object can be so dependent on others to operate correctly, documenting the application's requirements in detail, as we discussed in the previous lesson, becomes even more important.

Since so many objects might be part of the interoperability between objects, it may not be possible to completely build one object at a time. A Form section might only be visible in a specific Timeline activity, while a timeline activity may not need to run if a Form field contains a specific value. Since both the form and timeline interoperate in this way, you won't be able to completely configure the form until the appropriate Process Timeline has been completed. Similarly, you won't be able to completely configure the Process Timeline until the Form field you are evaluating in the Timeline activity exists on the form. Documenting these required interactions ahead of time enables you to know how your objects will interoperate before you begin building them.

Where to Start

Now that you've gotten your application folders organized, the choice is up to you as to what objects you begin building first. Happily, there's no right or wrong way to begin, but there are a few things you might want to think about.

If your Form and Process Timeline use Business Rules or Business Values, you may wish to build those objects first so that, when you get to the point of building the Form or Process Timeline to incorporate them, they already exist.

Alternatively, since the Form is often the most complex and time-consuming object to build, you may wish to start with the Form, to configure how the Form fields are

organized, and how the form will look. Similarly, you may wish to start with the Process Timeline to ensure that the process is mapped out before creating the form.

The good news is that there is no right or wrong way to start building the application objects.

Also, since both Knowledge Views and Email Templates almost always require either Form or Process Timeline information—or both—to operate correctly, you can safely create those objects last.

Best Practices

When developing applications, there are several Best Practices that we recommend. Please refer to the Best Practices documentation topic for the full list of recommendations. While all of them may not be appropriate for every application, consider implementing them where you can.

Development Testing

While you may have a formal testing stage once the application is complete, you should be constantly testing the application during development. This testing should take place at the object level, as well as the application level.

At the object level, special attention should be paid to forms, for a couple of reasons. First, they tend to be the most complex objects in an application. So, you should ensure that the data you present from external sources is correct, visibility and enabling settings and conditions work properly, data is saved in the proper format, and all other form operations run correctly. Second, Forms are the primary user interface available to the users for interacting with the application. If the form doesn't work correctly, efficiently, and conveniently for the end user, their experience with the application will be unfavorable.

At the application level, you need to ensure that the Process Timeline runs correctly, Start When and Needed When conditions are applied properly, and that interactions between the Form, Process Timeline, and other objects run correctly.

Your testing during development needs to be comprehensive enough to ensure that, when the application is released to end users and the customers in the testing stage, all obvious errors have been corrected.

The Testing Stage

The Testing stage of development serves two useful purposes.

In a perfect world, your development testing would be rigorous enough to release a perfectly functional application to the customer. Unfortunately, in the real world this rarely ever happens. You should expect that user testing will find errors that you did not find while conducting development testing. Users will often use the application in unexpected ways, or inadvertently enter wrong data that may affect the operation of your application. No matter how rigorous your development testing may be, a larger community of users will often find errors that you missed.

Beyond finding additional errors, however, it is not unusual for a customer to realize that the way they expected the application to work turned out to be inconvenient or difficult for the end user to operate properly. The application may be exactly what the customer asked for, but they may not have fully understood how end users would respond to the application. Again, this is not uncommon. The testing stage, therefore, is useful for catching the end-user objections that arise in day-to-day use.

Setting Up the Test

In order to make the Testing Stage as productive as possible, there are a few practices you should consider.

The customer can select a relatively small number of users to take part in the testing. The best test users are those who are familiar with the existing process that you're replacing, so that they have at least a conceptual understanding of what your application is designed to do.

Test users should also be comfortable with documenting any errors, dislikes, or other comments that they can pass onto both the customer and you. The success of your testing is going to be highly reliant on the users' observations, and you and the customer should be committed to responding to the test users' concerns.

Provide some basic training on the application for the testers, both to introduce them to your application, and to ensure they understand the basics of how to use it. The training should cover the basics of using the application, without being too detailed. Don't try to train them on every single aspect of the application, to avoid coloring their perceptions about how to use it. Leave some space for them to feel comfortable with experimenting with how the application works.

Once the Testing Stage has been set up, you can export the application from the Development Server to the Staging Server. Once the application has been impor-

ted into the Staging Server, you can give the test users access to it, and allow them to begin using it.

The Test Process

The length of the Testing Stage may be highly variable. A complicated application may take much more time to test properly than a simple application. The Testing Stage should be long enough to ensure that you have enough time to have the users adequately test the application, and for you to make changes based on any user feedback you receive.

During the Testing Stage, you should regularly collect user feedback, and, with the approval of your customer, make any necessary changes to the application. You can go about this in a couple of ways.

You can have multiple phases of testing, where each phase takes a specified amount of time, during which you gather user feedback. At the end of each phase, you can make changes to the application, then start the next phase of feedback. This method provides less timely response to user feedback but enables you to collect a list of changes that can be implemented at the same time. This is a useful method for complex applications where a change in one part of the application affects another part. This method enables you to make changes in a more orderly fashion.

Alternatively, you can run the Testing Stage in a single phase and make changes immediately from user feedback as it arises. This method is more immediately responsive to user feedback but means that you may be constantly revising the application once or more per day. This method may be appropriate for a simple application, where changes may be less troublesome to implement.

In both cases, you should make all changes on your Development system, then re-import the application to the Staging System.

Customer Communication

You should communicate regularly with your customer to ensure that they understand and approve of any changes you make. Good communication with the customer about user feedback and proposed changes is very important.

User feedback is very important, but not all feedback may be wise to implement. The customer may decide that some user feedback should not be included, because the change is unacceptable for business reasons. You may decide that

some requests are not technically possible to implement. Both you and the customer must understand what feedback should and should not be implemented and why.

Once both you and the customer decide that the Testing Stage has adequately ensured the application is functional and usable, you are ready to move to the Release Stage of development.

The Release Stage

Once testing is complete, and the application has been finalized, you are ready to release the application into production.

Since the application on the Development and Staging systems should be identical, you can export the application directly from the Development Server to the Production Server.

When creating a new application, there may be other Process Director objects, like Workspaces and Meta Data categories, created as part of the application, which makes the import more complex.

To perform complex import operations like these, the Process Director objects should be imported in the following order.

- **Partition Meta Data:** Meta data can be applied to any Process Director object, which means it sits at the highest level of the hierarchy.
- **Users/Groups:** Any new users or groups that have been created on the source system, and that are used in the application, must be imported. Exports of groups and users are, unlike other objects, exported and imported as Excel files rather than PDZ files.
- **Global Content List Objects:** If you have created additional business values, datasource objects, or business rules that reside in "global" content list folders, import them to the appropriate place in the Content List before importing the application.
- **Application Content List Objects:** Once the higher-level objects have been imported, you can now import the PDZ file containing the application folder and all its contents.
- **Workspaces:** Workspaces are usually configured with specific groups or users assigned to them. Similarly, a Workspace may display an application dashboard or knowledge view, or have navigation buttons that reference application

forms. Importing Workspaces before importing the required objects will result in an import warning. The Workspace will be imported, but any missing users, groups, or application objects will be removed from the Workspace.

Failing to import objects in the proper order will result in import errors that will necessitate re-importing the objects.

Versioning

Process Director implements versioning for all objects, and each version consists of a complete snapshot of the object. Each version can be downloaded, deleted, or applied to the object to replace the current version. While any object can be versioned, BP Logix recommends creating versions at the application folder level to create a snapshot of the entire application, rather than versioning each object individually.

Once the application has been imported into the Production Server, you should open the properties of the application folder and create an initial version of the folder, which will create a version snapshot of the entire application.

You should create a new application version every time you import the application, in order to maintain a full version history with all application changes over time.

Permissions

Your Production Server should already have a permissions methodology imposed, so importing the new folder should automatically inherit the proper permissions for all content list objects.

Additional objects that exist outside of the Content List, such as Workspaces, will need to be configured to add the proper users and groups for the workspace. Users of the new application will not be able to access it until the appropriate Workspace links have been made available to them.

You may wish to use a regular user account or impersonate a user who will have access to the new application. Ensure that the user can access and run the application. If so, the application is now live in your production system, and the application development cycle is now complete.

Best Practices for Implementations

There are a number of Best Practices that you should incorporate into your implementation projects. Following these guidelines will make your implementations work more smoothly, be easier to maintain, and will enable you to make necessary changes more quickly, as personnel or processes change and evolve. In addition to these best practices, you should also familiarize yourself with the basic design concepts that are explained in the Getting Started Guide, which also contains step-by-step instructions to build a generic application. Finally, a standalone

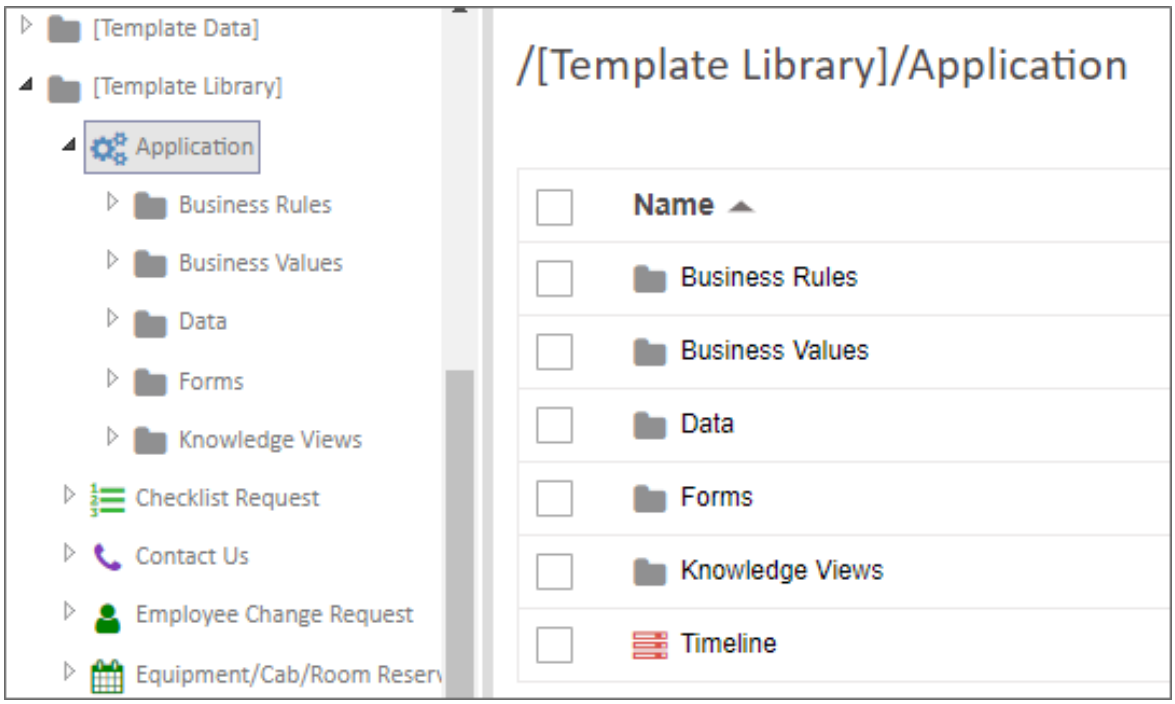
General

1. When you delete a form instance, ensure that you delete the corresponding Process Timeline instance as well.
2. Always attach objects as Process references instead of Form references, except in those cases where a form-specific attachment is required. Always use a Group Name for attachments. The Group Name is the only reliable handle we have for selecting attached objects.
3. As a general rule, you should create only one Datasource per database back end, and reuse that Datasource for all relevant implementation projects. Only create multiple Datasources when a) you are pointing to two or more databases, or b) You want to specifically restrict a Datasource to specific objects. For example, you might create a Datasource with a smaller subset of tables for use in a specific project, to ensure a class of users don't have access to more sensitive tables that might be included in the main Datasource.
4. Organize Datasources in root folders of the partition, separate from any implementation folders, so that implementers don't have to export Datasources when the implementation is exported from development to production. Exports and Imports are name-based, and relative folder position-based. Consequently, the root folder containing database sources should be mirrored on development and production servers, so that both the names and relative folder positions are exactly the same in both environments. If the mirroring is correct, implementations will be imported/exported between development and production without losing contact with the Datasources.
5. When creating conditions that require a specific value, use the "equals" (=) condition rather than the "contains" condition when possible. Only use

'contains' when a null value is desired as a return proxy for "show me everything" When building conditions testing matching strings, think hard about when to use “=” versus “contains”. Note that if the right hand side of the condition is blank, then “contains” will match all strings, whereas “=” will match none. If you're building a condition, for example, testing the result of a Process Timeline Activity, and the result might be “Accept” or “Do Not Accept”, then using a condition like “contains ‘accept’” will match either result—probably not what you intended. So it’s also a good idea to avoid results that are substrings of one another, like "Approve" and "Disapprove".

6. When configuring Process Timeline activities, enter actual instructions in the Instructions property, and descriptions into the Description property.
7. When creating new applications, place all of the application's objects, i.e., Forms, Process Timelines, etc., into a single parent folder. This makes importing/exporting applications between development and production environments much easier, as described in the Importing/Exporting Content topic of the Implementer's guide.
8. Create an application stub in the Template Library that contains all of the base objects you'd normally create for any new application, e.g., a Form and Process Timeline that are correctly linked, along with a default email template. This enables you to provide all of the initial applications objects, along with any desired Form styling or layout, to enable all new applications to inherit a desired look and feel. This stub should also include the common organizational folders for applications objects, such as Knowledge Views, Forms, etc. Create new applications directly from this stub. The example

below shows an application stub built into the Template Library.



Email Templates

1. As a guideline, use fewer email templates by using conditional sections and multiple email data controls, when applicable.
2. Use system variables to display activity names, instructions, and descriptions, rather than hard-coding them in the body of the email, so that little customization has to be done to the email template.
3. Configure a default email template in the Process Timeline definition. You can override it and use a different email template if a particular Timeline Activity requires it, by configuring the Using Email Template property on the activity's Notifications tab.
4. Use the `{EMAIL USER}` System Variable to specify the email recipient instead of `{CURR_USER}`. Email messages do not have a Current User context.
5. On Forms, use the System Variable Control rather than typing System Variables as text into the Online Form Designer. IT makes the Form's design look cleaner, and protects against inadvertent editing of the System Variable name.

Development/Testing

1. Your Development and Staging systems should be near-replicas of your productions system, including the same users, Meta Data, etc. Global objects, such as Datasource objects should have the same names, and be in the same folder locations on all environments, though they may not be otherwise configured the same. For instance, an "Employee" datasource might connect to real-time data in an employee database on the production system, but a sanitized database on a development system. But, as long as the two objects have the same name, and are stored in the same file location on both servers, imported applications that refer to these objects will import correctly, and return the appropriate data on each system.
2. Datasource objects, by the way, should not be imported between systems. They should be manually created on each system.
3. When developing, use real users as assignees for Timeline activities. In testing, use impersonation to perform the assignees' tasks. Do not assign tasks to yourself, as you need to see the process as the actual assignees will see the process.
4. In field properties for Process Timeline activities, enter actual instructions in the Instructions field, and descriptions into the Description field. (Remember, somebody else might be modifying this project after you—help them out!)
5. When you delete a form instance, ensure that you delete the corresponding Process Timeline instance and vice versa. Failure to do leaves orphaned objects in Process Director, i.e., Form instances without their Timelines. All related instance objects need to be deleted together.
6. You shouldn't ever perform testing in the production environment, except in unusual circumstances. Applications should be rigorously tested in the development environment before being imported to Production.

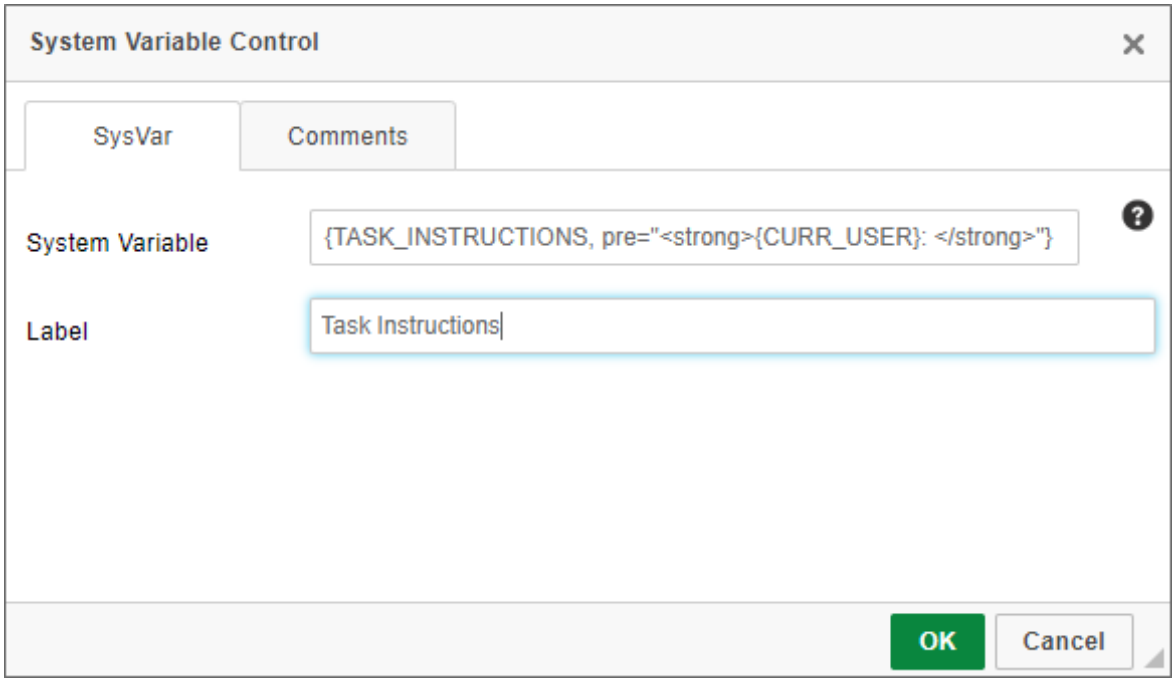
Forms

1. Use meaningful instance names for each form definition, by setting the Form definition's Instantiated Form Name property. By the same token, use simple, meaningful names for object and field definitions. Do not use Hungarian notation or other programming-style conventions. Names should be logical and recognizable. This is very helpful when the objects are used in other places, such as when defining conditions. Object names will appear sorted alphabetically.

2. Think carefully about when the Form's data fields should be editable by users. There are two properties on the Properties tab of the Form definition that control this: Form Fields are Enabled Only When and Entire Form is read-only when. In general, Form fields should be enabled for new form instances, but disabled in all other contexts. If form information does need to be edited, specify the field that should be edited during the process, and when editing is to be allowed, via visibility conditions on the appropriate fields. Section controls are extremely helpful here, because you can place the editable controls in their own Section, and enable or disable the Section control to enable/disable its constituent controls. Similarly, once the process is complete, the entire Form should be set to read-only, so that it cannot be edited after the process has completed.

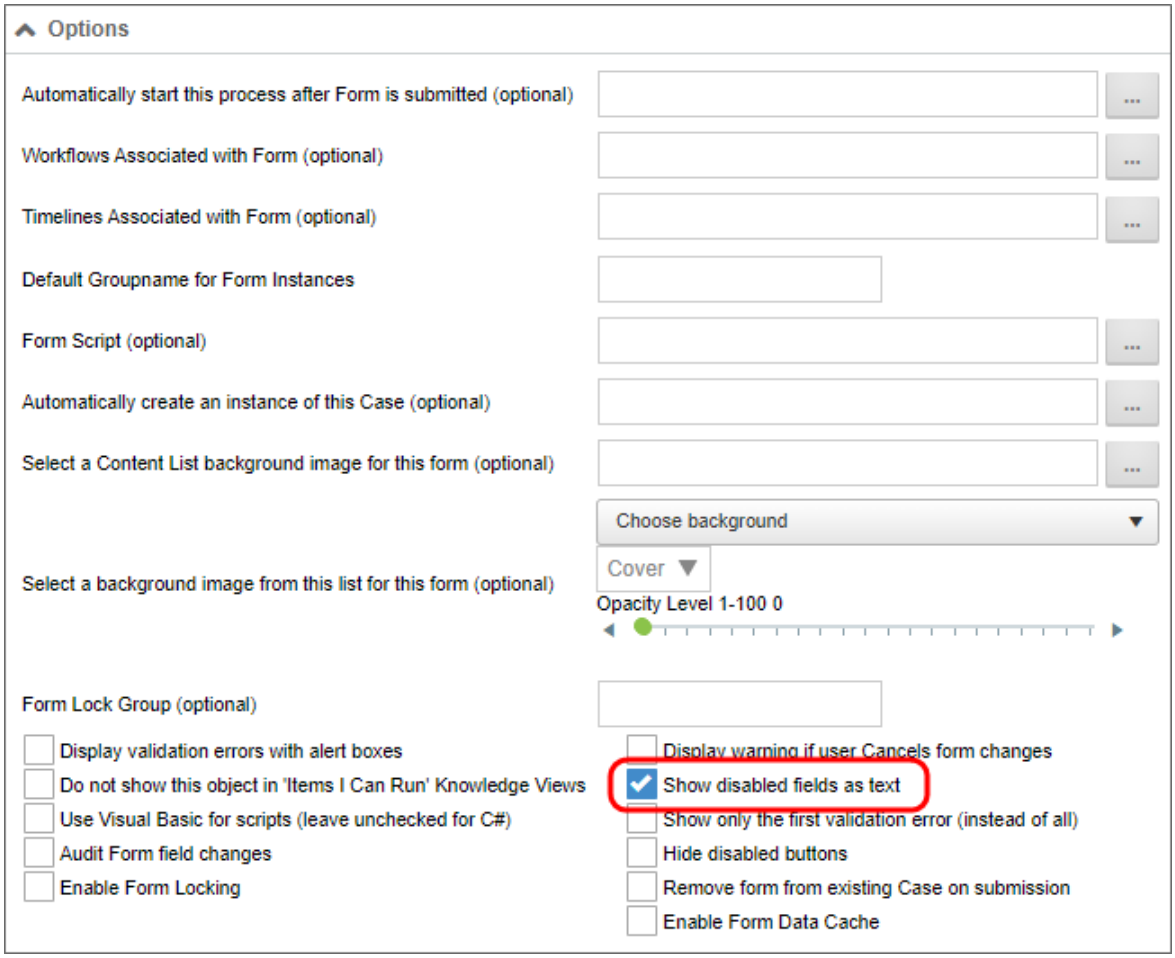
3. When using tables to provide Form layout, be sure to set the Role attribute, as described here.
4. Instructions should tell users to do something. For example, “Please review this form and approve or reject it using the buttons below”. Note that it’s nice to begin instructions with “Please”. Descriptions should explain something. Always use friendly names for form fields that are required, have validation rules associated with them, or may be used in Knowledge Views, so that Knowledge Views and other derived uses of the field display human-friendly field names. Keep in mind that Knowledge View column titles will use the friendly name, so brevity is important, as a friendly name like "This is the name of the user who filled out this form" might not be the best choice.
5. Always add the assigned user and task instructions at the top of forms and email templates. In a task context; the easiest way to be sure of that is to use the System Variable control, and set it to use the Task Instructions System Variable, which only appears in that context. You can append the task user's name to the control by typing "`{curr_user}:` " in the "Pre" formatter of

the variable's attributes, as shown below.



6. As a guideline, on each Form template, there should be three standard fields: the form submitter, submission date, and a unique request number. The three fields should be disabled for all users. In some cases, the submitter may be submitting on behalf of another user: those fields should be separate, so that the submitter is always automatically set (usually by setting its default value to Form Submitter), and the on-behalf-of user can be selected by the submitter using a user picker or other mechanism. The unique request number can be set via the use of the Sequence Number System Variable. It's best to set this value in the very first activity of the Process Timeline, so the Sequence Number is generated only after the Form has been submitted.
7. When possible, choose the Show disabled fields as text option in the Form definition, which will render disabled fields as plain text, rather than disabled

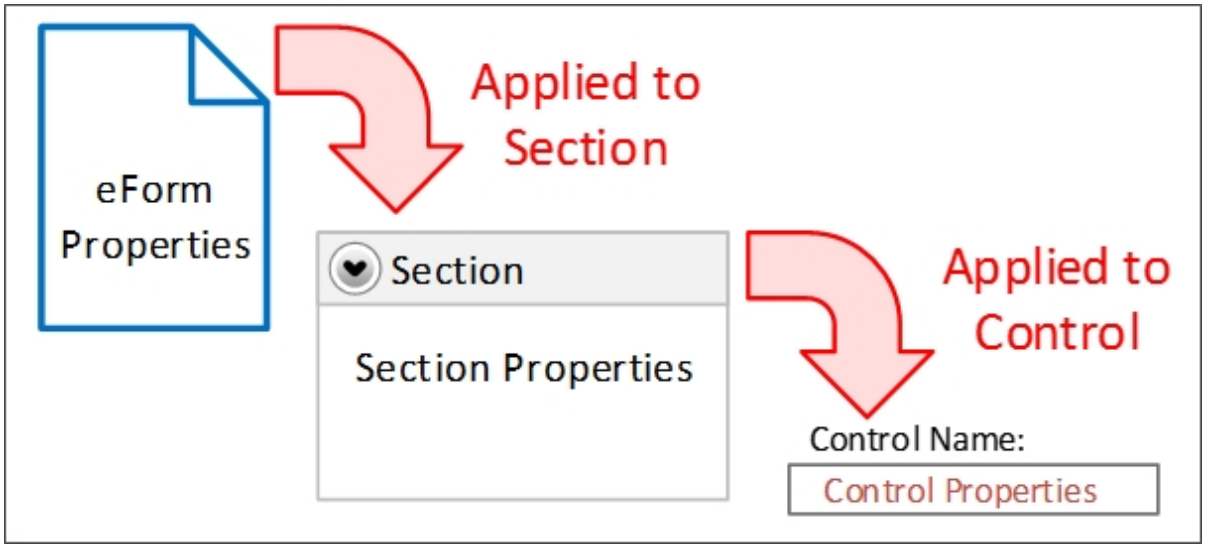
controls.



8. Form control appearance logic can be complicated. The following rules should apply:

- Think about when form control appearance logic should be implemented within a Business Rule, rather than rather than explicitly defined within the field properties. As a rule of thumb, consider a Business Rule when there are three or more conditions controlling the field's appearance, or when the same appearance logic is used on multiple fields. This is true of task assignment in general as well. If you put the assigned user in a Business Rule, you can more easily find and change the user later, as users often change in the course of regular promotions and attrition. Don’t name the Business Rule after the person, but rather, the role the person plays in that Process.
- Do not tie a specific user to the appearance logic for a field unless absolutely necessary.

- Avoid using the "Step Running" condition for appearance logic, because the fields you desire to show or hide will only be shown or hidden while that specific activity is running. In all probability, the actual condition you wish to use is "Step Reached", because you wish the fields to be available when the process reaches a specific point, and at all points in the process thereafter. It's very common to have form control logic (enabled/disabled, visible/hidden, required/-optional) driven by the state of the process, and the context in which the user is viewing that form (i.e., whether the user is in a task or not). As a general guideline, use "Activity Reached" when you want appearance based on whether or not the process has reached a certain point, and use "Task Name =" when you want appearance based on whether the user is interacting with the form in the context of a specific task assigned to that user.
- Appearance logic is strongly driven by inheritance. Inheritance is the ability of a container control to automatically incorporate, or inherit, properties of its parent container control. In the case of Form controls, for example, the "required" and "enabled" properties are inherited. The Form is the highest-level container control, and every control on the form is a child control of the Form. A Section control placed onto a Form will inherit the properties of the Form, while an individual control placed inside of a Section control will, in turn, inherit the section's properties.



For example, if the Required property of a Section control is set to "true", then inheritance ensures that all of the controls contained in that section will be

required. You can override inheritance, however, by explicitly setting the Required property of a control inside the section to "False". If there is conflict between the Required property of a container control, and the controls contained inside it, the control property at the lowest level wins, and overrides the inheritance. With the "lowest level wins" model of inheritance in mind:

- Disable forms when the form isn't a new instance, then enable individual sections as necessary for data input in succeeding activities.
- Do not use the Otherwise disabled or Otherwise enabled conditional appearance settings in field properties unless you specifically want to override the appearance logic for the parent container in all cases.
  - Setting a control to "Otherwise Xxxxx" when the parent container has its own appearance logic can result in unexpected behavior when the parent container's appearance logic conflicts with that of the control.

9. Label control values don't get saved to the database; therefore, labels should be used only to show text on forms. Label controls are display vehicles only; don’t try to use them in conditions, for example. When possible, associate each label with an Input control for Accessibility compliance.
10. Hide debug sections and other development conventions in forms from the non-developer form users. Users should never see anything on a form that isn't relevant to the process on which they are working.
11. The Routing Slip should appear below any user action buttons so that users don't have to scroll below the Routing Slip to perform their assigned task.
12. Use the branch/result order properties to ensure that options are presented consistently to users throughout a process. approvals are always displayed first in the button areas.
13. Sequence numbers should always be formatted as text, and should always use the "digits" formatter. There are a couple of reasons for this.

- First, there's a general rule that, if you aren't going to do math on a data item, it's not actually a number, even if all of the characters are numeric. This is why zip codes, or social security numbers are always formatted as text. Additionally, if a data item is formatted as a number, then you can't use the "Contains" operator in the Condition Builder, and can only use the "=" operator. This limitation can be troublesome, because if you leave the value blank in, say, a Knowledge

View filter condition that uses the "=" operator, Process Director returns no results, but leaving a "contains" condition blank returns all results. Most of the time, you want to return all results as a default, and then allow the user to filter the results to a smaller set by entering a value for the sequence number, which requires using the "contains" operator in the filter.

- Formatting the sequence number as text, of course, changes the way sorting works. A numeric field is sorted in numerical order, while a text field is sorted alphabetically, so the same set of sequence numbers will be sorted in different order, depending on whether they are text or numeric data.

NUMERIC SORTING	ALPHABETIC SORTING
1	1
2	100
11	11
21	2
100	21

- This is where the "digits" formatter for the sequence number comes in. If you set the digits formatter to "digits=4", the sequence number will always be a fixed length of four digits, and will use leading zeros for smaller numbers, so that sequence number "1" will be stored as "0001". Adding the leading zeros will fix the Alphabetic sorting issue, so that you can sort by the sequence number in the expected order, e.g., 0001, 0002, 0011, 0021, 0100.

Knowledge Views

In general, you should always return Form Instances with Knowledge Views unless you have a specific reason to return other objects such as Attachments, Timeline Instances, etc. This is especially true if you are creating filters that use the value of some form field for filtering. Returning, say, Timeline Instances will cause your filter criterion to fail, whereas form field filters will work as expected when returning Form Instances.

Business Rules, Business Values, and Goals

1. For the purposes of this discussion, we'll refer to Business Rules, Business Values, and Goals collectively as "value objects", since they primarily return a value or values. These value objects occupy a slightly different space than other Process Director objects. Most objects, like Forms or Timelines, are tied to a specific implementation project. Conversely, value objects are often not tied to a specific project at all. For instance, a Business Value that extracts some data from an external ERP or CRM system may be widely used across a number of projects. Similarly, a Goal that specifies some universal system state may be used by all projects. On the other hand, a Business Rule might return a global value that is not tied to a specific application, while a different Business Rule might not work outside the context of a specific Form or Process Timeline definition. This configuration issue raises the question of where the value objects should be placed in the Content List, and when—or if-—they should be exported/imported between development and production systems.
2. You may wish to consider whether to create a folder at the root level of the Partition to store value objects that are used by multiple projects, while storing project-centric objects within the relevant project folders. It's often quite easy to determine whether an object is project-centric or not: If the object can be used by any application, such as a Business Value, Dropdown Object, or DataSource object, it should probably be stored in a central location in the Content List, separate from any specific implementation project. BP Logix uses folder names set off with square brackets, e.g., [Business Values] to organize these global objects together in the Content List.
3. Indeed, you might wish to consider this method of centralizing storage for some other objects, as well. For instance, some Business Rules may be used in multiple projects, because they have no dependency on specific forms or other application objects. These Business Rules should also be stored in a central location, while application-specific Business Rules would be stored within the application's parent folder.

Changing Existing Processes

You'll occasionally need to alter an existing Form or process. Perhaps more data must be collected from the Form, or the process itself must change. Since the process exists, and probably has some in-flight processes already active, you should understand how changing a process or Form definition will affect existing instances.

In general, all process changes are implemented immediately, as soon as the edited process or Form is saved or imported. The change can affect running process instances in various ways. Now, for the most part, changes to a process tend, in actual practice, to be fairly minor. So, in most cases, the change will require no administrative intervention. Process director will apply the change, and any existing processes will simply use the new definition to complete the in-flight process, while retaining the data from tasks that are already completed.

In running processes, the Routing Slip will reflect the tasks that were completed in the old process, as they existed before the change, then subsequently mirror the new process after the point of change. Additionally, Process Director will check the running instances of the process to see if there are any notifications that need to be resent, and, of course, resend the notifications where appropriate. In most cases, therefore, changes to the process will be seamless.

There may be some exceptions, however:

- When you attempt to delete an activity from the Process Timeline definition, if there are any active processes that are currently running that activity, process Director will remove the activity from the dependence tree in the process Timeline definition, but will not delete the activity until all currently running instances of it are complete. Once the users complete the task, Process Director will, in most cases, resume with the next task that was previously scheduled. However, if you delete multiple timeline activities, Process Director may not be able to determine what activity should run next. In that case, the Timeline Instance will be placed in an error state, and will require administrative intervention. *Note that for versions of Process Director prior to v5.44, the system will allow you to delete Timeline Activities, but will not try to determine which activity should be invoked next, and place the timeline into an error state immediately.*

- If you have selected the option to ensure that at least one user must complete the task, Process Director will not attempt to determine what task should be accomplished next. Instead, it will place the process instance in an error state.
- When you add new activities to an early stage of the process, any existing instances that have passed the new activities won't return to complete them. The new activity won't run in that instance without some intervention, either administrative, or built into the process, such as by implementing a rollback for instances where the activity results have not been stored.
- When you add a field to a Form, all new instances will store the data properly, but existing instances won't contain any value for the new field. The value will, therefore be "null". In an existing process instance, if that field is used to make a determination of what process activity to start, the system won't evaluate the criteria as expected, which can lead to unpredictable results. For instance, Let's say you add a new Checkbox field to a form called NewField, and, in your process, you add a 'Needed When' condition to a Timeline Activity like "NewField is false". Your assumption might be that, since NewField is a check box and hasn't been marked "true", it must logically be false. But, actually, the field has a null value, because it didn't exist when the form was submitted, and is therefore, neither true nor false. In that case, the field will, when evaluated, fail the condition.
- When you delete a form control from a Form, existing Form instances will retain that data. The Form field will still exist in the Process Director database, even though the control has been removed from the Form's design surface. The data in that field will be retained until you go to the Form Controls tab of the Form definition and select the Delete all controls marked for deletion action link, at which point the data will be permanantly deleted from the Process Director database. Always remember, however, that deleting a form control from the Form template will eliminate the control from the Form's display in ***all*** Form instances. The data will exist, but the form control won't be displayed, because Forms always display the current version of the Form template. Also, any form conditions that used the deleted field will no longer work once the control is deleted from the form's design surface. Please see the Deleting Controls and Form Fields topic for more information on how to handle Form data from deleted controls by deleting or remapping it.

- When you add a new step, or steps, to an early stage of the process, any existing instances that have passed the new steps will not return to complete them. The instance will enter an error state, but the new task will not be performed in that instance without some intervention, either administrative, or built into the process, such as by implementing a rollback for instances where the step results have not been stored.

For users of process Director v4.06-5.44, when an Activity is deleted, any process that is running in that Activity will now be placed into an error state. It will remain in an error state until an administrative action is taken, e.g. jump to an activity, start an activity, rollback to a previous activity, etc. Any processes that went into an error state will appear in any Knowledge View that shows processes in error, and users will see a message indicating the reason for going into the error state.

For users of process Director v4.31 or higher, when a new Activity is added to a Timeline Definition, the new Activity will be marked as "Not Required" in any running Timelines if all of its dependencies have already completed.

For users of process Director v4.56 or higher, when a parent Activity is added above existing activities, and there are activities running in existing Timeline instances under that new parent, the system will now automatically start the new parent Activity.

Though Process Director will try to make the Process seamless, in many cases process administrators will still have to intervene in the running processes. To make the intervention easier, you should create a Knowledge View that returns only processes that are in an error state. This will provided you with a list of all the processes that require intervention, making it easier to clear the errors. Similarly, you can create a Knowledge View that displays processes where new activities have been introduced, but which have not been completed in existing instances. In both cases, the Knowledge View feature is your friend.

In general, you shouldn't attempt to create a new Process Timeline when you change a process, unless the change is so fundamental that it constitutes a new process. If the change is that fundamental, then you should recreate the entire application. Otherwise, you should always edit the existing process. Doing so retains the accuracy of your auditing and historical data, whereas creating an entirely new application will have its own, independent data. Depending on your regulatory requirements, you'll still need to retain the old process intact, so that it can be audited, which means keeping an inactive process in your production environment until the regulatory requirement expires.

Best Practices for Accessibility

The creation of the internet introduced a new era by creating a computing environment that works for everyone. Instead of requiring a specific computer platform, internet applications can be accessed by anyone, from anywhere, in any language. The Internet-based environment removed most of the old technical and physical barriers to communication and interaction.

When developers create badly designed web-based applications, however, they may re-introduce some of those old barriers to interaction for some users. Accessibility standards were created to ensure that the Web's promise of accessibility for all remained true.

Why Accessibility is Important

As the World Wide Web Consortium (W3C) notes, web accessibility refers to technologies that enable people with disabilities to use web-based applications. These disabilities may include:

- auditory
- cognitive
- neurological
- physical
- speech
- visual

Beyond that, however, web accessibility also benefits people without disabilities:

- The proliferation of mobile devices, which many people now use instead of traditional desktop or laptop devices, means that many people now view the web on much smaller screens, without traditional input devices like the keyboard or mouse.
- As people age, they may start having some level of natural physical impairment. The increasing number of older people means more and more people will require increased accessibility efforts on the part of business, educational, and government institutions to meet the needs of this population.

- Sometimes, people just get injured, and it may temporarily be hard to use a computer until they are healed, for instance, while wearing a cast on one's hand.
- People are often outdoors in bright sun, or in public where playing audio would bother others. While temporary, such situations can be inconvenient.
- People may be in a location with limited internet connectivity, and a more accessible experience can minimize the use of limited Internet bandwidth.

Web accessibility, therefore, benefits everyone, not just the disabled. This is especially true as the Internet

- Has become the nexus for nearly all aspects of information use in daily life, whether it's in education, commerce, government, or social interaction, and
- Has replaced most of the accessibility barriers to print, audio, and video.

The W3C also makes a good business case for using accessible design.

While it is often difficult to measure the return on investment for many information technology initiatives, there have been some interesting research findings in the area of accessibility implementation. For instance, a research study of Fortune 100 companies indicates that high-performing organizations commonly include disability inclusion as part of their overall strategy.

Implementing accessibility provides many benefits, such as reducing legal risks, improving the customer experience, and increasing worker productivity. Integrating accessibility also removes barriers to innovation, and provides varied and flexible ways for users to interact with websites and applications. Design of user interaction considers experiences other than screens when accessibility is a consideration. The result is interaction that is more human-centered, natural, and contextual. Thus, accessibility is closely related to general usability – both aim to define and deliver a more user-friendly experience.

A clear commitment to accessibility can demonstrate that a business has a genuine commitment to Corporate Social Responsibility. The benefits of this commitment include enhanced brand image and reputation, and improved workforce diversity and many other benefits. For instance, employing people with disabilities is an essential aspect of creating a diverse workforce. To be successful, the technology that employees use, including websites and applications, must be accessible.

Additionally, a Forrester Research study concluded that accessibility could contribute to cost savings when it is integrated into existing and ongoing development cycles. Technology updates and redesigns that include accessibility along with other best practices have demonstrated reduced costs for maintenance and service. Moreover, as accessibility features are included, overall customer satisfaction improves.

It's Not Just a Good Idea, It's the Law

When using Process Director, you should make end user accessibility a priority. Not only is it a nice thing to do in general, but most governments mandate some form of accessibility be implemented so that disabled persons can access web-based content more easily. In the US, this mandate comes from Section 508 of the Americans with Disabilities Act, while, more recently, Canada passed the Accessible Canada Act, which, along with various provincial accessibility laws, now mandates accessibility for all federal public institutions, Crown Corporations, and all federally regulated organizations. Fortunately, most accessibility laws specify conformance with the Web Content Accessibility Guidelines (WCAG) promulgated by the World Wide Web Consortium (W3C).

Over time, the legal risk of ignoring accessibility has increased with the adoption of more specific laws and government policies.

In the United States, for example, the number of legal actions continues to rise, and courts increasingly decide in favor of equal access, often citing the Americans with Disabilities Act. Between government oversight and regulation on the one hand, and increased legal action on the other, the legal landscape is rapidly changing in favor of equal access.

Similarly, in Canada, litigation for accessibility is commonly pursued under the Canadian Charter of Rights and Freedoms, the Canadian Human Rights Act, the Employment Equity Act, and, most recently, the Accessible Canada Act. Meanwhile, several provinces, such as Ontario, Manitoba, and Nova Scotia, have fairly extensive provincial laws that mandate accessibility standards.

With legal risks increasing, smart organizations should create and implement accessibility policies and programs to mitigate legal risk to protect both their assets and their reputations.

Public use of the web is more than 25 years old. It is no longer a novelty, but an integrated and critical tool of modern life. Smart organizations integrate accessible design into their processes, because they understand the need for equal access by all people. The legal risks of ignoring accessibility are significant, while the benefits of implementing accessibility have been demonstrated by leading organizations. By developing a long-term commitment to accessibility your organization is likely to thrive in unexpected and self-sustaining ways.

Accessibility in Process Director

In this section of the documentation, we'll l examine the WCAG accessibility standards, and discuss some guidance for implementing these standards within Process Director. Process Director provides accessibility tools for the creation of objects that will be accessed by end users—which primarily means Forms—and not for the administrative functions of the product itself. In order to organize these various practices, we will take a look at each of the WCAG standards, and describe the built in accessibility features of Process Director, along with some best practices for implementing each standard.

More detailed information about Process Director's compliance with accessibility standards is provided by the BP Logix Process Director Accessibility Conformance Report (VPAT© Version 2.3) as a PDF download.

Many accessibility features are built in to the product already, and govern specific control behaviors. These control behaviors can be reviewed in this downloadable PDF document.

Checking for Accessibility

There are a number of tools you can use to check for accessibility while designing your Process Director forms, but one of the easiest methods for running accessibility checks is to use a browser extension (or "add-on" for Microsoft browsers). Browser extensions are installed into the web browser, and run inside the browser without needing any other external software, so they are fairly easy to use.

Keep a couple of things in mind:

- *We can't specifically recommend* any third-party tool, or browser extension, only inform you about *some* of the available options.
- None of these browser extensions are perfect, and all of them have their own strengths and weaknesses.
- You should choose the accessibility checking tool that best meets your needs, after an appropriate evaluation.

With the above in mind, here are some browser extensions that are used fairly widely.

Lighthouse: This isn't actually an extension, but one of the Developer Tools that are already built into Chrome and Chrome-based browsers like Brave. It can perform a variety of checks, one of which is an accessibility check. In Chrome-based browsers, the Developer Tools can be accessed by pressing the [F12] key. Lighthouse has its own tab in the Developer Tools UI.

WAVE Accessibility Extension: Available for both Chrome and Firefox. This is one of the oldest and most widely-used browser extensions.

axe - Web Accessibility Testing: Available for Chrome, Firefox and Edge.

Accessibility Insights for Web: Available for both Chrome and Edge.

a11yTools - Web Accessibility: Available for Safari. Safari has a more limited choice of extensions, unfortunately.

IBM Equal Access Accessibility Checker: Available for Chrome and Firefox.

As mentioned previously, each of these tools have pros and cons to their use, and some experts recommend using more than one tool to check for accessibility issues.

Accessibility Principles

Please continue to see the **Accessibility Principles** that apply to no code/low code application development with Process Director.

Accessibility Principles

The W3C has specified three levels of accessibility in the WCAG Standards: Levels A, AA, and AAA. **Most legal requirements specify compliance with Level AA of the WCAG guidelines.** Since that is so, this document won't attempt to provide a listing of all Level AAA requirements. Also, keep in mind that some of the Level AA

requirements aren't especially relevant to Process Director either, so we will concentrate on the Level AA requirements that are most relevant. Finally, remember that some requirements simply aren't related to the controls or other features of Process Director, but must be implemented manually, such as giving logical names to form controls, or to the link text you use for hyperlinks. Process Director will automatically implement accessibility standards where it can, but much of the burden remains upon you, as a form designer, to implement accessibility standards as part of your design.

So, with the above in mind, let's take a look at the WCAG standards, and see if we can't provide some helpful guidance.

The WCAG standards are defined by four principles, with guidelines for each principle that specifies how it should be implemented in a web application. The guidelines for each principle are linked below.

Principle 1 - Perceivable: This principle relates to making information and user interface components presentable to users in ways they can perceive.

Principle 2 - Operable: This principle relates to how user interface components and navigation operate.

Principle 3 - Understandable: This principle relates to making information and the operation of the user interface understandable to users.

Principle 4 - Robust: This principle relates to making content robust enough to be interpreted by a wide variety of user agents, including assistive technologies.

Accessibility Principle 1 – Perceivable

This principle relates to making information and user interface components presentable to users in ways they can perceive.

Guideline 1.1 – Text Alternatives

Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language.

1.1.1 Non-text Content

Level A - All non-text content that is presented to the user has a text alternative that serves the equivalent purpose.

In the context of a Process Director form or Dashboard, this requirement is primarily applicable to the use of images, including logos and other branding images.

Image controls in Process Director's Online Form Designer contain an Alternative Text property. This property should be used to provide descriptive text for all images you display on forms.

Similarly, Button controls have an Alt Text property that you should use to provide descriptive text to buttons that only present images or icons.

All data entry controls should have a Label control associated with them. In the properties dialog of the Label control, the Associated Control ID property should contain the Name of the control to which the Label is associated. For instance, if you have an Input control with the Name "FirstName", the Associated Control ID of the Label would be "FirstName". Label controls also have an Alt Tag property to provide the required text alternative to the Text property of the Label. When using tables, the Label control should be placed in the same table cell as the data entry control to which it is associated, to make it easier for assistive devices to understand the relationship between the two controls on the page.

Tables have a Caption property to provide text-based table descriptions in the Table Properties dialog box. Additionally, a Summary property provides a text representation of the table's purpose or brief description of the data it contains.

Guideline 1.2 – Time-based Media

Provide alternatives for time-based media.

This isn't generally relevant to Process Director, as forms don't display video or audio presentations.

Guideline 1.3 – Adaptable

Create content that can be presented in different ways (for example simpler layout) without losing information or structure.

1.3.1 Info and Relationships

Level A - Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text.

You should use tables to present tabular data, and use the appropriate table headers. When doing so, Process Director's Online Form Designer will automatically insert the appropriate table, th, tr, td, etc., tags. You can specify the headers as the first row, first column, or both, using the Headers property of the Table's prop-

erties Dialog. Additionally, the Cell Properties dialog box has a Cell Type property you can use to specify if a table is a data or header cell.

Process Director automatically provides the HTML "scope" attribute for headers to associate header cells and data cells in data tables.

1.3.2 Meaningful Sequence

Level A - When the sequence in which content is presented affects its meaning, a correct reading sequence can be programmatically determined.

This isn't generally relevant to Process Director.

1.3.3 Sensory Characteristics

Level A - Instructions provided for understanding and operating content don't rely solely on sensory characteristics of components such as shape, color, size, visual location, orientation, or sound.

You should always provide textual identification of items that otherwise rely only on sensory information to be understood. In other words, image-only buttons, images used as buttons, or other UI conventions should always provide a textual identification in addition to the sensory-based convention, e.g., using explicit text labels for buttons, in addition to images or icons, where possible. There are exceptions when space constraints may not allow the use of such explicit text, in which case you must use the Alt Text property of the object to ensure a text alternative is always available for it.

1.3.4 Orientation

Level AA - Content doesn't restrict its view and operation to a single display orientation, such as portrait or landscape, unless a specific display orientation is essential.

As a best practice, you shouldn't fix widths for layout elements if doing so isn't essential. By default, for example, the default width of tables in the Online Form Designer is 100%, which enables them to scale to the screen width of the viewport. You shouldn't impose a specific screen size or viewport on your users, or attempt to replicate the exact layout of a printed form. Users who require a larger font, for example, may find such forms more difficult to read or use because of the layout restrictions you've imposed.

1.3.5 Identify Input Purpose

Level AA - The purpose of each input field collecting information about the user can be programmatically determined

As a best practice, all input fields should have logical, recognizable Name properties that reflect the type of data being collected by the input control, rather than use among conventions that make sense only to developers or other IT personnel. For instance, "FirstName" tells you fairly specifically what the purpose of the input control is, while "frm11a_txt_fname" is substantially more opaque.

Guideline 1.4 – Distinguishable

Make it easier for users to see and hear content including separating foreground from background.

1.4.1 Use of Color

Level A - Color isn't used as the only visual means of conveying information, indicating an action, prompting a response, or distinguishing a visual element.

As a best practice, any information conveyed by color differences should also be available in text. Use of color cues alone to display a status, or prompt a response isn't an acceptable accessibility practice. Color should be an enhancement to, and not the primary method of, communication.

1.4.2 Audio Control

Level A - If any audio on a Web page plays automatically for more than 3 seconds, either a mechanism is available to pause or stop the audio, or a mechanism is available to control audio volume independently from the overall system volume level.

This is generally not relevant to Process Director.

1.4.3 Contrast (Minimum)

Level AA - The visual presentation of text and images of text has a contrast ratio of at least 4.5:1, except the following:

Large Text: Large-scale text and images of large-scale text have a contrast ratio of at least 3:1;

Incidental: Text or images of text that are part of an inactive user interface component, that are pure decoration, that aren't visible to anyone, or that are part of a picture that contains significant other visual content, have no contrast requirement.

Logotypes: Text that is part of a logo or brand name has no contrast requirement.

As a best practice, text should be easily distinguishable from any background colors or images through the use of highly contrasting colors. Without sufficient contrast such text will be difficult for to read by users who are visually impaired. The WCAG specification links to a number of web-based tools that are available online for analyzing color contrasts to ensure that your design meets the required contrast ratios. In addition, we've provided a set of sample colors you might want to consider for creating the appropriate contrasts with white or black text, for use with UI elements like buttons.

1.4.4 Resize text

Level AA - Except for captions and images of text, text can be resized without assistive technology up to 200 percent without loss of content or functionality.

Most modern web browsers have 200%+ text resizing built in. As a form designer, you should remember that the text size in which you are designing the form may not be the size in which your users are viewing it. As a best practice, you should view your form at different text sizes, up to 200%, to ensure that the layout and usability of your form remains intact at high zoom levels. This also relates to the best practice identified for WCAG specification 1.3.4, above, as this requirement affect the proposed layout of your form at high magnifications.

1.4.5 Images of Text

Level AA - If the technologies being used can achieve the visual presentation, text is used to convey information rather than images of text except for the following:

Customizable: The image of text can be visually customized to the user's requirements;

Essential: A particular presentation of text is essential to the information being conveyed.

Note 1: Logotypes (text that is part of a logo or brand name) are considered essential.

As a best practice, use text, rather than images of text, wherever possible.

1.4.10 Reflow

Level AA - Content can be presented without loss of information or functionality, and without requiring scrolling in two dimensions.

Though there are some minor exceptions to this requirement in the WCAG 1.4.10 standard in general, you should ensure that your users don't have to scroll in more than one dimension, i.e., vertical scrolling alone is acceptable, as is horizontal scrolling alone, but requiring users to scroll both vertically and horizontally isn't acceptable. (As a best practice, horizontal scrolling should be eliminated entirely if it is possible to do so.)

When using installations with the mobile device component, Process Director will automatically reflow content, such as tables, to ensure it fits within smaller viewports without having to scroll horizontally. Otherwise, some relatively simple CSS styling applied to the form in the Properties tab of the Form Definition can accomplish the same refactoring to control the reflow of a table. A sample of this sort of reflow CSS is provided below.

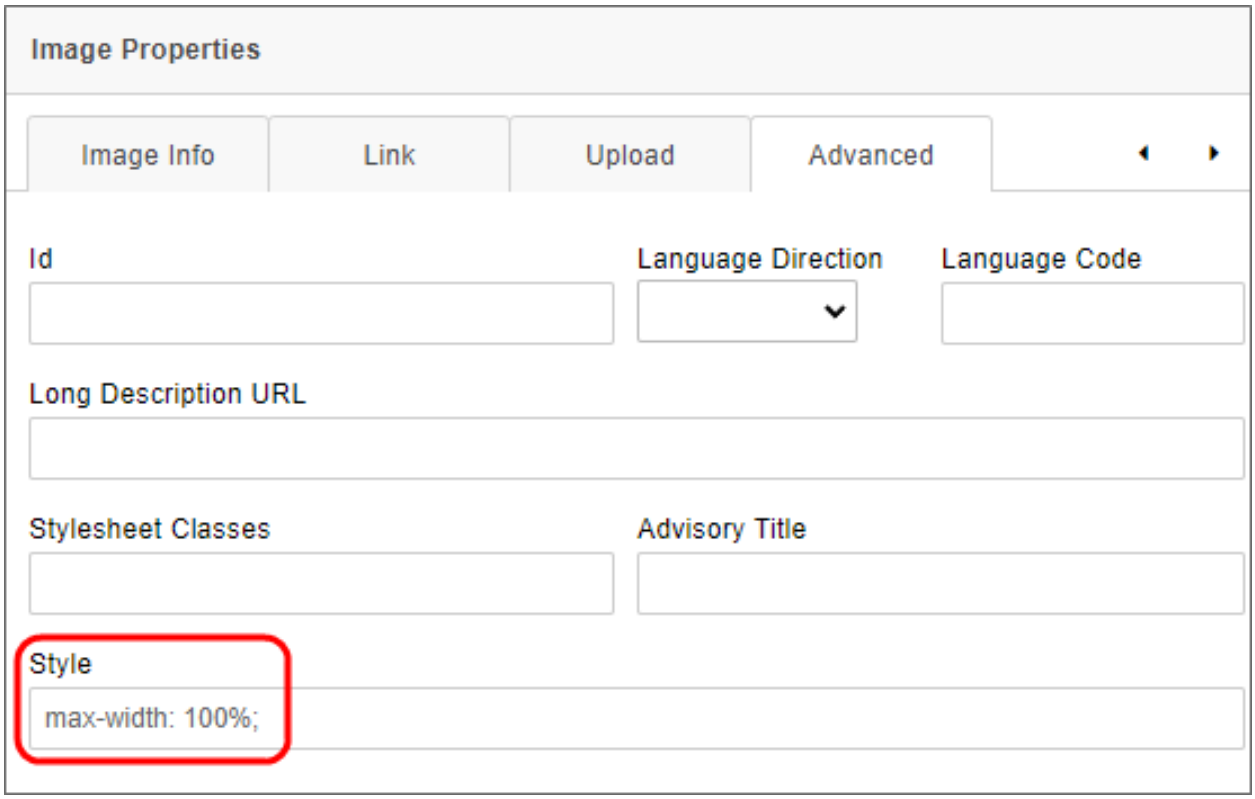
Code Sample

```
@media
only screen and (max-width: 860px),
(min-device-width: 860px) and (max-device-width: 1024px) {
    /* Force table to not be like tables anymore */
    table, thead, tbody, th, td, tr {
       display: block;
    }
```

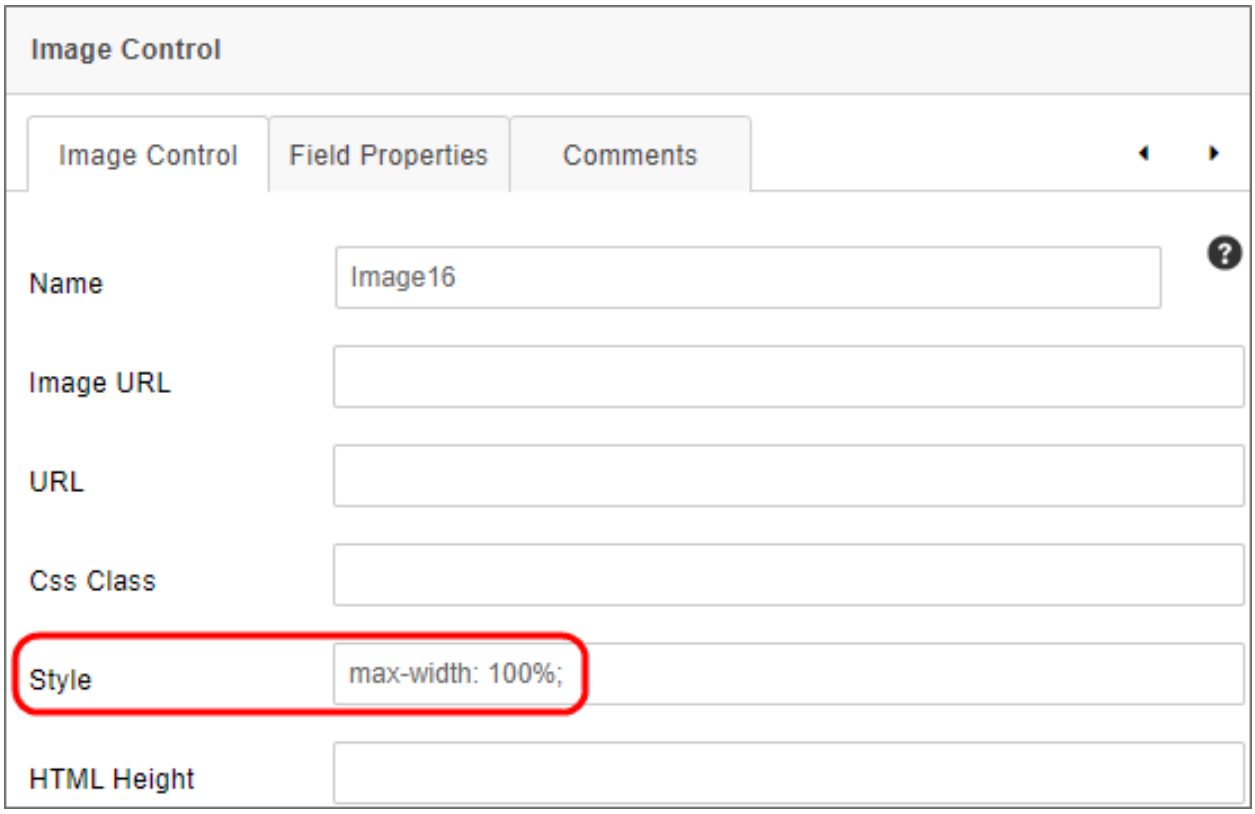
```
    /* Hide table headers (but not display: none;, for access-
ibility) */
    thead tr {
        position: absolute;
        top: -9999px;
        left: -9999px;
    }
    tr {
        border-top: 1px solid #eeeeee;
        margin-top:10px;
    }
}
```

For many UI elements, such as tables, or controls like the Section control, Process Director sets the default Width of the element to 100% of screen width automatically. In most use cases, this ensures that the element fills the screen horizontally, but doesn't extend beyond the screen's width. Images, however, can be particularly troublesome in this regard, since an image that displays well on a desktop screen may be too wide to display properly on a table or phone. These large images will push the effective screen width beyond the horizontal size of the screen, which will also force the vertical size of other elements off the edge of the screen. You may wish to set the CSS style for images (i.e., the `img` element, to a `max-width` of *100%* to ensure that, as screen widths change, the image never extends beyond the screen's width. This CSS style forces the width of the element to always be no larger than 100% of screen width, and enforces this setting by forcing the image to resize to the screen width when the screen is resized. This style can easily be applied in the Image control's properties by adding this style setting to the image in the Style property.

For the standard Image control, the Style property can be found on the Advanced tab of the Properties dialog box for the control.



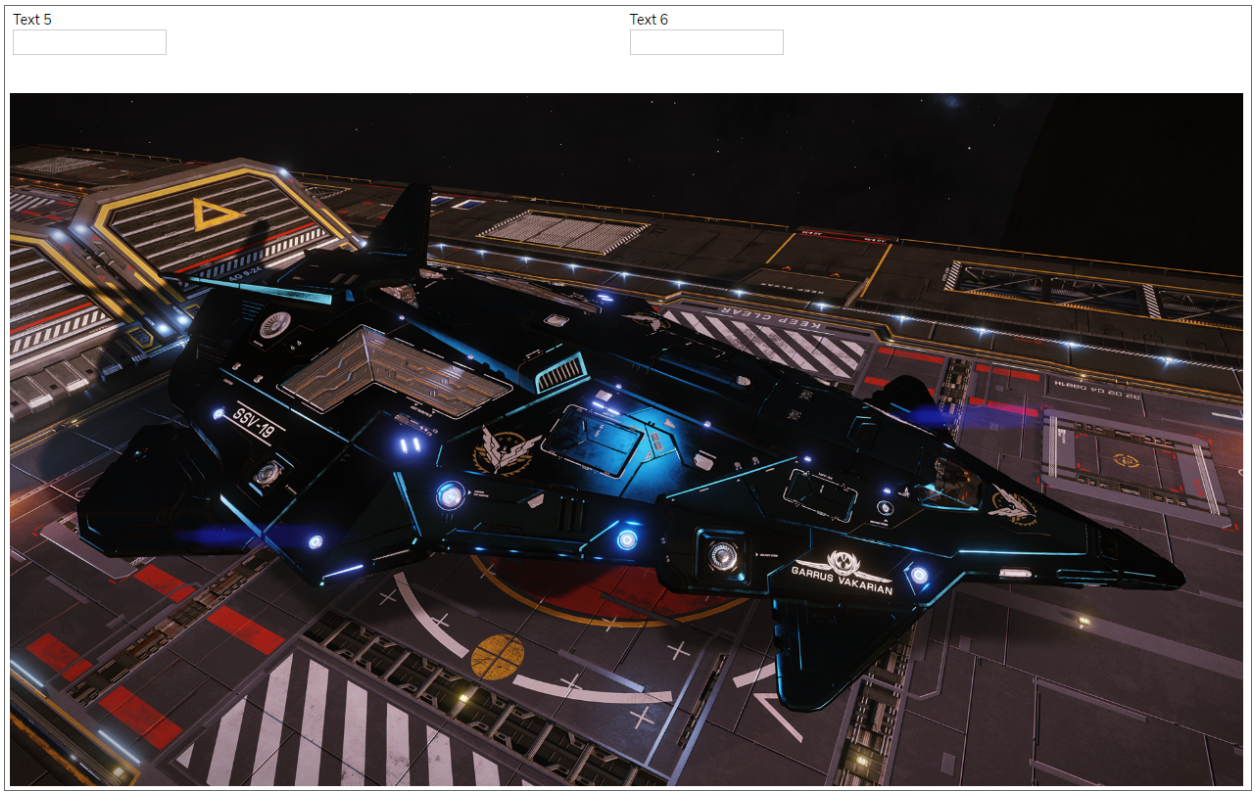
For the BP Logix-specific Image control that's accessed via the Other Controls menu, the Style property is displayed on the Image Control tab of it's Properties dialog box.



Once set, the image will always display properly, as shown in the examples below.

Image Example

Wide Screens:



Narrow Screens:

1.4.11 Non-text Contrast

Level AA - The visual presentation of the following have a contrast ratio of at least 3:1 against adjacent color(s):

User Interface Components: Visual information required to identify user interface components and states, except for inactive components or where the appearance of the component is determined by the user agent and not modified by the author;

Graphical Objects: Parts of graphics required to understand the content, except when a particular presentation of graphics is essential to the information being conveyed.

As a best practice, and as with the text contrast requirements of 1.4.3, above, user interface components and other graphical object must have a relatively high

contrast. For example, a button with blue text and a slightly darker blue background might look nice, but will probably not have enough contrast between the button's background color and text to meet the requirements of this specification. In Process Director, button background, text, and icon colors can all be specified b the designer, so ensure that the color contrasts are large enough for visually impaired users to easily differentiate between them. Again refer to the contrast analyzer tools linked in 1.4.3, above.

1.4.12 Text Spacing

Level AA - In content implemented using markup languages that support the following text style properties, no loss of content or functionality occurs by setting all of the following and by changing no other style property:

Line height (line spacing) to at least 1.5 times the font size;

Spacing following paragraphs to at least 2 times the font size;

Letter spacing (tracking) to at least 0.12 times the font size;

Word spacing to at least 0.16 times the font size.

Exception: Human languages and scripts that don't make use of one or more of these text style properties in written text can conform using only the properties that exist for that combination of language and script.

Though this is generally not relevant to Process Director, you should, as a best practice, not make the organization or presentation of information dependent on line, letter, or paragraph spacing. This is also a layout consideration that affects the requirements of both 1.3.4, and 1.4.4, above.

1.4.13 Content on Hover or Focus

Level AA - Where receiving and then removing pointer hover or keyboard focus triggers additional content to become visible and then hidden, the following are true:

Dismissible: A mechanism is available to dismiss the additional content without moving pointer hover or keyboard focus, unless the additional content communicates an input error or doesn't obscure or replace other content;

Hoverable: If pointer hover can trigger the additional content, then the pointer can be moved over the additional content without the additional content disappearing;

Persistent: The additional content remains visible until the hover or focus trigger is removed, the user dismisses it, or its information is no longer valid.

Exception: The visual presentation of the additional content is controlled by the user agent and isn't modified by the author.

While not generally relevant to Process Director in the specific context of a mouseover or hover action, there are times when you want information to appear or disappear based on the value of some evaluable condition, or when you select a control. Process Director provides both Section and Embedded Section controls that enable you to hide or reveal form areas. When shown, these sections are persistent until the condition changes or it is hidden manually. Also, this hiding or displaying of information is generally done via events other than mouseover, in order to provide both persistence, and user-level manual control over when the information is hidden or displayed. As a general rule, attempting to show or hide information based on the position of the mouse isn't a good practice.

Other Accessibility Principles

Principle 2 - Operable: This principle relates to how user interface components and navigation operate.

Principle 3 - Understandable: This principle relates to making information and the operation of the user interface understandable to users.

Principle 4 - Robust: This principle relates to making content robust enough to be interpreted by a wide variety of user agents, including assistive technologies.

Sample Accessible Colors

For your convenience, we've provided a list of colors and their contrast value that you make want to consider for use in UI elements, such as buttons. You can, for instance, add these colors to your customization file using the WorkflowColors Custom Variable.

The colors presented below are grouped by shade, with the appropriate hexadecimal HTML colors and level of WCAG compliance, i.e., AA or AAA. Most of the colors are AAA-compliant. The color names we've provided are arbitrary, and do not necessarily correspond to the formal HTML color names (though some do).

Please note that the HTML named colors **Red** and **Green**, are particularly difficult to use at the AAA compliance level. These mid-range colors do not particularly contrast well with either white or black text. You may wish to replace these colors with the colors we've designated as **Red (Contrast)** [`B10000`] and **Green (Contrast)** [`006200`] below, both of which are dark enough to be AAA-compliant when contrasted with white.

You can, of course, check contrasts for your own UI colors at ContrastChecker.com, or many others available online.

Compliance Levels

AA: Contrast ratio > 4.5:1

AA*: Contrast ratio > 3:1 for 18pt or larger text.

AAA: Contrast ratio > 7:1

AAA*: Contrast ratio > 4.5:1 for 18pt or larger text.

Suggested Colors

Shade	Hexadecimal Color	Suggested Name	Contrast Ratio With Black/White Text	WGAC Compliance Level
Grayscale	`000000`	Black	21	AAA
	`333333`	Dark Gray	12.63	AAA
	`555555`	Gray	7.46	AAA
	`CCCCCC`	Light Gray	13.08	AAA
	`F1F1F1`	Smoke	18.59	AAA
	`FFFFFF`	White	21	AAA

Shade	Hexadecimal Color	Suggested Name	Contrast Ratio With Black/White Text	WGAC Compliance Level
Blue	000066	Dark Blue	17.62	AAA
	0000FF	Blue	8.59	AAA
	96B6F5	Light Blue	10.31	AAA
	E2E2FE	Pale Blue	16.54	AAA
Purple	2C0047	Dark Purple	17.53	AAA
	6300A6	Purple	10.09	AAA
	C972FF	Lavender	7.33	AAA
	ECCEFF	Pale Lavender	14.84	AAA
Red	5C0000	Dark Red	14.43	AAA
	B10000	Red (Contrast)	7.08	AAA
	FF0000	Red (HTML)	5.25	AA, AAA*
	FF0000	Red (HTML)	4	AA*
	FF7F7F	Light Red	8.59	AAA
	FFC0C0	Pink	13.55	AAA
Orange	532D00	Brown	12.05	AAA
	FA8800	Orange	8.59	AAA
	FFBB69	Peach	12.56	AAA
	FFE3C0	Buff	17	AAA
Yellow	5E5400	Dark Yellow	7.65	AAA
	FFE600	Bold Yellow	16.57	AAA
	FFFF00	Yellow	19.56	AAA
	FBF3AA	Light Yellow	18.5	AAA

Shade	Hexadecimal Color	Suggested Name	Contrast Ratio With Black/White Text	WGAC Compliance Level
Green	003300	Dark Green	14.25	AAA
	006200	Green (Contrast)	7.64	AAA
	008000	Green (HTML)	4.09	AA*
	008000	Green (HTML)	5.14	AA, AAA*
	7CFF7C	Light Green	16.45	AAA
	C9FFC9	Mint	18.63	AAA
Teal	002D40	Dark Teal	14.49	AAA
	035D79	Teal	7.38	AAA
	2AEEFF	Light Teal	14.77	AAA
	C7FBFF	Mist	18.67	AAA

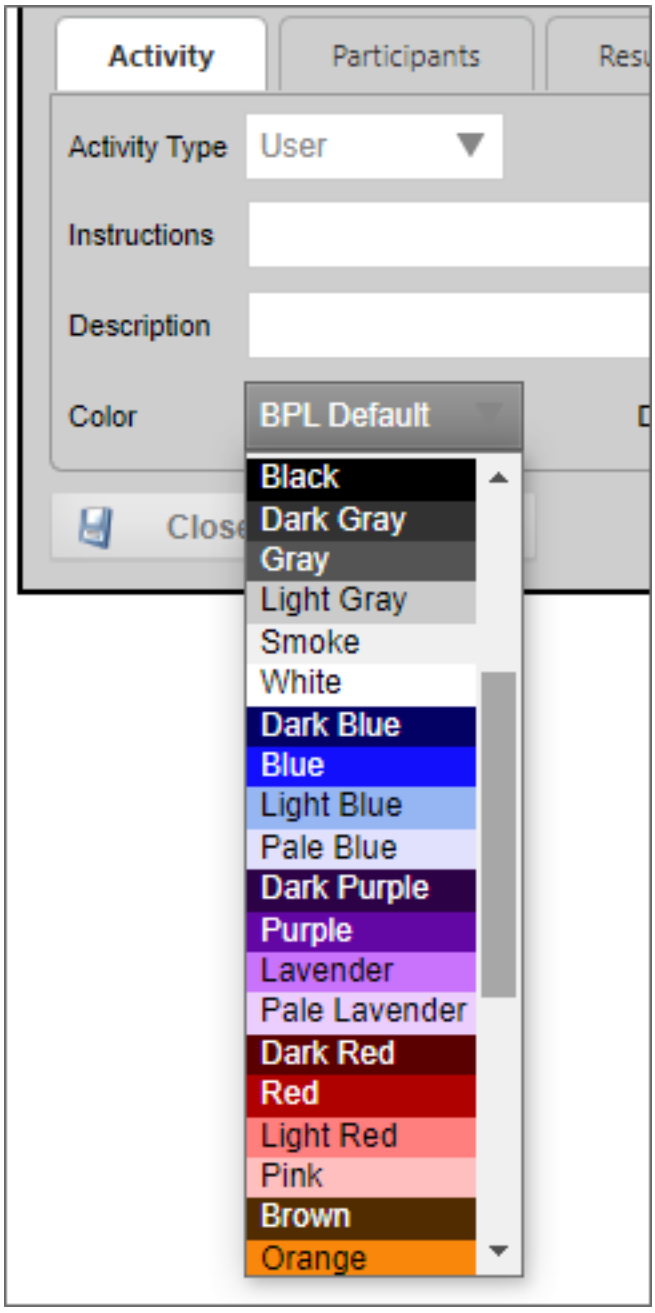
Should you wish to add the color set above to your installation, you can copy the code below, as is, into the `PreSetSystemVars()` section of your Process Director Installation's Custom Variables file.

```
bp.Vars.WorkflowColors = new List<ColorValue>();
bp.Vars.WorkflowColors.Add(new ColorValue("Default",
"#054FB9", "#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Black", "#000000",
"#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Dark Gray",
"#333333", "#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Gray", "#555555",
"#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Light Gray",
"#CCCCCC", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Smoke", "#F1F1F1",
"#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("White", "#FFFFFF",
"#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Dark Blue",
"#000066", "#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Blue", "#0000FF",
"#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Light Blue",
```

```
"#96B6F5", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Pale Blue",
"#E2E2FE", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Dark Purple",
"#2C0047", "#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Purple", "#6300A6",
"#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Lavender",
"#C972FF", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Pale Lavender",
"#ECCEFF", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Dark Red",
"#5C0000", "#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Red", "#B10000",
"#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Light Red",
"#FF7F7F", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Pink", "#FFC0C0",
"#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Brown", "#532D00",
"#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Orange", "#FA8800",
"#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Peach", "#FFBB69",
"#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Buff", "#FFE3C0",
"#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Dark Yellow",
"#5E5400", "#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Bold Yellow",
"#FFE600", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Yellow", "#FFFF00",
"#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Light Yellow",
"#FBF3AA", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Dark Green",
"#003300", "#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Green", "#006200",
"#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Light Green",
"#7CFF7C", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Mint", "#C9FFC9",
"#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Dark Teal",
"#002D40", "#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Teal", "#035D79",
"#FFFFFF"));
bp.Vars.WorkflowColors.Add(new ColorValue("Light Teal",
```

```
"#2AEEFF", "#000000"));
bp.Vars.WorkflowColors.Add(new ColorValue("Mist", "#C7FBFF",
"#000000"));
```

The code above will **replace** the default colors that are installed with Process Director with the new color set shown above. You can also simply add these colors to the existing default colors by deleting the first two lines of code in the example above. Once applied, this color set will appear in the UI's color chooser dropdowns like the example below.



Accessibility Principles

Principle 1 - Perceivable: This principle relates to making information and user interface components presentable to users in ways they can perceive.

Principle 2 - Operable: This principle relates to how user interface components and navigation operate.

Principle 3 - Understandable: This principle relates to making information and the operation of the user interface understandable to users.

Principle 4 - Robust: This principle relates to making content robust enough to be interpreted by a wide variety of user agents, including assistive technologies.

Accessibility Principle 2 – Operable

This principle relates to how user interface components and navigation operate.

2.1.1 Keyboard

Level A - All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes, except where the underlying function requires input that depends on the path of the user's movement and not just the endpoints.

Process Director uses HTML controls and links for all form controls. All user-accessible objects are navigable via keyboard, e.g., users can use the tab key to navigate from one control to the next. The path of the user's movement is controlled by the location of the control on the form. Navigation operates from left to right and from top to bottom for browsers that have European language settings, and from right to left and top to bottom for other browser language settings. Additionally, the use of the Set Focus Custom Task enables you to specify a custom tab order, if you desire, though this is generally not necessary in the context of a form, which generally has a logical structure to begin with, and corresponds with the natural reading order of the specified culture.

2.1.2 No Keyboard Trap

Level A - If keyboard focus can be moved to a component of the page using a keyboard interface, then focus can be moved away from that component using only a keyboard interface, and, if it requires more than unmodified arrow or tab keys or other standard exit methods, the user is advised of the method for moving focus away.

Note 1: Since any content that doesn't meet this success criterion can interfere with a user's ability to use the whole page, all content on the Web page (whether it is used to meet other success criteria or not) must meet this success criterion.

Though all controls on a form are keyboard navigable automatically through a set tab order in Process Director, it is possible for you to create a keyboard trap when using the Set Focus Custom Task. For instance, you can set the focus to field 3 when a customer completes field 1, then, in field 3, set the focus back to field one.

This would create a never-ending tab cycle between field 1 and field 3, thus creating a keyboard trap, requiring the user to use the mouse to navigate to a different field. You should, as a best practice, extensively test any use of the Set Focus Custom Task to ensure you don't create a keyboard trap.

2.1.3 Keyboard (No Exception)

Level AAA - All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes.

Process Director's tab order is generally compliant with this specification, though, as mentioned in 1.2.1, above, you can, as a designer, break it.

2.1.4 Character Key Shortcuts

Level A- If a keyboard shortcut is implemented in content using only letter (including upper- and lower-case letters), punctuation, number, or symbol characters, then at least one of the following is true:

Turn off: A mechanism is available to turn the shortcut off;

Remap: A mechanism is available to remap the shortcut to include one or more non-printable keyboard keys (e.g., Ctrl, Alt);

Active only on focus: The keyboard shortcut for a user interface component is only active when that component has focus.

This is generally not relevant to Process Director.

Guideline 2.2 – Enough Time

Provide users enough time to read and use content.

2.2.1 - Timing Adjustable

Level A - For each time limit that is set by the content, at least one of the following is true:

Turn off: The user is allowed to turn off the time limit before encountering it; or

Adjust: The user is allowed to adjust the time limit before encountering it over a wide range that is at least ten times the length of the default setting; or

Extend: The user is warned before time expires and given at least 20 seconds to extend the time limit with a simple action (for example, "press the space bar"), and the user is allowed to extend the time limit at least ten times; or

Real-time Exception: The time limit is a required part of a real-time event (for example, an auction), and no alternative to the time limit is possible; or

Essential Exception: The time limit is essential and extending it would invalidate the Activity; or

20 Hour Exception: The time limit is longer than 20 hours.

This is generally not relevant to Process Director forms, as they aren't content-timed. This may be relevant to Knowledge Views, however, which do have a Automatically refresh results (in seconds) property available on the Configure tab of the Knowledge View Definition. As a best practice, ensure that you don't impose automatic refreshes on Knowledge Views unless it is essential for real-time event viewing.

2.2.2 Pause, Stop, Hide

Level A - For moving, blinking, scrolling, or auto-updating information, all of the following are true:

Moving, blinking, scrolling: For any moving, blinking or scrolling information that (1) starts automatically, (2) lasts more than five seconds, and (3) is presented in parallel with other content, there is a mechanism for the user to pause, stop, or hide it unless the movement, blinking, or scrolling is part of an Activity where it is essential; and

Auto-updating: For any auto-updating information that (1) starts automatically and (2) is presented in parallel with other content, there is a mechanism for the user to pause, stop, or hide it or to control the frequency of the update unless the auto-updating is part of an Activity where it is essential.

This is generally not relevant to Process Director.

2.2.3 No Timing

Level AAA - Timing isn't an essential part of the event or Activity presented by the content, except for non-interactive synchronized media and real-time events.

Process Director is generally compliant with the specification, unless an automatic refresh is intentionally implemented by the designer, as in the case of Knowledge Views, as mentioned in 2.2.1, above.

2.2.4 Interruptions

Level AAA - Interruptions can be postponed or suppressed by the user, except interruptions involving an emergency.

Process Director is generally compliant with this specification, as users control the timing of data submission by manual action.

2.2.5 Re-authenticating

Level AAA - When an authenticated session expires, the user can continue the Activity without loss of data after re-authenticating.

Reauthentication isn't required in process Director by default, though designers can impose it. For example, in the Advanced tab of a User Timeline Activity, there is a Re-authenticate users property that can be set to require a user to reauthenticate *prior to* performing an Activity, but this reauthentication occurs before

the data entry session is invoked. While it is possible to set up more onerous reauthentication requirements outside of Process Director, i.e., at the network security level, and Process Director will respect those requirements, Process Director doesn't, by default, require reauthentication or time users out.

2.2.6 Timeouts

Level AAA - Users are warned of the duration of any activity that could cause data loss, unless the data is preserved for more than 20 hours when the user doesn't take any actions.

As mentioned in 2.2.5, above, this isn't generally relevant to Process Director.

Guideline 2.3 – Seizures and Physical Reactions

Do not design content in a way that is known to cause seizures or physical reactions.

2.3.1 Three Flashes or Below Threshold

Level A - Web pages don't contain anything that flashes more than three times in any one second period, or the flash is below the general flash and red flash thresholds.

As a best practice, designers should never include such elements in an object viewable by end users. Additionally, any motion animation triggered by user interaction must have the option for end users to disable the animation, unless the animation is essential to the functionality or the information being conveyed.

2.3.2 Three Flashes

Level AAA - Web pages don't contain anything that flashes more than three times in any one second period.

As a best practice, designers should never include such elements in an object viewable by end users.

2.3.3 Animation from Interactions

Level AAA - Motion animation triggered by interaction can be disabled, unless the animation is essential to the functionality or the information being conveyed.

As a best practice, designers should never include such elements in an object viewable by end users.

Guideline 2.4 – Navigable

Provide ways to help users navigate, find content, and determine where they are.

2.4.1 Bypass Blocks

Level A - A mechanism is available to bypass blocks of content that are repeated on multiple Web pages.

While it's difficult to think of a use case in which this might be relevant, the Section and Embedded Section controls can be used to enable users to show or hide repeated blocks of content, as a best practice.

2.4.2 Page Titled

Level A - Web pages have titles that describe topic or purpose.

All user-viewable objects in Process Director have an instantiated name property that can be customized. This instance name serves as the page title when displayed to end users. As a best practice, designers should provide the appropriate instance names for every object. By default, when an object is created, a generic instance name is created for that object. For example, when creating a new form definition, the default Instantiated Form Name is "{FORM_DEF_NAME} Submitted On {CREATE_DATE}". When a new instance of a form definition named "Vacation Request" is created on 1 January, 2020, the user will see the title of the page as "Vacation Request Submitted on 1/1/2020".

2.4.3 Focus Order

Level A - If a Web page can be navigated sequentially and the navigation sequences affect meaning or operation, focusable components receive focus in an order that preserves meaning and operability.

This specification is generally covered b the answers provided in 2.2.1-2.2.3 above, along with the same caveat that the use of the Set Focus Custom Task requires extensive testing to ensure that the focus is properly navigable by end users.

2.4.4 Link Purpose (In Context)

Level A - The purpose of each link can be determined from the link text alone or from the link text together with its programmatically determined link context, except where the purpose of the link would be ambiguous to users in general.

As a best practice, designers creating hyperlinks using the Hotlink control, or manually creating hyperlinks in the raw HTML of an HTML control, should comply with this requirement, so that end user know the context and purpose of the link before clicking it.

2.4.5 Multiple Ways

Level AA - More than one way is available to locate a Web page within a set of Web pages except where the Web Page is the result of, or a step in, a process.

In addition to the links provided to objects in the user's Task List, or in the Forms I can Submit Knowledge View, designer should, as a best practice, incorporate navigation buttons and/or dashboard/workspace links to all forms or other objects to which users have access. Workspace, dashboard, and Desktop interface links are fully configurable by designers.

2.4.6 Headings and Labels

Level AA - Headings and labels describe topic or purpose.

In addition to the automatic heading incorporated int Process Director's Section controls, The Online Form Designer also incorporates Header formatting for other content. As a best practice, designers should ensure that all heading and labels are appropriately descriptive.

2.4.7 Focus Visible

Level AA - Any keyboard operable user interface has a mode of operation where the keyboard focus indicator is visible.

Process Director is generally compliant with this specification, and highlights the field/control that has the focus when using keyboard navigation, i.e., tabbing between controls.

Guideline 2.5 – Input Modalities

Make it easier for users to operate functionality through various inputs beyond keyboard.

2.5.1 Pointer Gestures

Level A - All functionality that uses multipoint or path-based gestures for operation can be operated with a single pointer without a path-based gesture, unless a multipoint or path-based gesture is essential.

This is generally not relevant to Process Director.

2.5.2 Pointer Cancellation

Level A - For functionality that can be operated using a single pointer, at least one of the following is true:

No Down-Event: The down-event of the pointer isn't used to execute any part of the function;

Abort or Undo: Completion of the function is on the up-event, and a mechanism is available to abort the function before completion or to undo the function after completion;

Up Reversal: The up-event reverses any outcome of the preceding down-event;

Essential: Completing the function on the down-event is essential.

This is generally not relevant to Process Director.

2.5.3 - Label in Name

Level A - For user interface components with labels that include text or images of text, the name contains the text that is presented visually.

This is generally not relevant to Process Director, as all labels are text-based.

2.5.4 Motion Actuation

Level A - Functionality that can be operated by device motion or user motion can also be operated by user interface components and responding to the motion can be disabled to prevent accidental actuation.

This is generally not relevant to Process Director.

2.5.5 Target Size

Level AA - The size of the target for pointer inputs is at least 44 by 44 CSS pixels except when:

Equivalent: The target is available through an equivalent link or control on the same page that is at least 44 by 44 CSS pixels;

Inline: The target is in a sentence or block of text;

User Agent Control: The size of the target is determined by the user agent and isn't modified by the author;

Essential: A particular presentation of the target is essential to the information being conveyed.

As a best practice, designers should ensure that user interface targets are sized appropriately.

Other Accessibility Principles

Principle 1 - Perceivable: This principle relates to making information and user interface components presentable to users in ways they can perceive.

Principle 3 - Understandable: This principle relates to making information and the operation of the user interface understandable to users.

Principle 4 - Robust: This principle relates to making content robust enough to be interpreted by a wide variety of user agents, including assistive technologies.

Accessibility Principle 3 – Understandable

This principle relates to making information and the operation of the user interface understandable to users.

Guideline 3.1 – Readable

Make text content readable and understandable.

3.1.1 Language of Page

Level A - The default human language of each Web page can be programmatically determined.

Process Director's entire user interface is customizable for culture/language, based on the user's selected culture setting. All forms and other user-viewable objects are also fully customizable for culture as well. Please see the Localization topic of the Developers Guide.

3.1.2 Language of Parts

Level AA - The human language of each passage or phrase in the content can be programmatically determined except for proper names, technical terms, words of indeterminate language, and words or phrases that have become part of the vernacular of the immediately surrounding text.

This is generally not relevant to Process Director, though designers can, through the use of the HTML control, provide the appropriate language indicators in the raw HTML via the "lang" attribute of an HTML text tag.

Code Sample

```
<blockquote lang="de">
    <p>
        Da dachte der Herr daran, ihn aus dem Futter zu schaf-
fen,
        aber der Esel merkte, daß kein guter Wind wehte, lief
fort
        und machte sich auf den Weg nach Bremen: dort, meinte
er,
        könnte er ja Stadtmusikant werden.
    </p>
</blockquote>
```

3.1.3 Unusual Words

Level AAA - A mechanism is available for identifying specific definitions of words or phrases used in an unusual or restricted way, including idioms and jargon.

This isn't generally relevant to Process Director, though designers can, through the use of the HTML control, use the "dfn" tag to specify a definition.

Code Sample

```
<p>The Web Content Accessibility Guidelines require that non-
text content has a text alternative. <dfn>Non-text
content</dfn> is content that isn't a sequence of characters
that can be programmatically determined or where the sequence
isn't expressing something in human language; this includes
ASCII Art (which is a pattern of characters), emoticons, leet-
speak (which is character substitution), and images rep-
resenting text .</p>
```

Guideline 3.2 – Predictable

Make Web pages appear and operate in predictable ways.

3.2.1 On Focus

Level A - When any user interface component receives focus, it doesn't initiate a change of context.

This is generally not relevant to Process Director. To the extent context is changed, it is changed as the result of a manual user input, not a result of simply gaining focus.

3.2.2 On Input

Level A - Changing the setting of any user interface component doesn't automatically cause a change of context unless the user has been advised of the behavior before using the component.

As a best practice, designers should provide users with explanatory text that a change to a user input will change context, so that users understand the result of any input action.

3.2.3 Consistent Navigation

Level AA - Navigational mechanisms that are repeated on multiple Web pages within a set of Web pages occur in the same relative order each time they are repeated, unless a change is initiated by the user.

As a best practice, designers should ensure that any navigation scheme they create is consistent, so that the user receives the same experience over time. This is important in form design, particularly in the use of result buttons for user action buttons displayed on a form when the user is in the context of a task. The order of positive and negative result buttons, their colors, and, to the extent possible, their text, should be the same in all forms. In other words, one form shouldn't present the options Approve, Deny, while a different form displays them as Deny, Approve. The user experience should be as consistent as possible.

3.2.4 Consistent Identification

Level AA - Components that have the same functionality within a set of Web pages are identified consistently.

As a best practice, similar to 3.2.3, above, component definitions and explanations should be consistent.

Guideline 3.3 – Input Assistance

Help users avoid and correct mistakes.

3.3.1 Error Identification

Level A - If an input error is automatically detected, the item that is in error is identified and the error is described to the user in text.

In addition to the default test notifications for required fields, wrong data types, etc., Process Director enables designers to create their own custom validation conditions and associated text notifications. As a best practice, always ensure that your validation notifications are helpful and descriptive. By default, error notifications appear in text banners at the top and bottom of the form, though designers can change these locations through the use of the Form Error/Info String controls in the form.

3.3.2 Labels or Instructions

Level A - Labels or instructions are provided when content requires user input.

Process Director provides a number of methods for designers to accomplish this. In addition to the Label controls, input fields have an Empty Text property and Tooltip property that should be used to provide brief instructions about what information goes into the field. Additional instructions can also be provided via alert boxes using the Alert Custom Task.

3.3.3 Error Suggestion

Level AA - If an input error is automatically detected and suggestions for correction are known, then the suggestions are provided to the user, unless it would jeopardize the security or purpose of the content.

As a best practice, designers should include appropriate error correction suggestions in the error message they configure.

3.3.4 Error Prevention (Legal, Financial, Data)

Level AA - For Web pages that cause legal commitments or financial transactions for the user to occur, that modify or delete user-controllable data in data storage systems, or that submit user test responses, at least one of the following is true:

Reversible: Submissions are reversible.

Checked: Data entered by the user is checked for input errors and the user is provided an opportunity to correct them.

Confirmed: A mechanism is available for reviewing, confirming, and correcting information before finalizing the submission.

While the onus of incorporating this specification lies mainly on the process designer, Process Director provides a number of tools such as alert boxes in the form, confirmation requirements in the Results tab of User Timeline Activities, requiring text comments for a specified results, custom validation rules, and many more. Additionally, the Compliance component enables form field auditing when turned on via the Audit Form field changes property of the Form definition. Implementing this type of error prevention can be complex, so BP Logix is always ready to offer Direct Assistance for such implementations.

3.3.5 Help

Level AAA - Context-sensitive help is available.

Process Director provides a number of ways to provide context sensitive help, such as tooltips, empty text for form fields, and callable alert boxes. Designers should, as a best practice, incorporate context-sensitive help appropriately.

Other Accessibility Principles

Principle 1 - Perceivable: This principle relates to making information and user interface components presentable to users in ways they can perceive.

Principle 2 - Operable: This principle relates to how user interface components and navigation operate.

Principle 4 - Robust: This principle relates to making content robust enough to be interpreted by a wide variety of user agents, including assistive technologies.

Accessibility Principle 4 – Robust

This principle relates to making content robust enough to be interpreted by a wide variety of user agents, including assistive technologies.

Guideline 4.1 – Compatible

Maximize compatibility with current and future user agents, including assistive technologies.

4.1.1 Parsing

Level A - In content implemented using markup languages, elements have complete start and end tags, elements are nested according to their specifications, elements don't contain duplicate attributes, and any IDs are unique, except where the specifications allow these features.

This is a core feature of Process Director; however, designers must also ensure that the use of HTML controls uses raw HTML that is similarly parsable within the DOM.

4.1.2 Name, Role, Value

Level A - For all user interface components (including but not limited to: form elements, links and components generated by scripts), the name and role can be programmatically determined; states, properties, and values that can be set by the user can be programmatically set; and notification of changes to these items is available to user agents, including assistive technologies.

This is a core feature of Process Director.

Tables manually inserted into a Form should have the Role attribute set appropriately, as outlined in the Online Form Designer topic.

4.1.3 Status Messages

Level AA - In content implemented using markup languages, status messages can be programmatically determined through role or properties such that they can be presented to the user by assistive technologies without receiving focus.

This is a core feature of Process Director. Designer-configured status messages, such as validation errors, are always included in the existing, text-based, messaging system, which never received focus when error or other status messages are displayed.

Other Accessibility Principles

Principle 1 - Perceivable: This principle relates to making information and user interface components presentable to users in ways they can perceive.

Principle 2 - Operable: This principle relates to how user interface components and navigation operate.

Principle 3 - Understandable: This principle relates to making information and the operation of the user interface understandable to users.

Accessibility and Responsive Design

n Process Director, responsive design is an approach to creating Forms that ensures the information displayed on the Form is rendered in a manner that makes the Form usable on a variety of devices and screen sizes. A Form that is properly responsive will enable users to view the form legibly, no matter what device or screen size they are using.

Process Director will automatically implement some features of responsive design, just as it does for accessibility. But, like accessibility, the Form designer is also responsible for implementing responsive features, as well.

Responsive design is primarily implemented through the use of Cascading Style Sheet (CSS) styles, either by setting the Style property of controls on the form, or by creating CSS classes, and applying those classes to the form control. For the most part, Form designers do not need to have extensive knowledge of CSS, but there are a few key style commands that you should know to implement responsive design elements, in addition to those that are built into Process Director.

Applying Styles

In Process Director, responsive CSS styles can be applied in three different ways.

Most Form controls have a Style property into which you can add short CSS style commands. Appling styles at the control level in this way usually overrides the styles set for that control in the general Process Director style sheet.

On the Properties tab of the Form definition, the Default Styles section contains a property named Additional Body CSS Styling. This property enables you to add a custom CSS style sheet to the form. Form controls can then apply the styles named in that stylesheet by typing a style name in the CSS Class property of the Form control.

You can create a custom CSS stylesheet and upload it into the `/custom` folder of your Process Director installation. On the Properties page of the IT Admin area's Installation Settings section, the CSS property enables you to specify the URL of the CSS stylesheet. Once you do so, that CSS stylesheet becomes universally available for use in any form. Setting the CSS Class property to one of the named styles in the CSS file will apply that style to the control.

For Cloud customers who don't have direct access to their installation's folder structure, BP Logix will upload the custom CSS stylesheet for you via a technical support request.

Brief CSS Primer

CSS describes how HTML elements are to be displayed on screen. Every object on a form, whether it is text, a form control, or any other object, can have a CSS style applied to it. CSS can be a complex subject as there are many different CSS style elements, and different objects may have different styles that are applicable to them. While we can't teach you the details of CSS in this lesson, we can give you a quick overview of how CSS styles work.

CSS styles have a specific syntax. A CSS command consists of a style attribute, a colon, the style setting for that attribute, ended with a semicolon. For example, if you want to create a boldface style, the style attribute is named font-weight, and the setting would be bold. This would be written in a stylesheet as:

```
font-weight:bold;
```

Any text to which this style is applied will appear as boldface text. For instance, if you type this command into the Style property of a Label control, the control's text would appear in boldface when the form runs.

If you wanted this style to be applied in many different places, you might want to create a CSS stylesheet in the Additional Body CSS Styling property of the Form definition. To create a CSS style in a stylesheet, the style needs a name. To do this, we have to add some additional syntax to our boldface setting. The style name would come first, with the style setting for that style name encased in curly brackets, like this:

```
SomeStyleName {
    attribute:setting;
}
```

The name can be an HTML element, in which case, the style will automatically be applied to every instance of that HTML element on the form. For instance, if I want all of the Level 1 headings on a page to be boldfaced, I would need to use the "h1" HTML element as the name for my style:

```
h1 {
    font-weight:bold;
}
```

The name can also be a CSS class, which is a style name you create to apply to different controls or HTML elements, as desired. You can name a class anything you'd like, but you cannot use spaces in the name, and the name must start with a period. So, for example, you could create a CSS style called "boldtext", with the following syntax:

```
.boldface {
    font-weight:bold;
}
```

You could then set the CSS Class property of a Label control to `boldface`, and the style would be applied to the control when the form runs.

A CSS class can have more than one attribute set. For example, if you wanted the text to be both boldface and colored red, you might create the following class:

```
.redbold {
    font-weight:bold;
    color:red;
}
```

Note that, for readability purposes, we've set each attribute on its own line, as we have done for the last curly bracket. We've also indented the attributes. Adding the extra lines and spaces helps us to read the style setting more easily, but doesn't have any effect on how the styles work.

The W3C offers free and extensive training on CSS syntax and practices.

Table Refactoring

For Subscription and Cloud licenses of Process Director, some key elements of responsive design are built into the product. Process Director implements automatic table refactoring, which means that, at small screen sizes, Process Director reformats tables with multiple columns into a single column at smaller screen

sizes. For example, a table that has four columns when viewed on a desktop computer will, when viewed on a phone, be presented in a single column.

Since most form designers use tables to configure the layout of the form, this means that, in most cases, Process Director will automatically implement a responsive view of the form's layout without any additional configuration by the Form designer.

This automatic table refactoring will solve most of the responsive design issues faced by Form designers. You can simply create a form layout using tables and know that, on smaller screen sizes, the form will be displayed properly to the users.

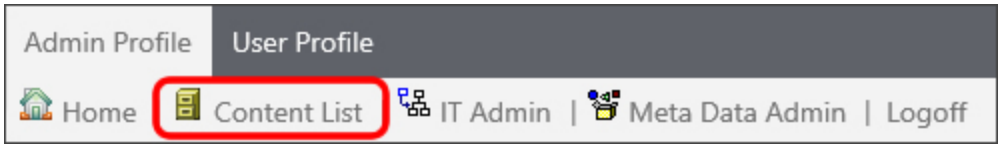
Managing Content

Process Director is a content management system that provides control, security and management of your digital data. This topic provides the essentials for managing objects in the database. All data and documents are typically stored in the database*[1] , which provides easier administration, higher security, distributed/remote database support, and a standard backup mechanism.

Process Director provides a database to securely store content objects. These objects include Folders, Documents, Files, Forms, etc. This chapter will provide an overview of folders, documents, and files. For information on Process Timelines, Forms, or Knowledge Views, refer to the respective chapters in this document.

All content is viewed through the use of Knowledge Views. A Knowledge View defines what data in the repository should be displayed. It is also used to view the Content List.

To view content in the Process Director database for v6.0.X or lower, click on the Content List navigation button in the Admin Profile Workspace.



For v6.1.X or higher, the content list can be displayed by clicking the Content List menu item at the top of the page.

Content List

The Content List is a hierarchical list of all Process Director objects. The Content List includes all of the objects created in Process Director, such as Forms, Process Timelines, Knowledge Views, etc. In addition, the Content List can include uploaded files such as Excel spreadsheets used to import data, Word documents used as templates for output documents, and any other files that become part of a process definition.

---

[1]Process Director also provides file-based document storage as an alternative to database storage. Consult your system administrator regarding the setup at your site.

The Content List consists of a folder structure that is located on the left side of the screen, and a folder/file list that is displayed on the right side of the screen. Navigation through the Content List is done by clicking on a folder on either side of the screen.

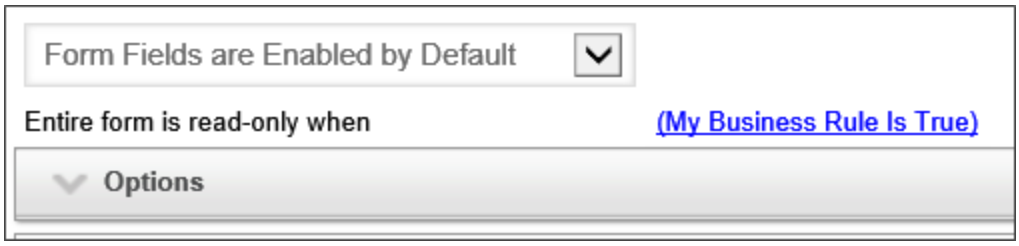
Many users think of the Content List as a file system that users can create inside of Process Director, but this isn't really correct. The Content List organizes Process Director objects in a logical fashion for users who are browsing through the system, but the actual objects in Process Director aren't organized in the way that the Content List displays.

For instance, in a file system, the file name is the key element for identifying a file. Let's say you have a word document named "sample.docx", and you change the name to "NewSample.docx". If you go into Microsoft Word, and try to open the "sample.docx" file from the list of recent files, you'll receive a message telling you that the file can't be found. Changing the name of the file changes the identity of the file on your computer, making any references to the old file name invalid.

In Process Director, on the other hand, an object's name doesn't affect the identity of the object. Process Director tracks objects by a unique ID that you, as an end user will rarely, if ever, use. The object name is merely an attribute of the object that you can change at will without Process Director losing track of the object. Here's an example that demonstrates how object tracking works in Process Director.

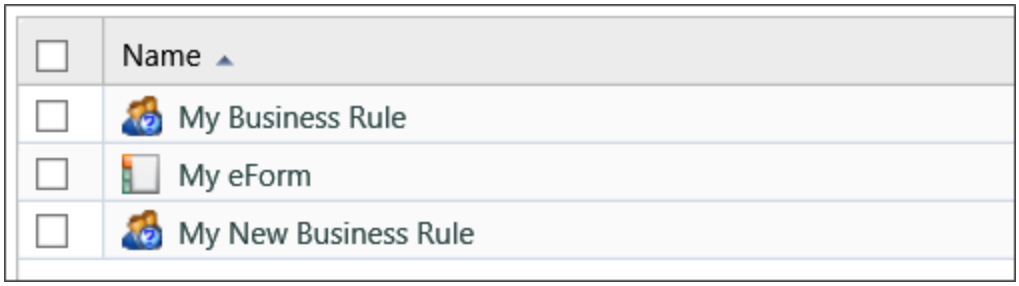
1. In this example, you have a Business Rule named "My Business Rule" and a Form named "My Form".

2. The Form references the Business Rule to determine when to enable the fields on the Form, as shown below.

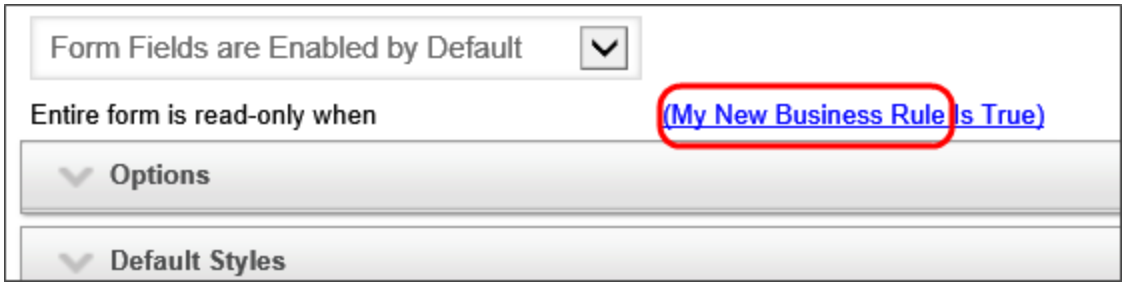


3. Change the name of the Business Rule to "My New Business Rule".

4. Now, create a new Business Rule named "My Business Rule". This is the same name as the original Business Rule.



5. Note that, even though you have a Business Rule with the name "My Business Rule", Process Director doesn't use it as the Business Rule in condition on the Form. Instead, Process Director has updated the Form definition to show the new name you gave to the original Business Rule.



By the way, you can see the ID that Process Director uses to identify an object by opening the object definition. On a Form, for example, at the bottom of the Form

definition screen, you'll see the property for the Form URL, written in a format similar to:

```
https://ServerName.com/form.aspx?pid=dc1c51ea- 612e- 48c0- 8cc6-4ff9276cbfc9&formid=aa213c6f-a8aa-454f-a04d-30b56fd2e493
```

The "formid" URL parameter contains the ID that Process Director uses to uniquely identify the Form object, which, in this example, is

```
aa213c6f-a8aa-454f-a04d-30b56fd2e493
```

Changing the name of the Form, or any other property, doesn't change this ID. So, you can identify the Form with a name that is easily readable by humans, while Process Director identifies the Form with this ID, which is easily readable by computers.

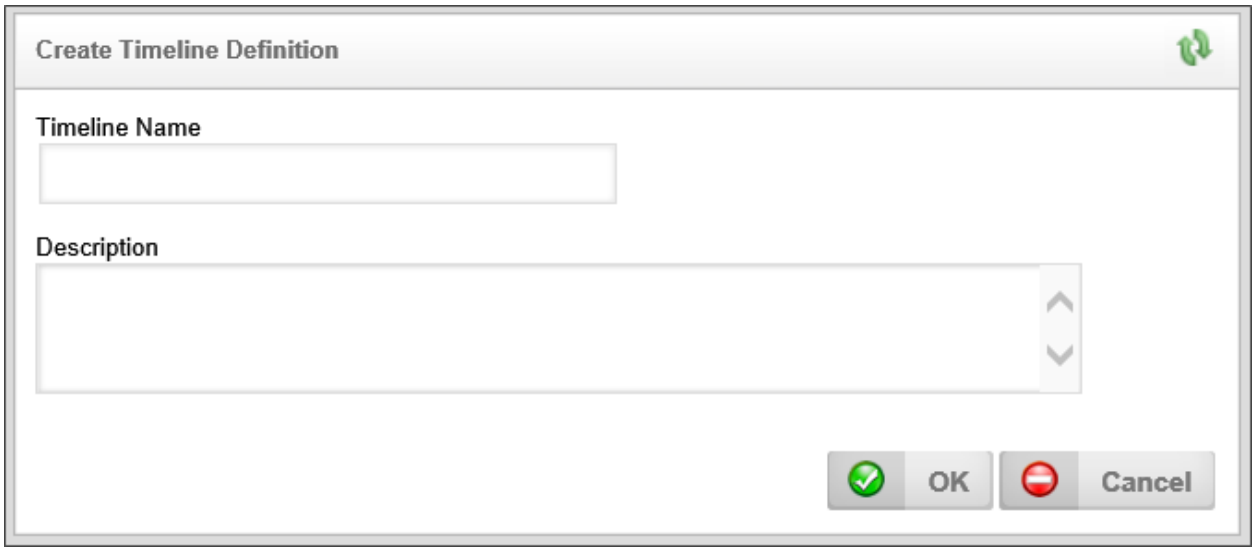
Creating Content List Objects

All objects are created either from the Content List, and, for Process Director v6.0.300 and higher, from the Home page.

Creating from the Content List

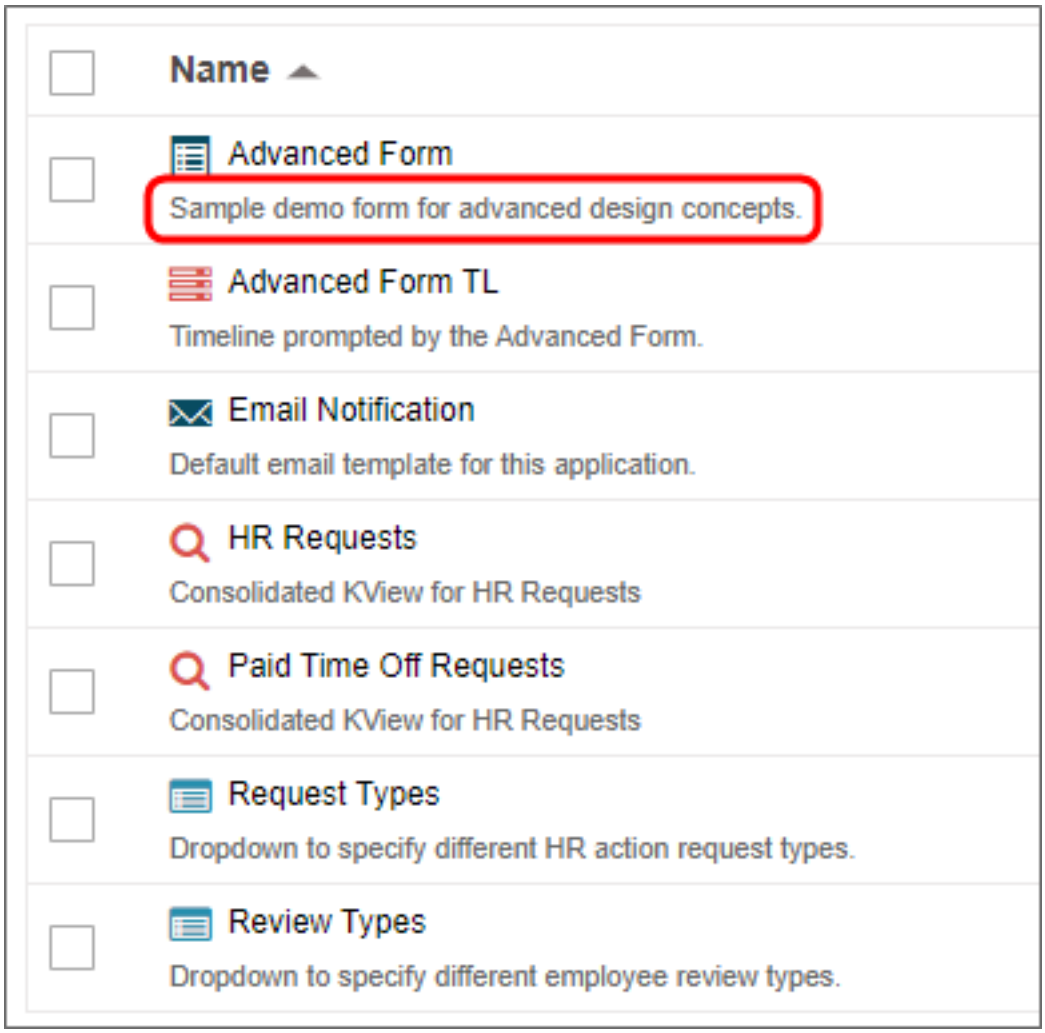
A Create New... dropdown menu is located at the top right each Content List page. This dropdown menu contains all of the process Director objects you can create. To create a new object, simply select the desired object from the dropdown menu.

When you select the desired object, Process Director will immediately display a Create [Object] screen, that will display the Name and Description properties for the object.



The Name property is mandatory, while the Description property is optional. Once you have given the object a Name, you can click the OK button to save and create the new object. If you have configured the Description property, the object's

description will appear on the second line of the Content List where the object appears.



Content List Object Types

The following Content List objects are available in Process Director.

CONTENT LIST OBJECT	DESCRIPTION
Business Rule	The definition for an object that encapsulates a key decision or value used in a business process.
Business Value	The Business Value is an object that provides data virtualization for any accessible external data.

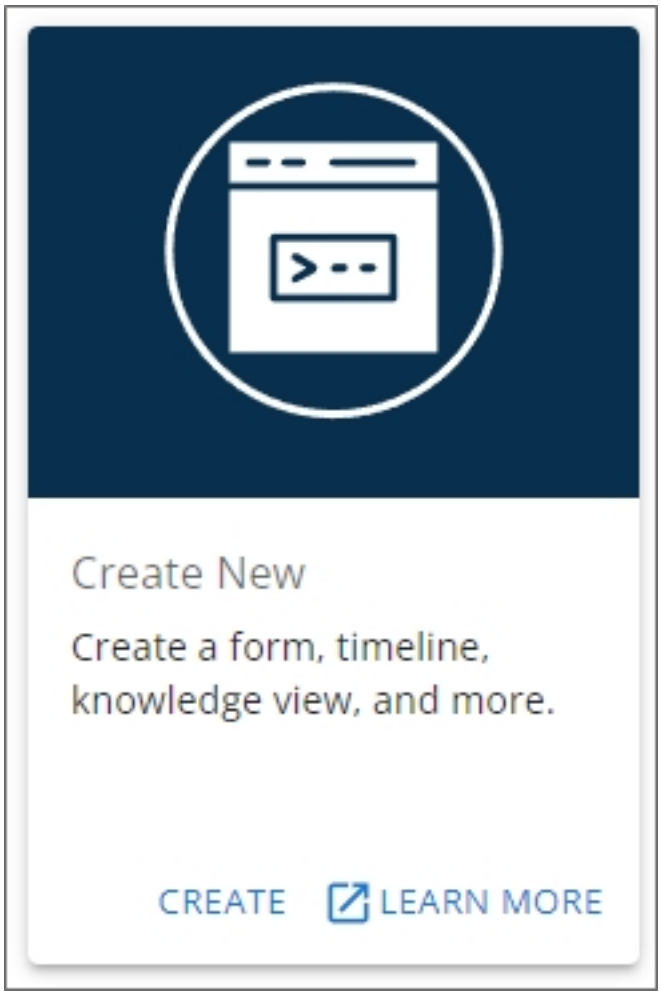
CONTENT LIST OBJECT	DESCRIPTION
Case Definition	The Case definition stores the properties/attributes for a Case Management application. The Properties can be shared across the multiple Forms and processes that might be part of a Case.
Chart	The Chart provides a graphical representation for any accessible data.
Dashboard	The Dashboard object provides a way to display many different Process Director objects in a single interface.
Data Source	An object that accesses information found in a database.
Document/File	The Content List placeholder for documents that you upload to process director for use as part of a process definition.
DropDown Object	The definition of labels and values that will be applied to a dropdown control on a Form.
Folder	The folder is the container for all other Content List objects.
Form Definition	The definition of an electronic form. It stores the Form template, the fields that are created in the template, the user-defined events that are initiated while using the Form, and the validation rules for determining if the information provided in a Form is valid when submitted.
Goal	The Goal object creates a scheduled evaluation of conditions, and, based on the evaluation, sets a global system state and/or automatically starts a process.
Import Objects	Implements a process to import an XML file

CONTENT LIST OBJECT	DESCRIPTION
	containing an exported set of Process Director objects.
Knowledge View	The definition of a view of the process and/or Form data, containing user-defined fields for display in a tabular format.
Machine Learning Definition	An object that enables you to use Process Director's Artificial Intelligence capabilities to review a dataset, and make predictions based on the state of that dataset.
Process Timeline	A process model that implements the patented BP Logix method of implementing a time-based method of modeling processes.
Report	The definition of a report object that uses the advanced reporting tool. The advanced reporting tool is available to users of cloud-based installations, and on-premise installations that explicitly include the Advanced Reporting module.
Stream Action	This object will read a recordset stream from a Datasource, and enable you to submit a form instance for each record in the stream.
Workflow Definition	A process model that implements the traditional, flowchart-based method of modeling processes. **This model is the legacy process model for early versions of Process Director, and has been largely replaced by the patented Process Timeline object. New process features since Process Director v4.5 have been added only to the**

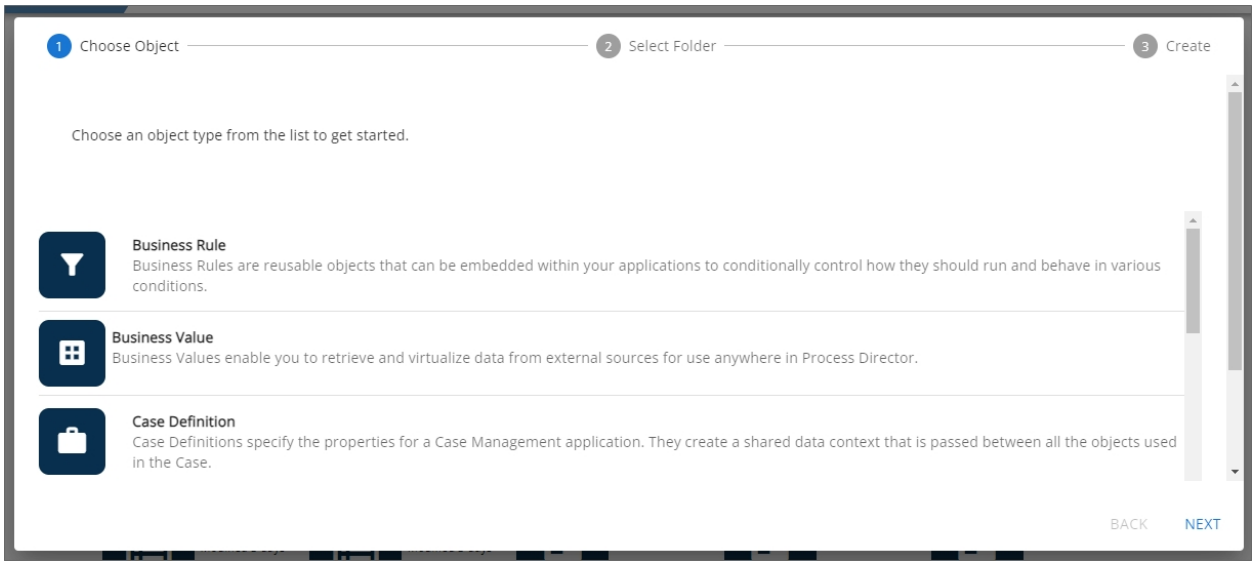
CONTENT LIST OBJECT	DESCRIPTION
	Process Timeline.

Creating From the Home Page

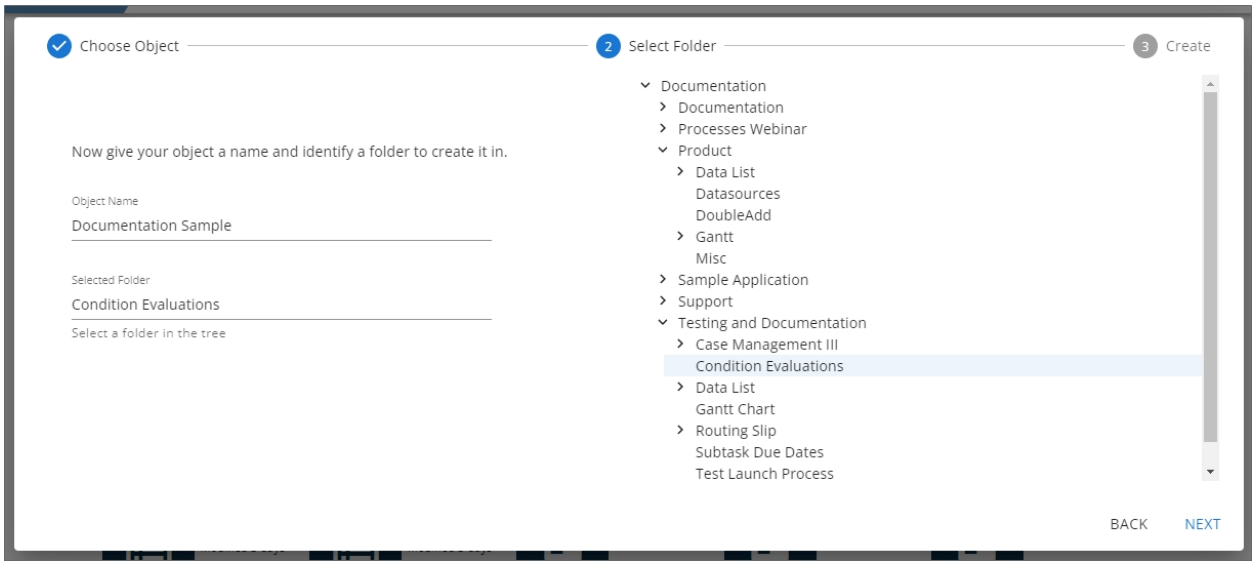
For Process Director v6.0.300 and higher, the Home page displays a panel labeled "Create New", which enables you to create a new Content List object. A Learn More link will, when clicked, open this documentation topic in a new window.



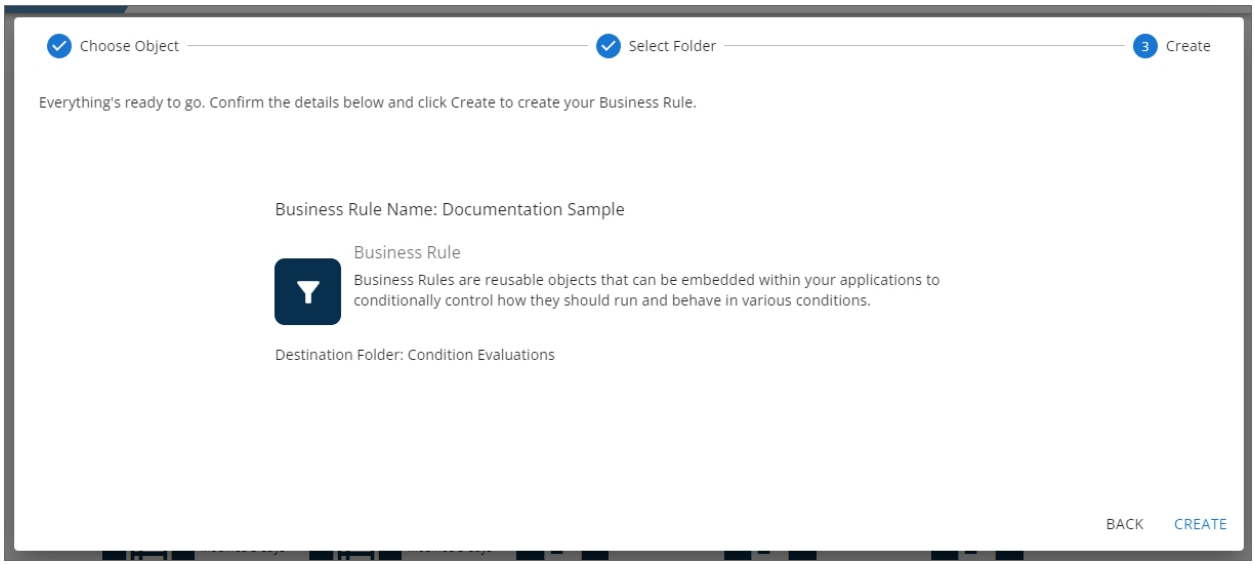
The Create link will, when clicked, open a wizard that will guide you through the process of creating the new object you desire.



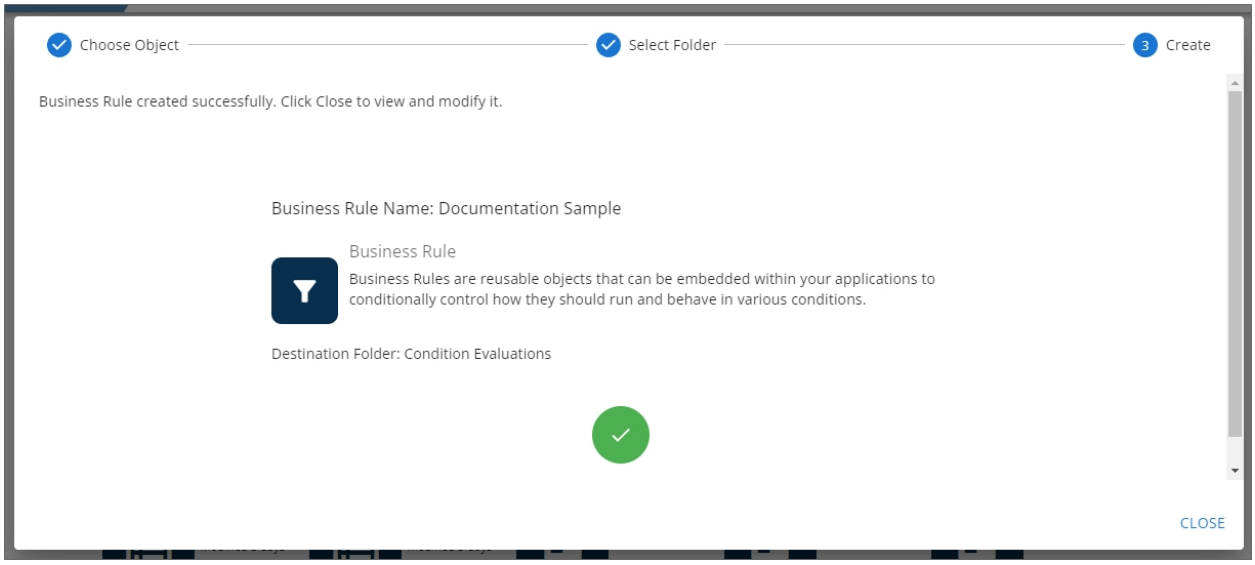
The first pane of the wizard is the Choose Object pane, which displays a list of all Process Director objects you might wish to create. Simply scroll to the object you wish to create, and click it to select it. Once you've done so, click the Next button at the bottom right corner of the wizard to advance to the Select Folder pane.



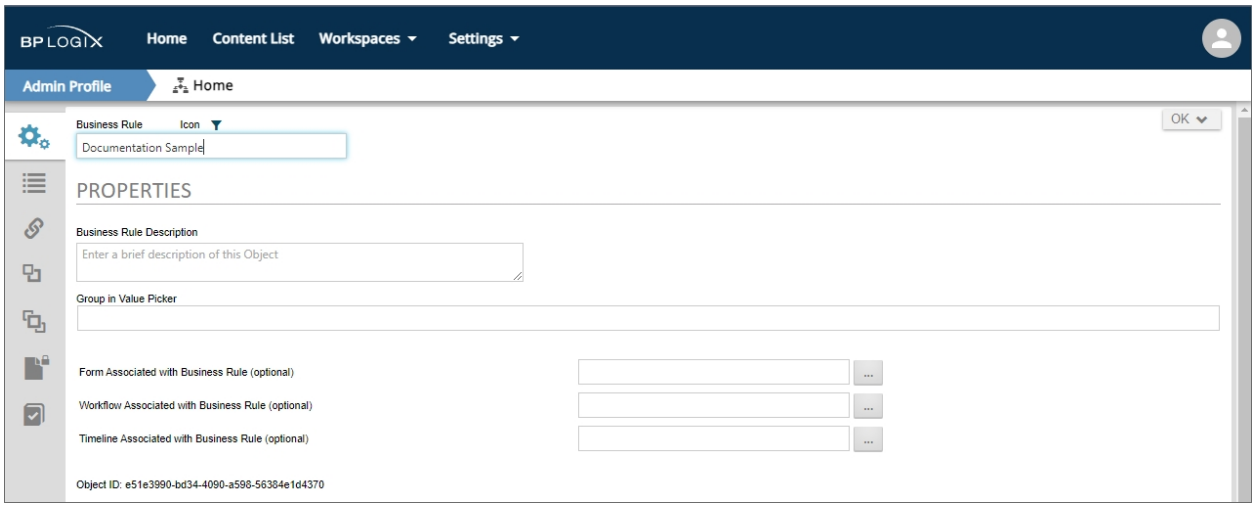
On the left side of the Select Folder pane, enter the name for the new object in the Object Name property. Once you've done so, you can navigate through the folder list displayed on the right side of the pane to find and select the folder into which you want to add the new object by clicking it (The folder must already exist in the Content List). When you do so, the folder name will appear automatically in the Selected Folder property, as shown above. When you're done, click the Next button again.



The Create pane enables you to review the choices you've made in the previous two wizard panes. You can, at any time, click the Back button to return to a previous pane to edit your choices. Once you finished reviewing your choices, you can click the Create button to create the new object. When you click the create button, a "working" animated icon will briefly appear, followed by a success icon when the object has been created.



Click the Close button to exit the wizard. Process Director will automatically navigate you to the new object definition and open it for editing, so that you can complete its configuration.



Creating a New Application

Most Process Director implementations consist of standalone applications. Each application should reside in a single Content List folder that contains all of the objects that you create as part of the application. Placing your application inside a single folder makes importing and exporting your application between the development and production environment much easier.

As a minimum, an working application requires at least two objects, a Form and Process Timeline that, when linked, provide the data used by the application (the Form), and the process the application will use to perform the activities required to operate (the Process Timeline).

The Form and Process Timeline must be associated with each other to ensure that 1) the Form, when submitted, starts the Process Timeline properly, and 2) the Process Timeline can reference or evaluate any required Form field values correctly. Together, the linked Form and Process Timeline provide the core of your application, or the "application stub". The application may require the use of other objects, such as Business Rules, Knowledge Views, etc., all of which might reside in the application folder, but the Form and Process Timeline application stub will always be the central operating elements of an application.

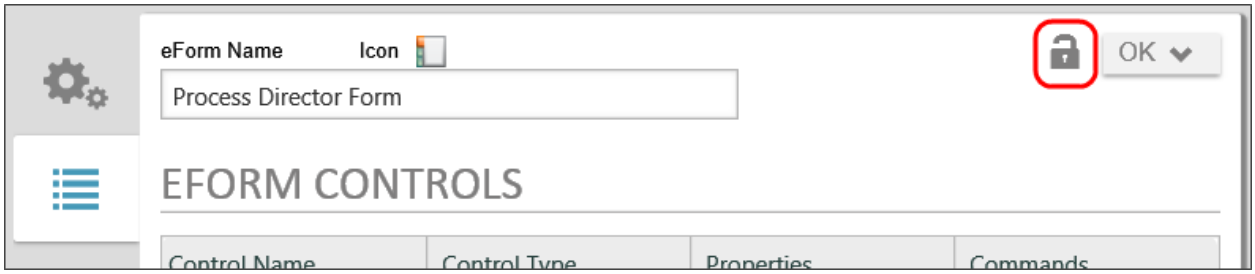
As explained in the Folder Structure section of the Importing/Exporting Content topic, more complex applications might implement more complex methods of organization, but the general rule of thumb is that most application objects should be included in the application's single parent folder.

Object Locking

Process Director v3.49 and above implements a new object locking system for object definitions (e.g., Business Rules, Knowledge Views, or Process Timeline definitions) to ensure that multiple users can't edit the same object definition at the same time, and thus overwrite each others' changes. Process Director implements object locking by default, but object locking can be turned off by setting the ObjectLockingEnable custom variable to "false" in the Custom Vars file.

When object locking is enabled, users who edit an object definition can click the object's Lock Object button to ensure that any other user who opens the definition will see the object only in read only mode. The locking user will be the only user with editing capability while the object is locked.

Once an object is locked, the lock will remain in effect until the locking user manually clicks the object's Release Lock button to release the lock. Even if the locking user saves or closes the form, or logs out of Process Director, the object will remain locked until the locking user manually releases the lock.



Object locking can also be required by setting the ObjectLockingForce custom variable to "true" in the Custom Vars file.

BP Logix recommends requiring object locking.

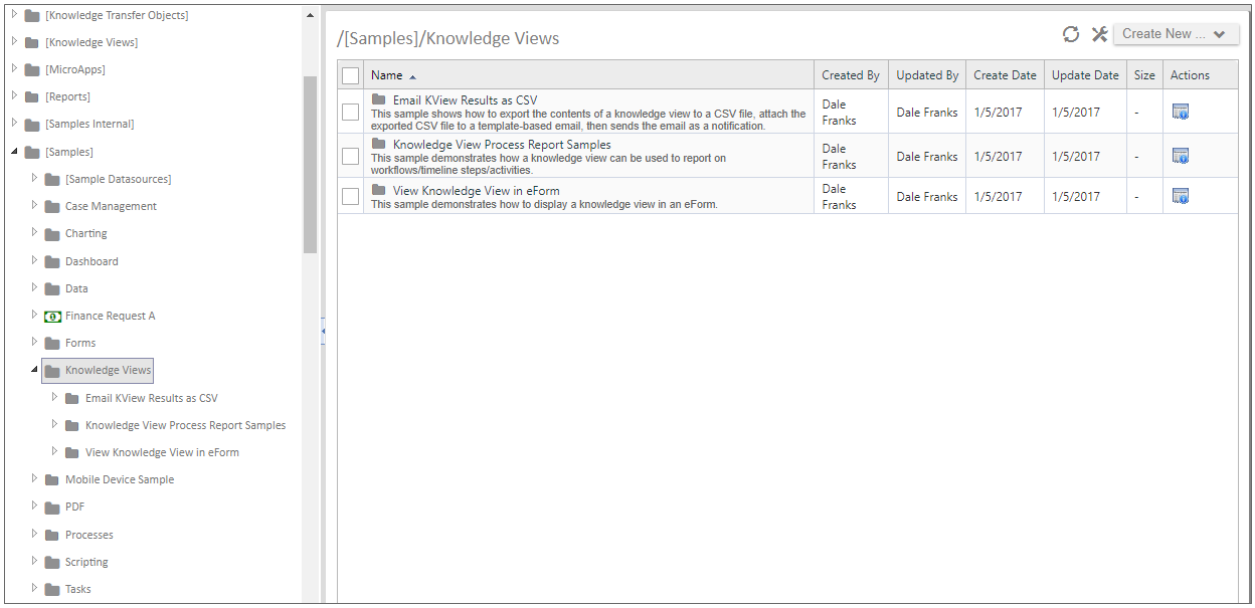
When object locking is required, object definitions are opened in "read-only" mode for all users. Any user who wishes to edit the object definition can only do so by

checking out the object. When the locking user clicks the object's Lock Object button, the definition will display in edit mode for the locking user, and remain in read-only mode for all other users. Again, the lock will remain in effect until the locking user manually clicks the object's Release Lock button to release the lock. Once the locking user clicks the Release Lock button , the object definition, when opened, will display in read-only mode for all users.

If you lock or release a lock on a folder, the operation will be applied to all of the contents of the folder, assuming that you have the right to lock or unlock the child objects. Process Director will silently skip objects the current user doesn't have the right to unlock. Locking, similarly, will preserve any existing locks in the folder that are held by someone other than the current user (that is, they'll still be held by the original lock holder). Folder locking will lock all objects recursively in the folder tree, not just the direct children of the folder. The button text for the lock/unlock button will reflect this difference to locking applied to folders.

Organizing Content List Objects

In general, Content List objects are sorted into folders, much like files in your computer's file system. There is no hard and fast method of organizing the folders, except that folders and subfolders should be organized logically. For instance, one possible method of organizing folders is shown below.

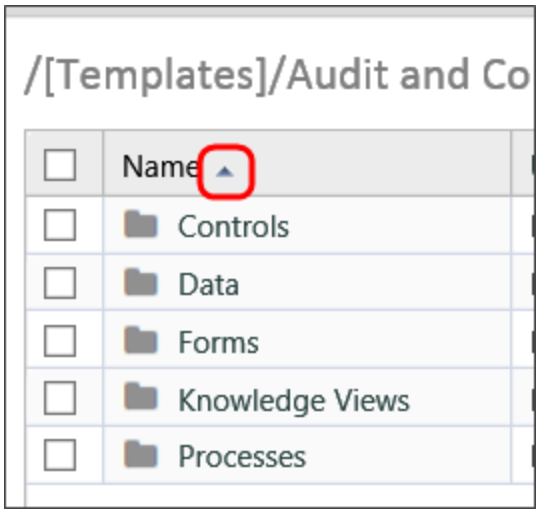


In this particular organization scheme, you can see that the [Samples] folder on the left side of the screen contains subfolders for different Process Director

Objects. In the Knowledge Views folder, you can see the folders are further subdivided into three sample application folders. The Knowledge Views folder is selected, so the Content List on the right side of the example shows all three of the objects in the selected folder. These three folders—though not shown on this example—might further be subdivided into folders named Forms, Processes, Knowledge Views, etc., into which the appropriate Process Director objects would be placed.

You do not, of course, have to copy this particular method of organization, but you should, as a best practice, define some standard method of organizing your Content List so that all implementers are familiar with how an application should be organized. Another general rule of organization—and implementation in general—is that you should use the fewest number of objects necessary to model and automate your processes.

The Content List of the right side of the screen is sortable by column. To sort the items displayed in the Content List, click on the column name. To change the sort from ascending to descending, click on the column name a second time. A small arrow icon will appear next to the column name to show whether you are sorting the column in ascending or descending order.

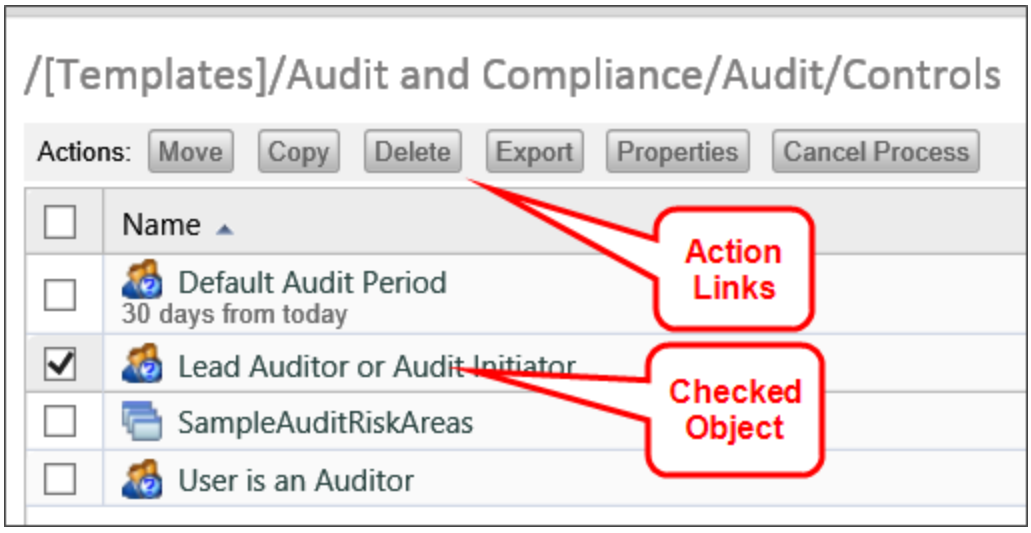


Each time you click on the column header, the sorting function will occur automatically.

Managing Objects

On the left side of each object in the Content List is a check box that can be checked to select one or more objects. When you check one or more of these check

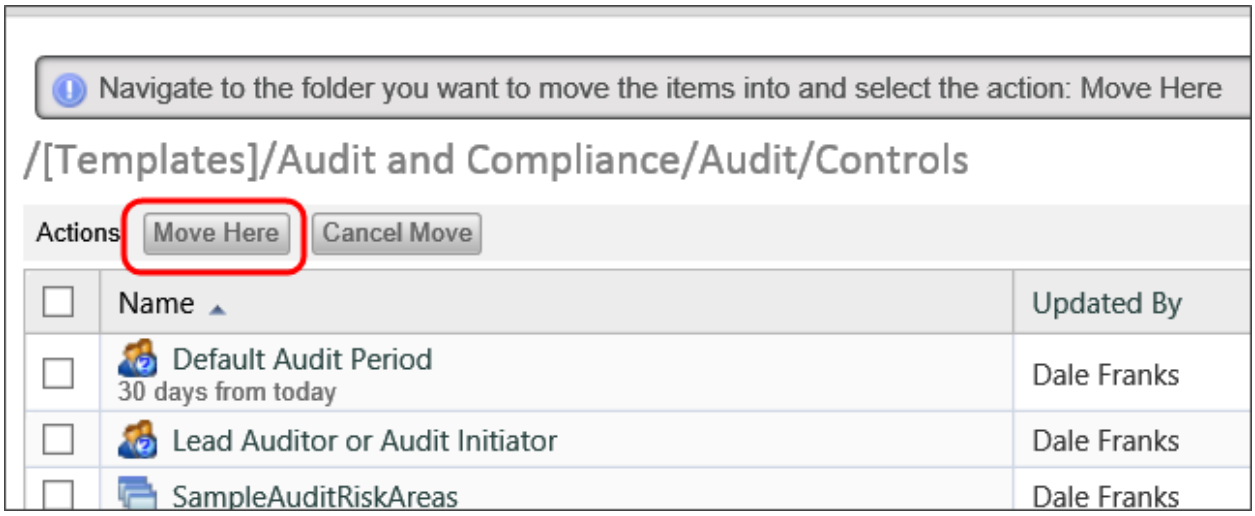
boxes, a series of Action Links will appear at the top of the Content List that enable you to copy, move, etc. the checked objects.



The following action links are available.

Move

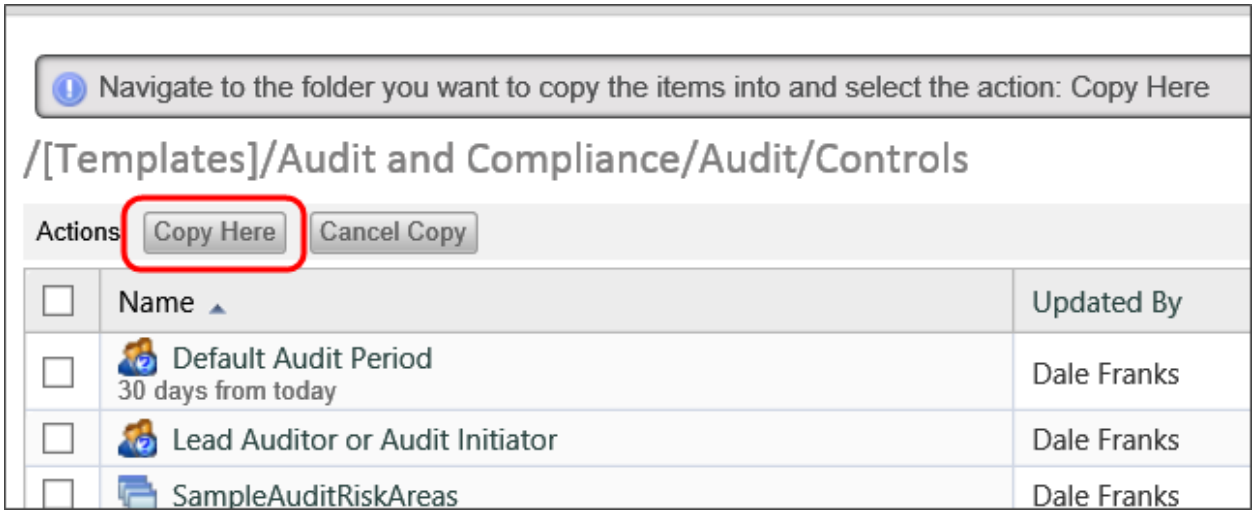
Clicking the Move button will enable you to move the object, along with its complete configuration, to another location in the Content List. When you click the Move button, a new set of action buttons will appear.



Simply navigate to the folder into which you'd like to place the object and click the Move Here button to place the moved object into the desired location. Process Director will move the object from its original location to the new location you have selected. You can cancel the move at any time by clicking the Cancel Move button.

Copy

Clicking the Copy button will copy the basic object definition but ***not*** any of the configured Custom Task event mappings. Just as with the Move button, clicking the Copy button will cause process Director to present you with a new set of action buttons.



Simply navigate to the folder into which you'd like to place the object and click the Copy Here button to place the copied object into the desired location. Process Director will copy the object from its original location and place the copy in the new location you have selected, while leaving the original copy in its place. You can cancel the copy at any time by clicking the Cancel Copy button.

Delete

When you click the Delete button, a confirmation dialog box will be displayed to enable you to confirm that you wish to delete the object. Clicking the OK button on the confirmation dialog box will immediately delete the object.

When you delete a definition object, like a Form or Process Timeline definition, all of the instances of that object will also be completely removed from the system. There is no undo action for deletions!

Export

Clicking the Export button will start the process of exporting the content object. If you have selected a folder to export, all of the folder's contents will be exported,

and the folder/subfolder hierarchy will be exported exactly as it appears in the Content List. See the Importing/Exporting Objects topic for more information about this process.

Properties

Clicking this button will open the properties screen for the selected object. This option will only appear if a single item is selected in the Content List.

Run

Clicking this button will run the object, i.e., will open a Form for editing, or start a process. This option will only appear if a single item is selected in the Content List, and will start a specific action, depending on the object type, as shown below.

OBJECT TYPE	ACTION
Form Definition	Open a new instance of that form.
Process Definition	Launch a new instance of that process
Knowledge View, Dashboard, etc.	Opens the object in display mode.

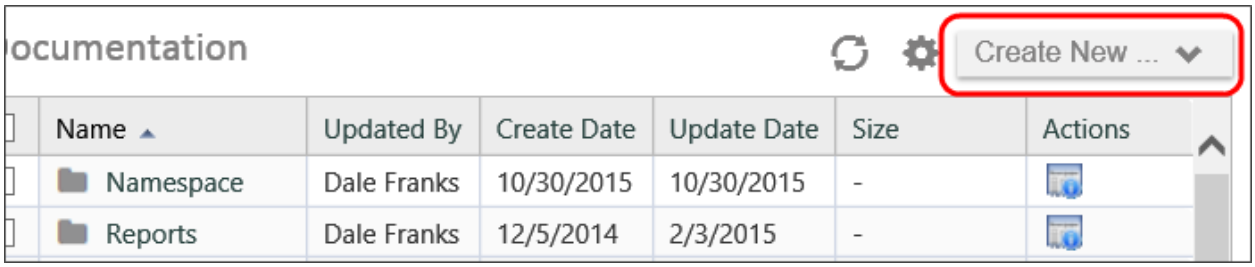
Cancel Process

Clicking this button will cancel a running process.

Creating Folders

Folders provide a hierarchical structure to content in the database. The system supports folders and sub-folders. Each folder can contain any number of objects or sub-folders. A folder can have any name and may contain an optional description. Folders provide the organizational foundation for your data. They can be used to group related items according to Process Timelines, or structured according to user roles.

To create a new folder, use theCreate New dropdown in the Content List and choose the Folder menu item. A user must have Modify permission to the parent folder to create a folder. If a user doesn't have the appropriate permission, the dropdown won't appear.



Managing Documents

Process Director securely stores your documents and files. As an electronic document management system (EDMS), it can store any type of document or file in their native formats. Any file type can be uploaded and managed by the server, including file types such as .PDF, .TXT, .GIF, .JPG, .DOC, .DOCX, .XLS, .XLSX, .PPT, .PPTX, .BMP, .PNG, .ICO, .EPS, .AI, and .TIFF to name a few. Process Director can convert documents or files to a PDF or HTML format; while still maintaining the native file format in the database. Process Director doesn't provide the document authoring tools; instead it utilizes the native authoring tools you are currently using to create and view your files. The type of document can be viewed by clicking on the item in the Content List. The native viewer for the specific document type will be used.

Uploading Documents

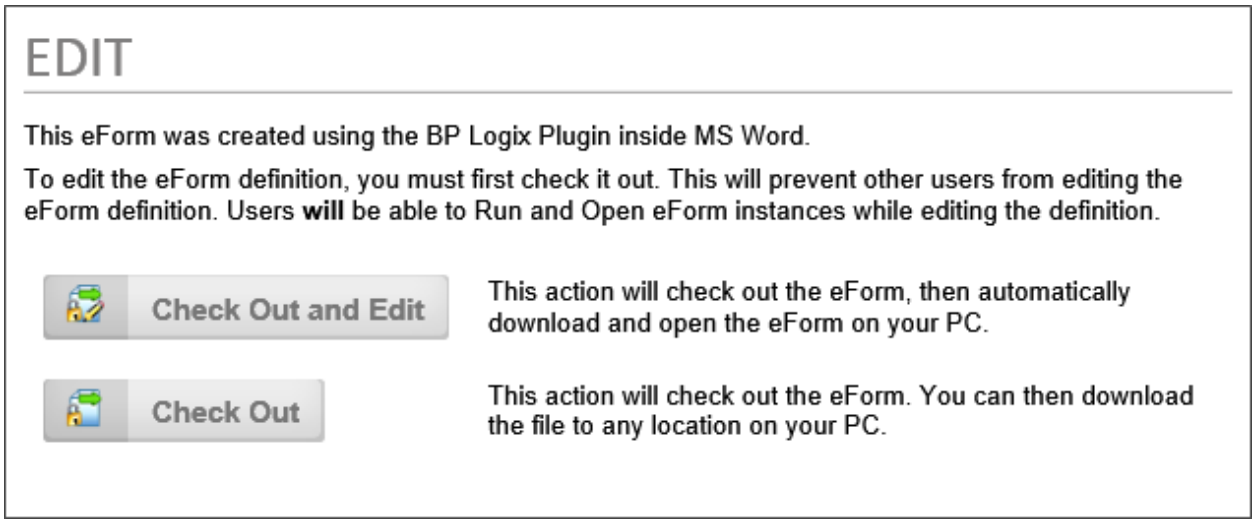
Documents are added to Process Director database by uploading them from your computer. To add a new document, navigate to the appropriate folder on Process Director where the document should be added. Select the Create New dropdown and choose the Document/File entry. A user must have Modify permission to the parent folder to add the new document. If a user doesn't have the appropriate permission, the Create New dropdown item won't appear.

Uploading With the BP Logix Plug-in [Legacy Information]

The BP Logix Plugin uses Microsoft's ActiveX technology with Internet Explorer. ActiveX technology has been deprecated by Microsoft, and Microsoft is phasing out support for Internet Explorer, with Windows 10 targeting 15 June, 2022 as the phase-out date for that operating system. Additionally, ActiveX can't be used with any 64-bit version of Microsoft Office, or recent versions of Office 365. Information about the BP Logix plug-in, therefore, is largely provided for operators of legacy systems

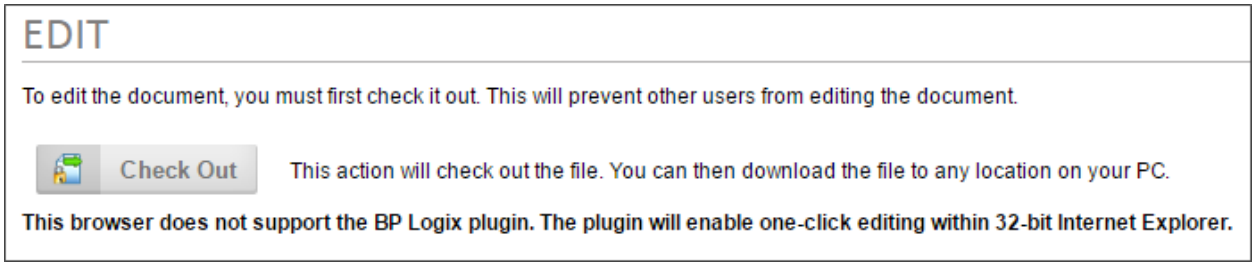
that don't use Microsoft products released after 2019, or any 64-bit product.

The BP Logix Plug-in makes the process of uploading and/or creating documents easier. When uploading a document with the BP Logix Plug-in installed, a button labeled Check Out and Edit will display which will open the editor associated with that file.



Uploading Without the Plug-in

The BP Logix Plug-in isn't required to upload documents. You'll notice on the following screen that the Check Out and Edit button isn't displayed. This will happen when adding a new document without the BP Logix plug-in.



Searching For Content

Process Director supports the ability for users to search through the entire database for any object type using a Knowledge View. The list of objects returned that match the search request will be limited to only those for which the user has View permission.

Searches may be case-sensitive. This is determined by the back-end database system you use. If you are using SQL Server, the default is case-insensitive. If you are using Oracle, the default is case-sensitive.

There is an optional search feature called FTS (full-text search). This feature enables users to search the actual contents of a document for the existence of a word. This support requires SQL Server with FTS enabled. For information on how to enable the full-text indexing on SQL Server, please refer to the Full-Text Search topic in the Administrator's Guide.

Controlling Who Can Create Content

Process Director enables you to control a user's permission to create content in a folder. Administrators can use permissions to determine what type of objects or content can be created by users. You may only want certain types of users to be able to create objects like Process Timeline and Form definitions. See the Permissions topic for detailed information on how to set permissions.

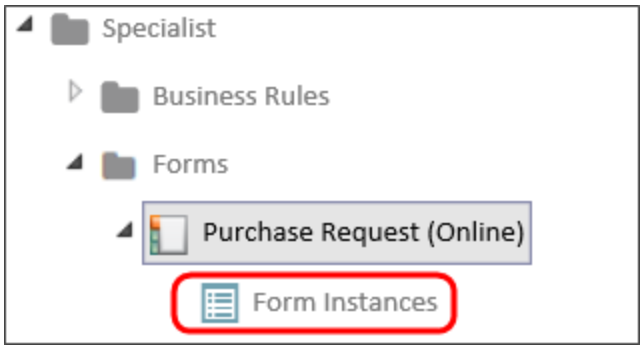
Object Instances

So, far, we've talked about Content List objects in terms of object definitions, but now we need to discuss object instances. The purpose of an object definition is to provide a template for how a form looks, and the data it will contain, or what order of activities a Process Timeline will follow. When a user opens a form for submission, or when a process starts, Process Director creates a copy of the object that will be used for that specific form submission, or that specific iteration of the process. Each of these copies are called an **instance** of the form or process.

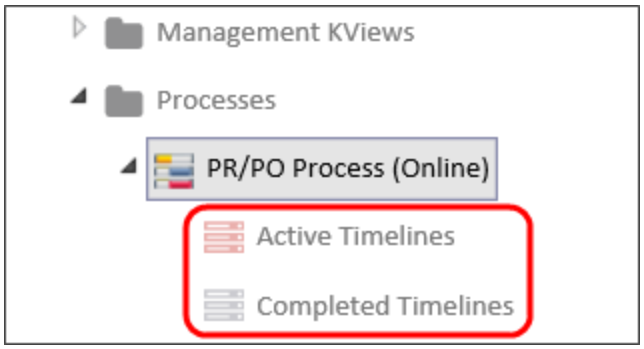
Let's look at a form, for example. The form definition defines what fields will appear on the form, and how the form will look. In other words, the definition specifies they basic structure of all of the form that users will submit. Each individual instance of that form contains all the properties specified for the form, and will also contain all of the data entered into that form for this specific submission. So, each instance of the form contains the data for that specific form submission. Similarly, each instance of a process Timeline contains the basic structure specified in the definition, as well as all of the task start dates, due dates, etc. that occur during that specific run of the process.

All Process Director objects that support instances will display those instances in the Content List, as child objects of the object definition. For instance, a Form

definition will have a node that appears in the Content List when you open the definition, named Form Instances.



Similarly, A Process Timeline will have two nodes that appear in the Content List, Active Timelines (which show all of the Timeline instances that are currently running), and Completed Timelines (which show all of the Timeline instances that have finished).

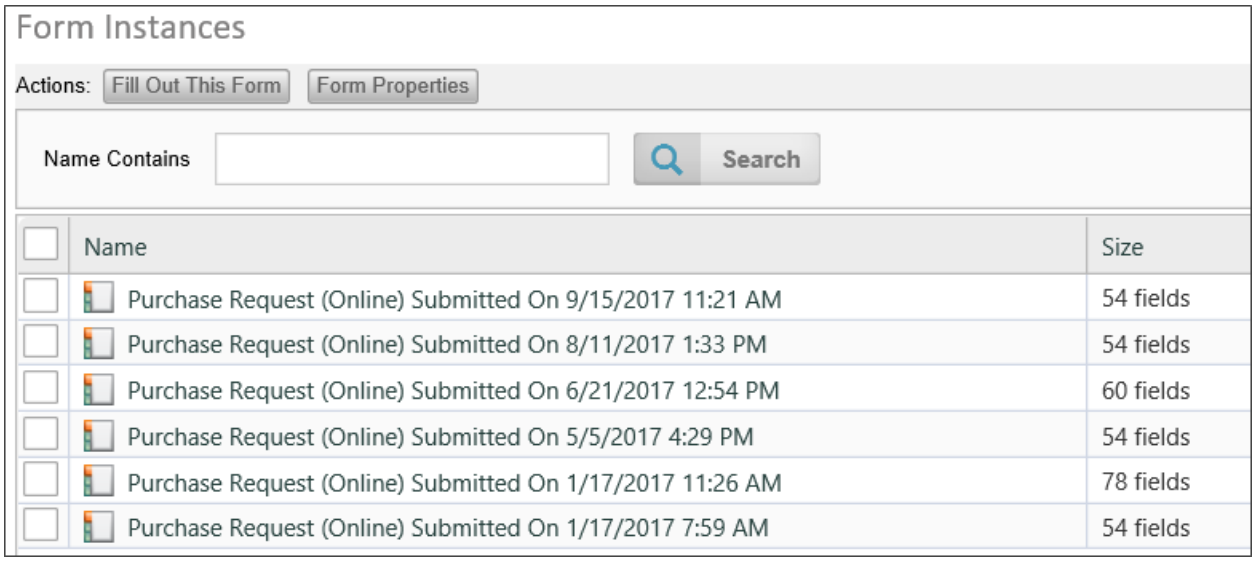


Object instances have their own properties, which are different than the properties of the object definition. Additionally, each type of instance has a set of properties unique to that instance type, e.g., the properties of a form instance are different than the properties of a Timeline instance. We'll describe the Properties of the Form and Timeline instance below.

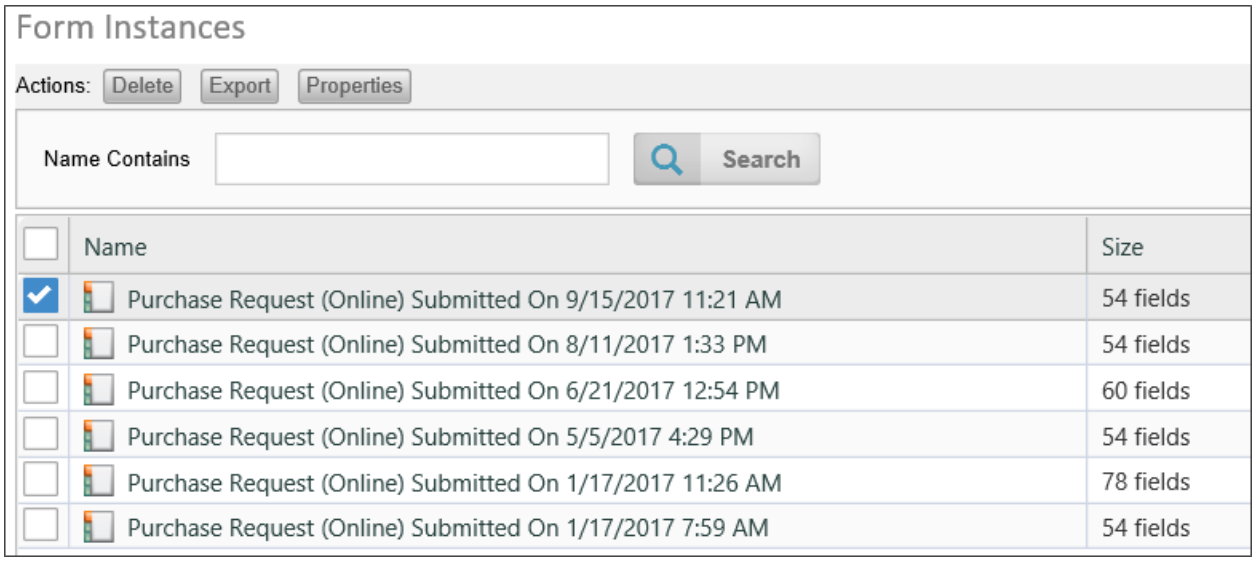
The Object Instances page will display a list of all applicable instances in a Knowledge View. Where appropriate, the Knowledge View will contain a Name search filter and Search button to narrow the list of instances to those whose names match the search filter text.

Form Instances

When you click on the Form Instances node of the Content List, a list of Form instances will appear.



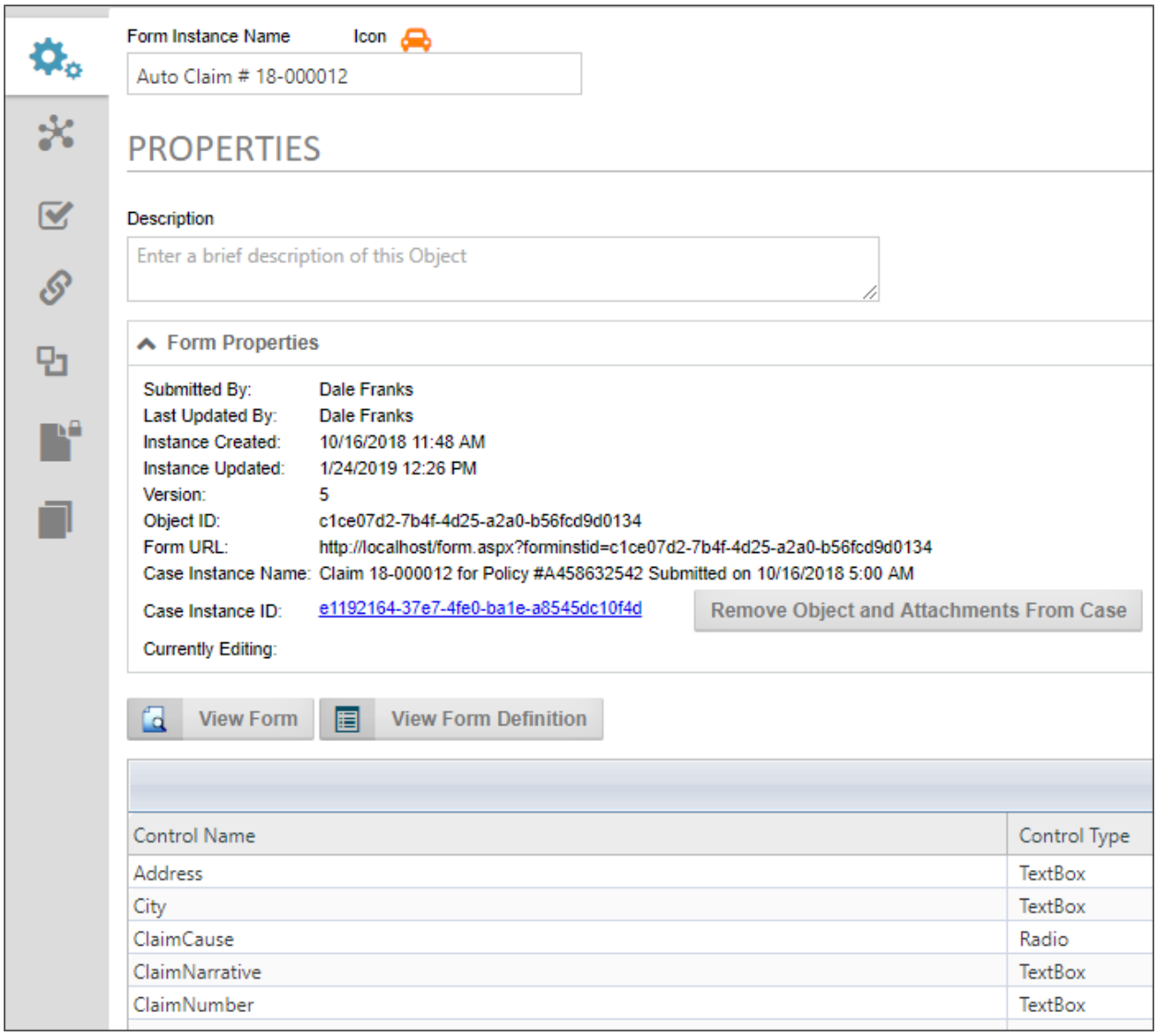
You can use the action buttons above the list of instances to open the form to fill out data by clicking the Fill Out This Form Button, and open the Form definition by clicking the Form Properties button. When you select a check box next to one of the form instances, the action buttons will change.



You can delete the instance by clicking on the Delete button. You can export the Form instance by clicking on the Export button. Exporting form instances is rarely done, and the use case for doing so is extremely limited. It's not a feature that you'd usually use, but there are some cases where a form can be used to provide configuration data for a process, and you may want to export the instance containing the configuration data from the Development environment to Production. Other than that, though, BP Logix doesn't recommend exporting Form instances.

Finally, you can view the properties of the Form instance by clicking the Properties button. When you open the properties screen for the Form instance, you'll see the tabbed interface common to Process Director objects. The following tabs are available for the Form instance.

Properties Tab



The Properties tab displays the Form Instance name and Description, along with a Form Properties section that has some basic information about the form instance, when it was created, it's ID, etc. If the instance is associated with a Case definition, the object and all of its children can be removed from the Case instance by clicking the Remove Object and Attachments from Case button. Below that is a View Form

button to enable you to view the instance in it's current state, and a View Form Definition button to open the form definition.

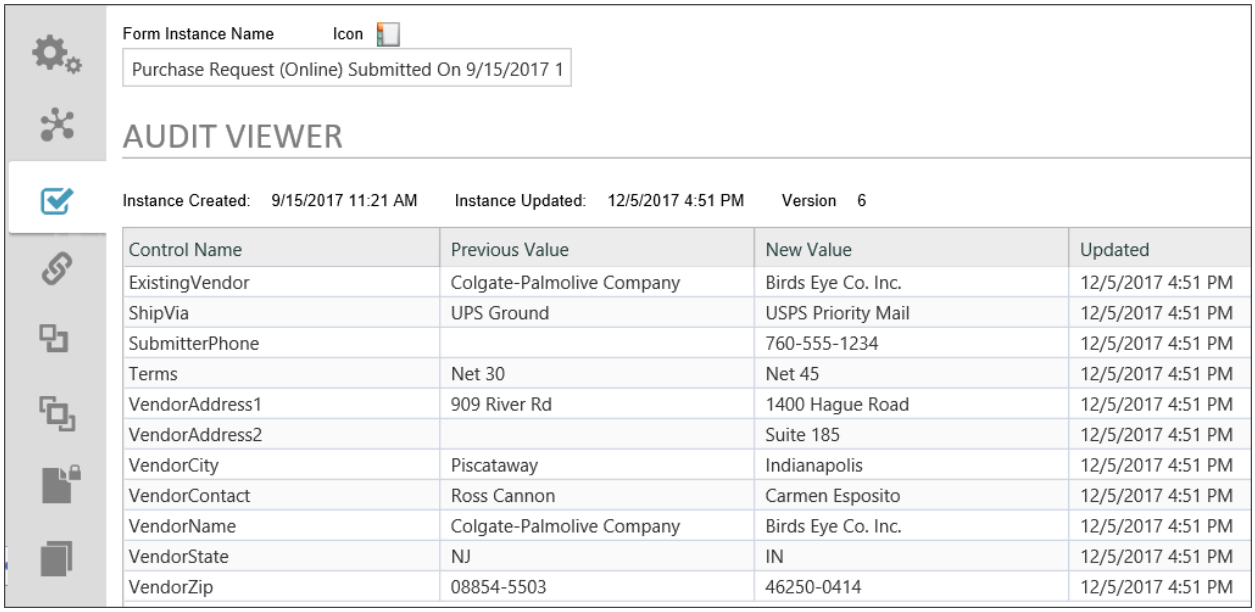
Below that is a list of all the fields on the form, and the current values stored in each field.

Show Parents Tab

The Show Parents tab displays all of the parent process objects, such as running Process Timelines. You can check one of the processes, then click the Properties button to open the Properties screen of the process instance.

Audit Viewer Tab

If the Form definition is configured to audit field changes, the Audit Viewer tab will display all of the field changes that have been made to the form fields, showing the old value, the new value, and the change date and time.

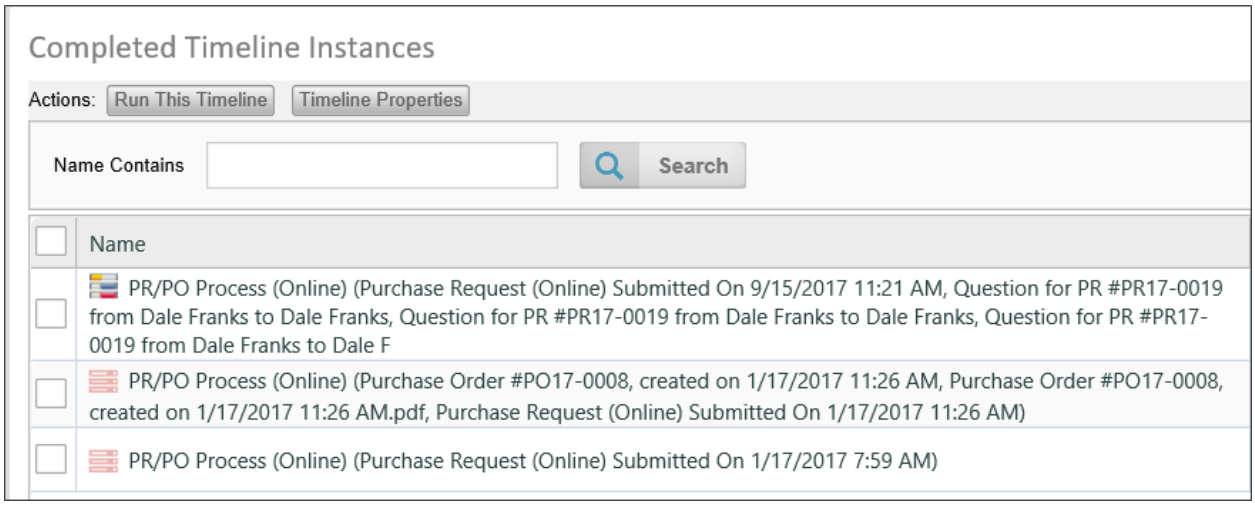


Control Name	Previous Value	New Value	Updated
ExistingVendor	Colgate-Palmolive Company	Birds Eye Co. Inc.	12/5/2017 4:51 PM
ShipVia	UPS Ground	USPS Priority Mail	12/5/2017 4:51 PM
SubmitterPhone		760-555-1234	12/5/2017 4:51 PM
Terms	Net 30	Net 45	12/5/2017 4:51 PM
VendorAddress1	909 River Rd	1400 Hague Road	12/5/2017 4:51 PM
VendorAddress2		Suite 185	12/5/2017 4:51 PM
VendorCity	Piscataway	Indianapolis	12/5/2017 4:51 PM
VendorContact	Ross Cannon	Carmen Esposito	12/5/2017 4:51 PM
VendorName	Colgate-Palmolive Company	Birds Eye Co. Inc.	12/5/2017 4:51 PM
VendorState	NJ	IN	12/5/2017 4:51 PM
VendorZip	08854-5503	46250-0414	12/5/2017 4:51 PM

The remaining tabs of the Form instance are the common tabs used for all objects, such as Meta Data, Permissions, etc.

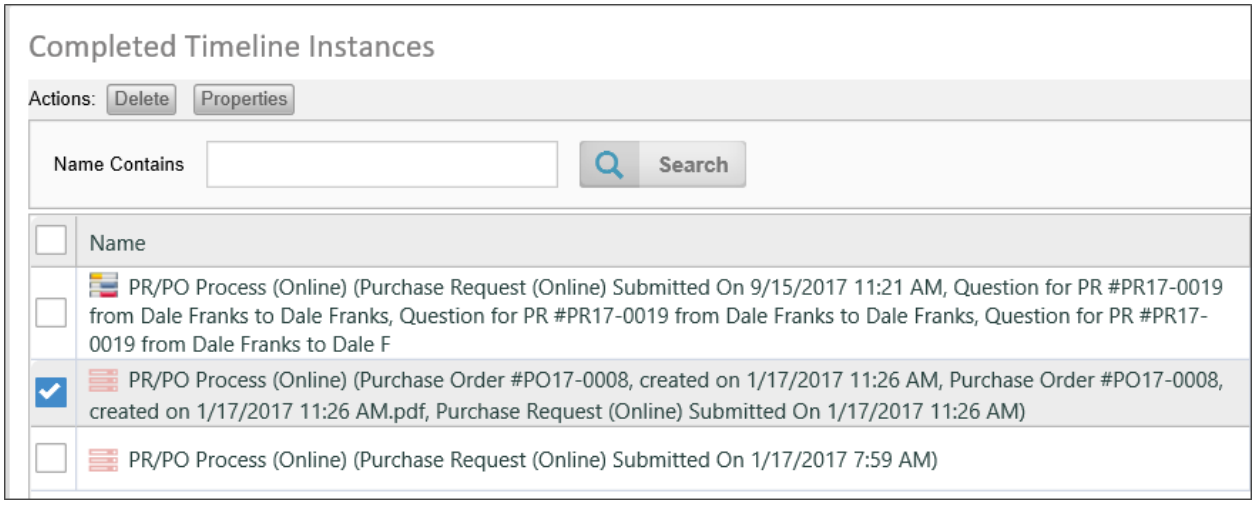
Timeline Instance

As mentioned previously, in the Content list, you can view either the running Timeline instances by clicking the Active Timelines node on the treeview, or you can view Timelines that have finished by clicking the Completed Timelines node. In the example below, you can see a list of completed Timeline instances.



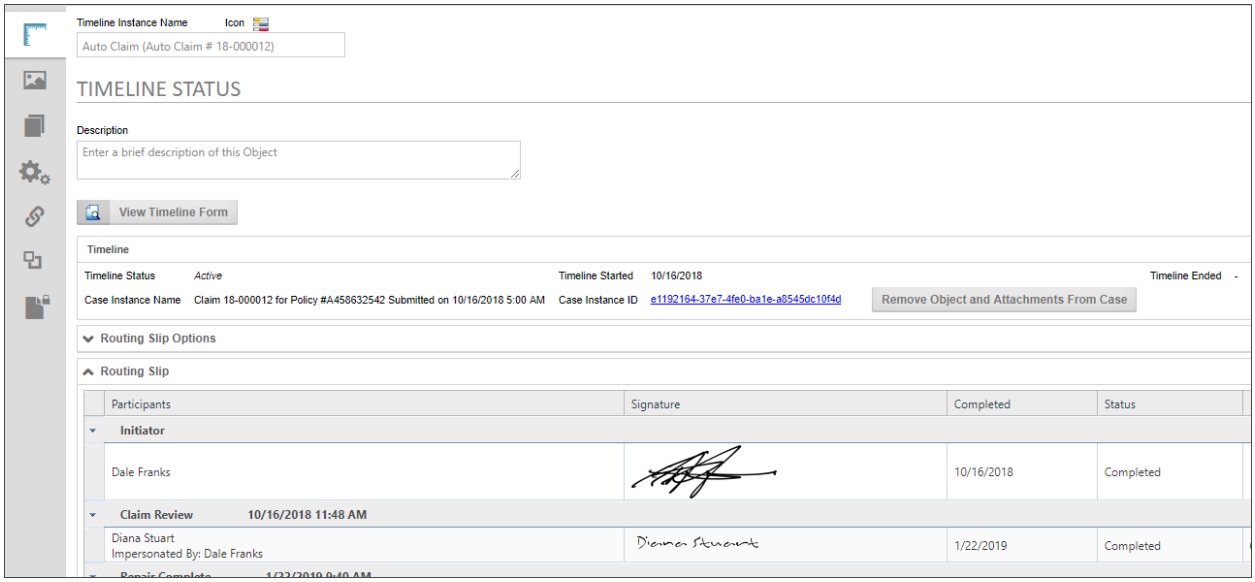
Just as with the Form instances, you can run a new instance of the Timeline by clicking the Run This Timeline button, or you can open the Timeline definition by clicking the Timeline Properties button.

If you select a Timeline instance by checking the box next to the instance name, you see a Delete button to delete the instance, and a Properties button to open the instance properties. A running Process Timeline will have an additional Cancel button that, when clicked, will cancel the running Timeline.



Opening the properties of the Timeline instance will display the regular tabbed interface for the instance properties.

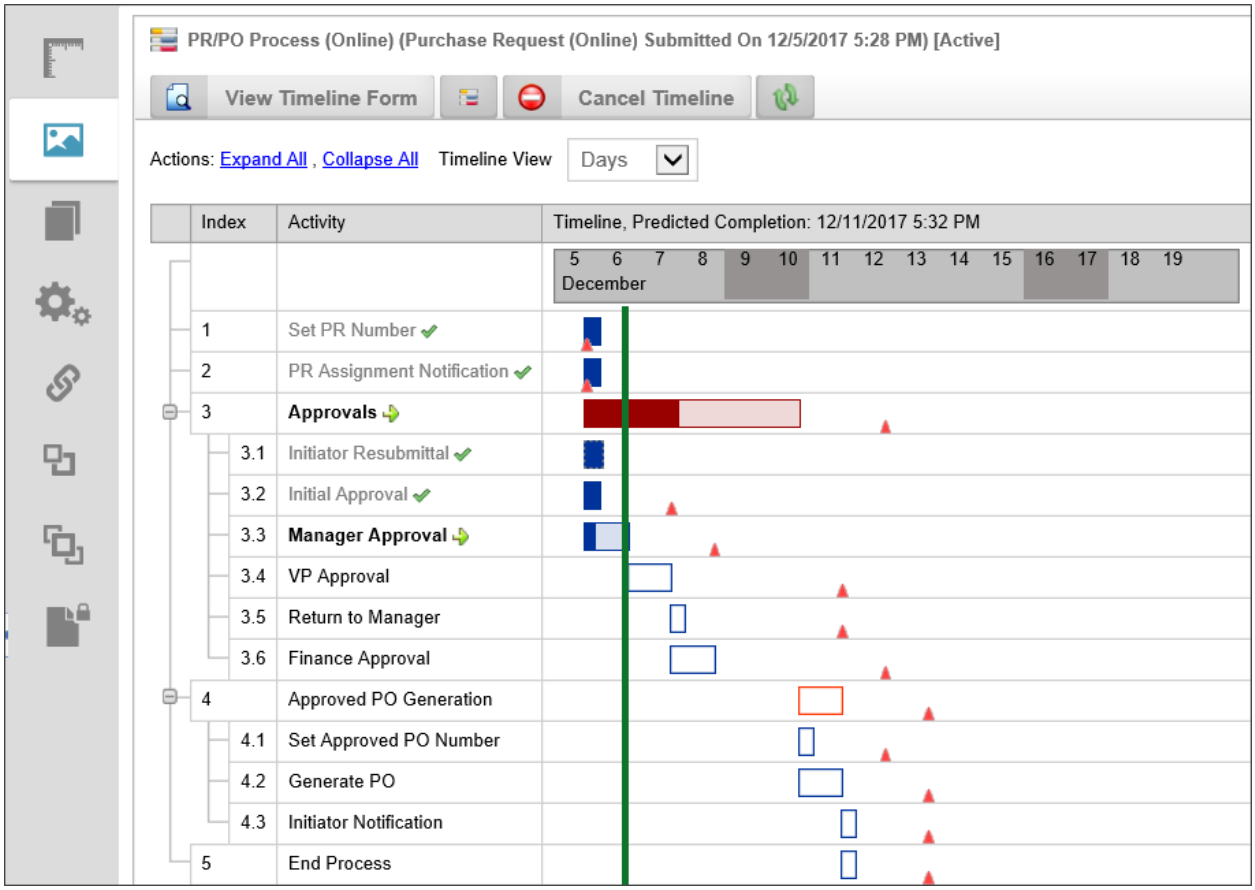
Timeline Status Tab



You can view the Form associated with the Timeline instance by clicking on the View Timeline Form button. If the instance is associated with a Case definition, the object and all of its children can be removed from the Case instance by clicking the Remove Object and Attachments from Case button.

The remainder of this tab displays the status of the Timeline instance, along with a Routing Slip that shows all of the activities that have occurred during the lifetime of the instance.

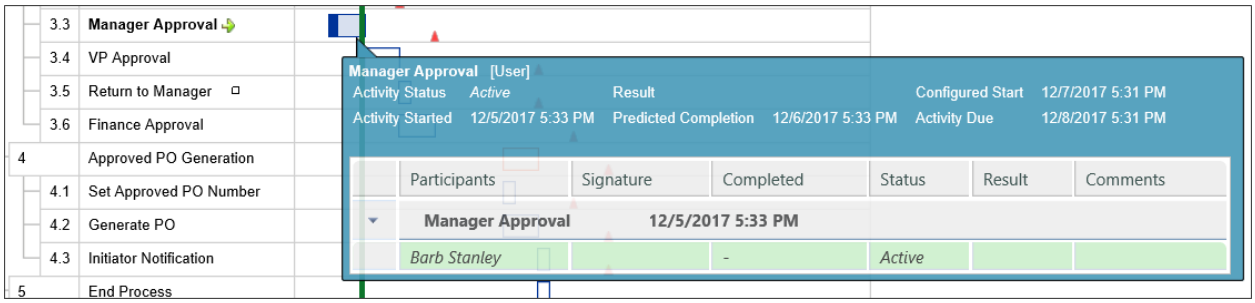
Administration Tab



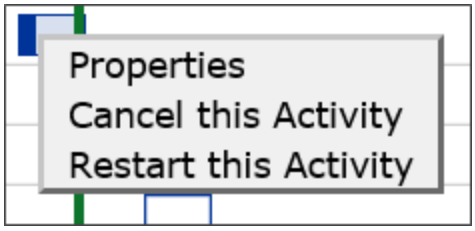
At the top of the Administration tab, there is a series of buttons.

- The View Timeline Form button will open the associated form.
- The View Timeline Definition button will open the parent Timeline definition.
- The Cancel Timeline button will cancel the running Process Timeline.
- The Refresh button will reload the Gantt Chart for the Process Timeline to show its current status.

Each of the Timeline Activities will be displayed in a Gantt chart. Completed activities will be displayed with a green check mark next to the Activity name, while the currently running activities will show a green arrow next to the Activity name. Due dates for each of the activities will be shown by red triangles, which, when moused over, will display the due date information. Each of the bars denoting the Timeline activities will also, when moused over, show the relevant information for the Activity.



Right-clicking one of the Activity bars will open a pop-up menu that will enable you to view the properties of the Activity, cancel it, or restart it.



The remainder of the tabs for the Timeline instance are the common tabs available to all objects.

For more information about managing Timeline instances, please see the Managing Process Timelines topic.

Importing/Exporting Content

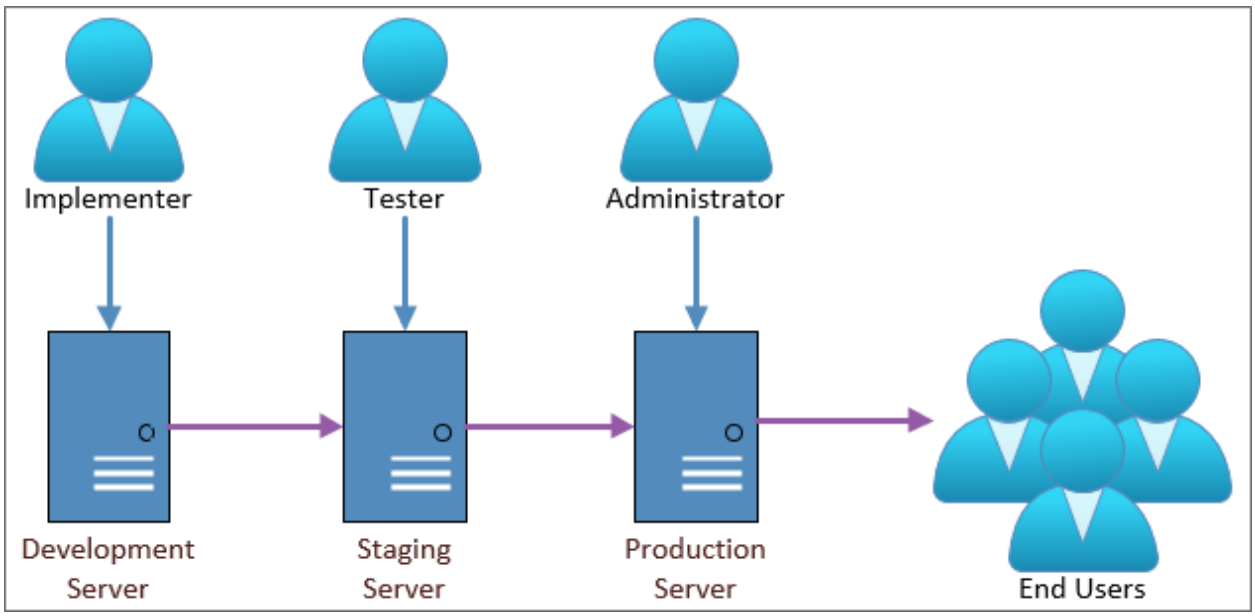
When importing and exporting content, there are a number of vital considerations that need to be kept in mind. While import/export is a routine procedure, it must be done correctly to avoid serious problems. As such, it is important to discuss some best practices—not only in the import and export process itself, but in some basic practices for configuring Process Director objects.

In an ideal environment, there would be three different Process Director installations:

- A **Development** system that is relatively open, and can be used for testing and development in a fairly unrestricted atmosphere.
- A **Staging** system that exactly mirrors the configuration, users, permissions, and Content List structure of the production server. This server should replicate the production environment so that QA testing can be completed in a copy of the production environment before an implementation is moved to production.

Fresh replications of the Production system should be made on the Staging server before any pre-release testing is performed.

- A **Production** system where all of the real-world operations are conducted.



Some Cloud customers may only have two environments, a Development/Staging system where both development and QA testing takes place, and a Production environment where the real-word operations occur.

Content List Folders

In Process Director, Content List Folders have two purposes.

First, they make it easier to establish permissions that allow different classes of users to access different Content List objects. The best practice is to set permissions only on folders, and not on individual Forms, Knowledge Views, Business Rules, etc. Permissions are inherited from parent objects, and when new child objects are created, they automatically have the same set of permissions as the folder in which they are contained. (If you change permissions on the folder, however, the permissions on objects within the folder won't be changed unless you click one of the "Replicate Current Permissions... buttons.)

Different folders might have different permissions to ensure that administrators, process managers, and users are only granted access to objects appropriate to their roles. For example, you might have certain Knowledge Views that only managers can access, while other Knowledge Views can be accessed by any user. In this case, you could create a Manager KViews and a User KViews folder. By setting

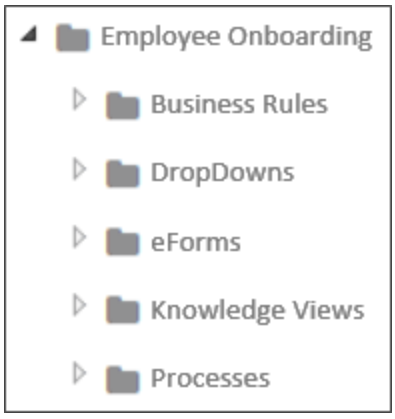
the permissions of the Manager KViews folder to allow only Managers to view it, any Knowledge View that you place in the folder will inherit those permissions, and won't be accessible by other users.

It is important to remember, however, that permissions don't export. The first time you create a folder, manually or via the import process, you must explicitly set its permissions. Once you've done so, however, any new objects created or imported into the folder will automatically inherit the folder's permissions. (Remember to replicate permissions to child objects if the folder already contains objects.)

To use our example of the Manager KViews folder above, if you create new objects in the Manager KViews folder on Staging, then, when you reimport the folder to Production, all of the new objects will immediately be restricted to the permissions you set on the Manager KViews folder in the Production system. Because permissions don't export, you can set much less restrictive permissions on the Manager KViews folder in the Development environment to allow non-managers to test the system with non-sensitive data. When you re-import the folder to Staging and Production, the full set of access restrictions will be automatically applied to all the objects in the folder on the destination systems.

Folder Structure

Structuring your folders properly makes it easier to deploy individual projects or implementations. As mentioned in the Creating Applications topic, the best practice for structuring folders is to organize them by application, and include subfolders for various Content List object types contained in the application. For instance, look at the sample structure below.



The image above is an example of structuring an implementation project logically. The top-level folder provides the name of the application you're modeling, in this

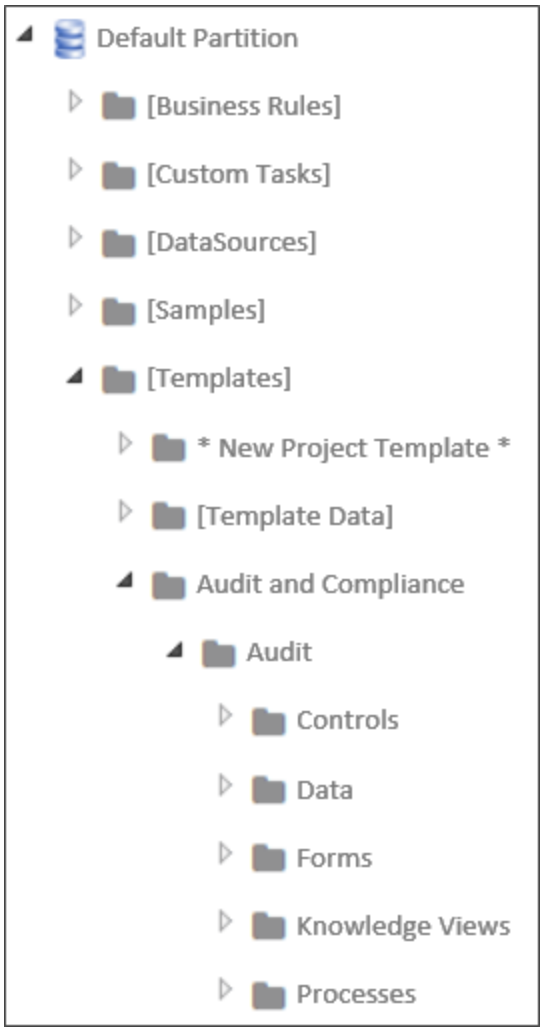
case, a New Employee Onboarding process. All of the individual Forms, Knowledge Views, etc. are organized into the lower-level folders. With this type of organization, any changes you make to the application are contained inside this folder structure, which allows you to simply export the New Employee Onboarding folder, and import it into production, with all of its Content List objects intact, and without having to worry about interfering with any other application.

There are, of course, some exceptions to application-based organization. For instance, shared objects, like Datasources, Goals, and Business Values should probably each be grouped together in their own folder at the root level of the partition. You may have noticed that the Custom Tasks are organized in this fashion. You really shouldn't export/import Datasources between servers, to avoid mixing connections to test data with production data. The Data Sources should have exactly the same name, but the best practice is to create your Production and Staging Datasources manually. The important thing to remember is that you need to keep the folder structure exactly the same on your staging and production server, so that shared objects have the same relative paths on both servers.

Also, remember that any Datasources that are pointing to an external production system may need to be changed on the test server to point to an external test system. The same may be needed for some Custom Tasks that don't use a Datasource but connect to the external application directly, such as the Web Service CT. For the most part, this is important for external systems that are being updated from Process Director. If Process Director is just reading from the external system this may be less of an issue, but something you should keep in mind.

You may also want to create a root level Business Rules folder for those Business Rules that can be used in multiple projects. Business rules that are specific to a single project can be contained in the project's folder structure. Similarly, some common sub-processes might be organized into a root-level folder as well.

With the above in mind, a sample partition might look something like this:



Notice that the root folder contains folders for all of the shared objects, as well as the high-level organization for different types of applications the organization has implemented. At the second level, the HR Process folder contains the individual applications for various Human Resource operations. Finally, at the third level, the New Employee Onboarding folder contains folders for all of the Content List objects that are needed to run that specific application. BP Logix recommends that you implement a similar type of folder structure to make importing/exporting projects (not to mention simply finding things) easier.

Users and Groups

Users and User Groups aren't imported or exported in Process Director when exporting applications (though they can, of course, be imported and exported separately by system administrators, using the procedure described below. So, be sure that any test users or groups to which a project refers in the Development

environment also exist in Staging and Production. If you are syncing users in all three systems from Active Directory, this won't usually be a problem. But there may be cases where you've manually created users in the Development system, and in such cases, the users will have to be manually created on, or exported to, the Staging and Production systems prior to the application.

So, make sure that all users match in your test and production environments. If they do not, and you attempt to import a project with a reference to a user that doesn't exist on the target system, an import error will occur. (Keep in mind, though, that explicit references to individual users within an application's Process Timeline activities aren't a good idea to begin with.)

Speaking of import errors, if an import throws an error or warning...**stop**. Fix the problem and try again. There are several errors that are common when importing, such as the user mismatch we just discussed. Another very common error is forgetting to add a Datasource for a dropdown that uses the Fill Dropdown from DB Custom Task. Commonly, the problem arises when you create a new Datasource on the development server, but neglect to create a matching Datasource on the production system before performing the import. The bottom line is that if you see an import error, your project didn't import properly, so you need to fix the problem and try to import it again.

Another important caution to remember is to avoid manually renaming any Content List object on the target system (such as the Production system) that may ever be updated via import. Doing so can result in no end of problems the next time you perform an import. If you do need to rename an object, you must do that manually on both the destination and source servers prior to doing the import. The development/staging/production objects must all have the same names.

To understand why, we need to delve down a bit into how Process Director tracks objects, and how the import process works. Process Director is database-driven, meaning that every single object in Process Director is stored in a database. The database uniquely identifies every object by assigning it a special identifier known as a **Globally Unique Identifier** (GUID). A **GUID**[1] is a 128 bit, 36 character value,

---

[1]GUIDs generated from random numbers are so large that the probability of the same number being generated randomly twice is negligible. Assuming uniform probability for simplicity, the probability of one duplicate would be about 50% if every person on earth owned 600 million GUIDs. (Wikipedia)

and is a series of **hexadecimal**[1] numbers that are created using algorithms that ensure uniqueness. The GUID is generally represented in the format:

`aa213c6f-a8aa-454f-a04d-30b56fd2e493`

GUIDs don't mean anything to us humans, of course, so we always provide logical names, like "Manager KViews", that we humans can understand when we create objects. But, in almost every case, Process Director never uses the name. It always uses the GUID. Two important exceptions to GUID-based identification are import/-export, and when referencing objects and controls via the system variable syntax, e.g. `{RULE:RuleName}, {:FieldName}`.

But, ***GUIDs can't be exported or imported because they are unique***. An object on the Staging system will have a completely different GUID than the parallel object on the Production system. Since that is so, the import process has to try and match the logical name that we have given the object when performing an import.

So, to return to our Manager KViews folder, if we change the name of the folder on the Production system to Mgr KViews, then the next time we import from Staging, problems will arise. The import process will be looking to match the Manager KViews folder on Staging with the same folder on Production. Because we've changed the name on production, however, the import can't match the text "Mgr KView" to "Manager KView". It will, therefore, ignore the renamed Mgr KViews folder on the Production system, and create a new Manager KViews folder on import. Similar problems arise if you rename the folder on the Staging system before importing it to Production.

The new Manager KViews folder won't have any of the restrictive permissions you placed on the original folder. As far as the Production system is concerned, the newly-imported Manager KViews folder is a completely new folder, with the default permissions that will allow anyone to access it. You now have a security hole in your Production system, while the secured object has been orphaned, with all of the imported links now pointing to an unsecured folder.

---

[1] In mathematics and computing, hexadecimal (also base 16, or hex) is a positional numeral system with a radix, or base, of 16. It uses sixteen distinct symbols, most often the symbols 0–9 to represent values zero to nine, and A, B, C, D, E, F (or alternatively a, b, c, d, e, f) to represent values ten to fifteen. (Wikipedia)

So, it is vitally important to remember that you should only rename objects in the target environment when there is a real need, and if you do, you must manually rename it in the source environment.

Now, at this point, you might be thinking that you could simply delete the orphaned object. But wait, it gets worse.

If there are other objects on the production system that were not part of the import, and that were linked to the Mgr KViews folder when you renamed it, those links still exist. Those object links were not replaced with links to the Manager KView folder. So, now, your production system may have some objects linked to the Mgr KViews folder, while other objects are linked to the unsecured Manager KViews folder.

If you simply try to delete an orphaned object, the remaining links to orphaned object will now be broken, so you may break all of those related applications as well. But, even worse, when you delete an object, you also delete every instance of that object. That might not be a big deal if the object is just a Knowledge View, but if the object is a Process Timeline or Form, then every single instance of that object, and all of the data associated with those instances, will be deleted from the system. You have just deleted all of the historical data in your system for that process or Form. Also, you'll delete any link that any other object has to the object, so may destroy more than one application.

If you find yourself in a situation like the above, where you've renamed an object, performed an import, and now have two objects, you have two options. First, you can call BP Logix Technical Support for assistance, which is your best option. Second, you can try to fix the problem yourself by following the procedure below:

1. Back up your Process Director database for the Production system.
2. Identify the new objects that were just created on the Production system because they were renamed.
3. Remove the new objects from Production.
4. Rename the objects on Production so they match the Staging system.
5. Re-import the objects.

With the above in mind, you may conclude that, as long as you aren't importing or exporting, you can rename or delete objects that don't have instances, like KViews or Business Rules, without running into problems. But, that's not completely true either, because, in addition to importing/exporting, there is one other case where

the name of the object, rather than the GUID, is vitally important. Sometimes, you refer to objects through System Variables. For instance, you might have a condition in a process or Form that relies on the result of a Business Rule where you've used a System Variable to return the result, e.g., `{RULE:RuleName}`. System Variables are name-based, not based on the GUID. That means that if you rename or delete an object that is named in a System Variable, you'll break the object that uses the System variable as a reference.

The general rule, therefore, is that deleting object definition in the Content List of a Production system is an extraordinarily bad idea. If you do so, and eliminate vital historical data, then you have to try to restore your database which is a complicated and risky course of action.

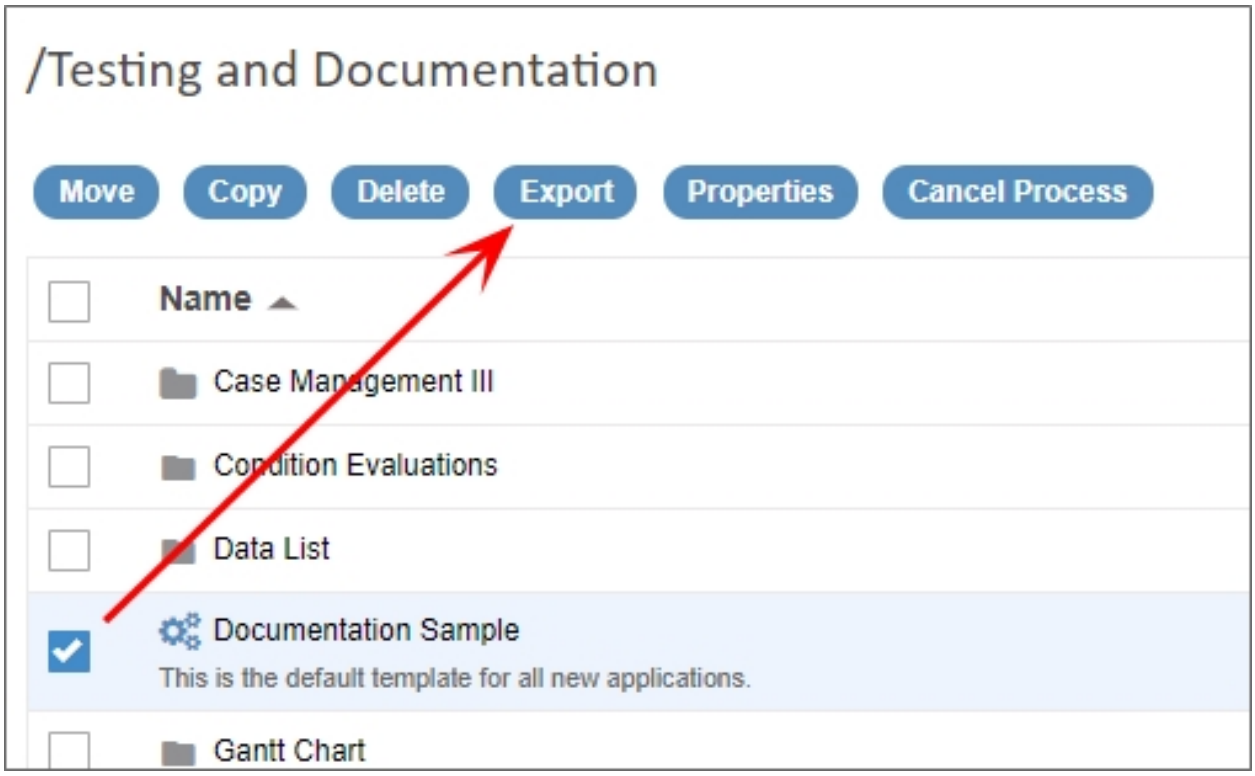
So, to avoid the massive headaches any deletion can cause, only Partition Administrators should be able to delete anything on Production. Moreover, no actual user should ever be given Partition Administrator privileges, in order to ensure that no user can accidentally delete anything.

Instead, create a specific Partition Administrator account, and require administrative personnel to log out of their personal accounts and log into the Partition Administrator account before deleting anything. And even then, they should take extraordinary care before deleting an object. There are, in fact, a number of other security options you should consider as well, all of which can be found in the Securing Process Director section of the System Administrator's Guide.

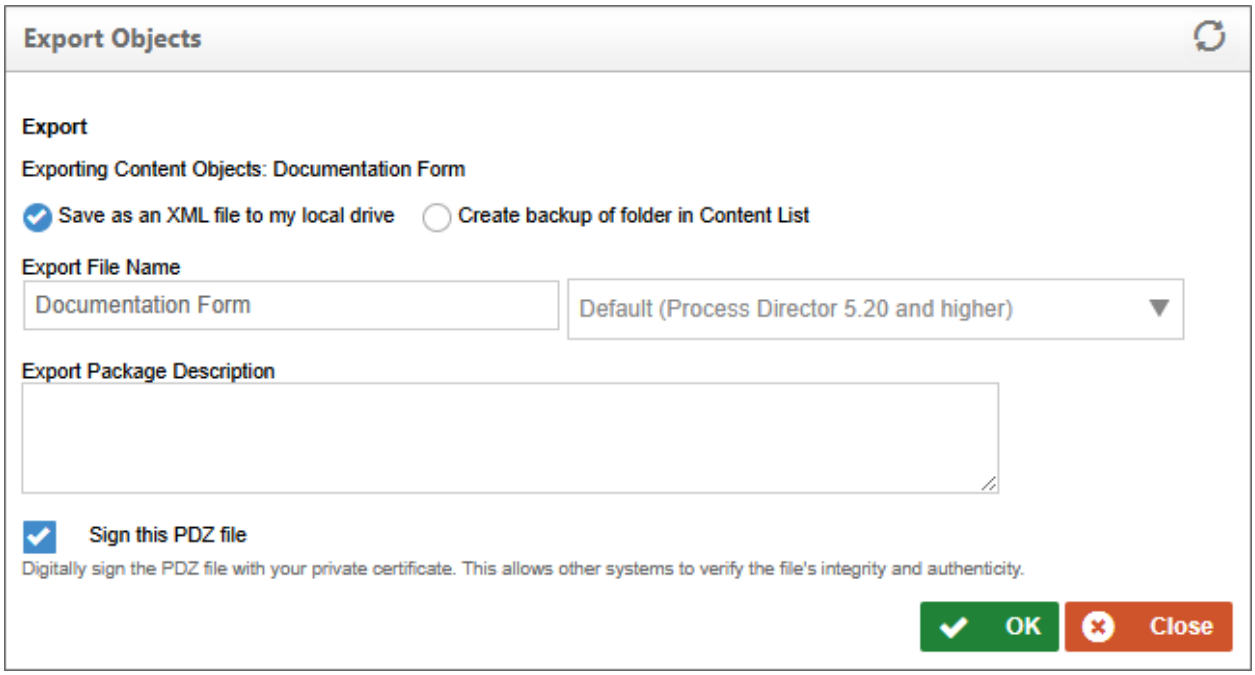
Exporting and Importing Objects

Exporting Content List Objects

Process Director allows groups of objects (folders, documents, process definitions, etc.) to be exported or imported as a single file. This is needed when copying a Content List object/folder/application from a test system to a production server or importing Custom Tasks in to your system. It can also be used to backup entire Process Timelines as you are working on them. To export objects, check off the box next to the names of the files you wish to export, and click the Export action button.



Prior to Process Director v5.25, exports were done in the XML format by default, though v4.55 and higher accepted exports/imports in the ZIP file format, with the ZIP file containing a compressed version of the XML export file. Process Director v5.20 and higher enables exports in the compressed PDZ file format by default. Users have the choice of selecting the ZIP file format for export to v4.55-v5.20, and XML format for all earlier versions of the product. The PDZ file format isn't a special or proprietary format. It 's actually a regular ZIP file, with the file extension simply changed from ZIP to PDZ (Process Director Zip) to indicate that it's a ZIP file containing a Process Director XML export. Renaming a PDZ file extension to ZIP will result in a normal ZIP archive file.



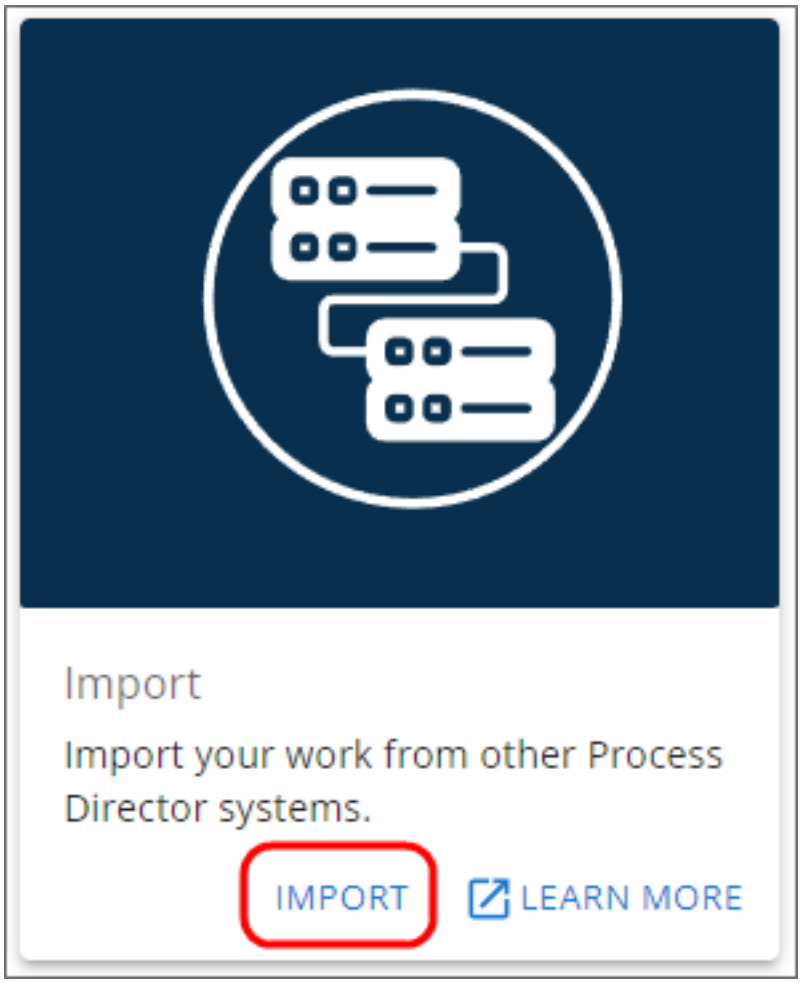
For Process Director v6.1.600 and higher, an additional property, Sign this PDZ file will, when clicked, use the server's private certificate to digitally sign the exported PDZ file. Once a PDZ file is signed, it can only be imported into a Process Director installation that contains the public certificate for the source server's private one. Please refer to the PDZ Certificates topic of the System Administrator's reference for more information on how signing certificates are used.

When you click the Export button, a dialog box will appear than enable you to make changes to the export file name or format. Clicking the OK button will begin the file export, which, for most browsers, will automatically export the file to the default Downloads folder n your local machine.

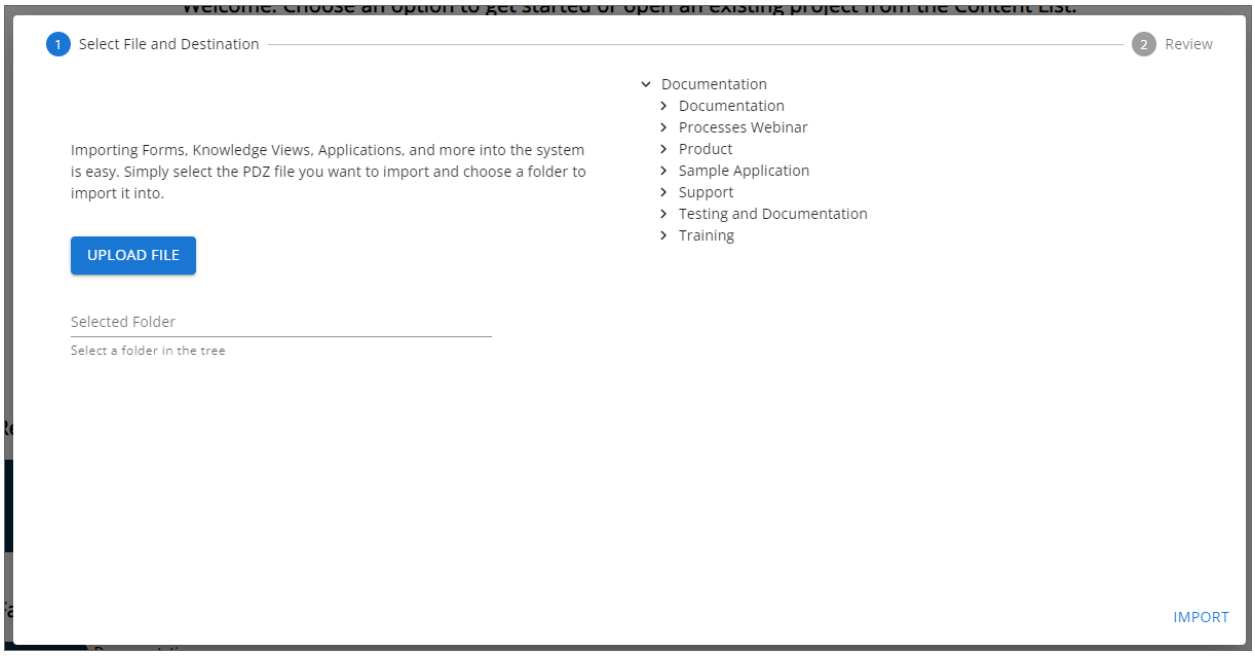
Content List items and Meta Data must be exported separately.

Importing Content List Objects from the Home Page

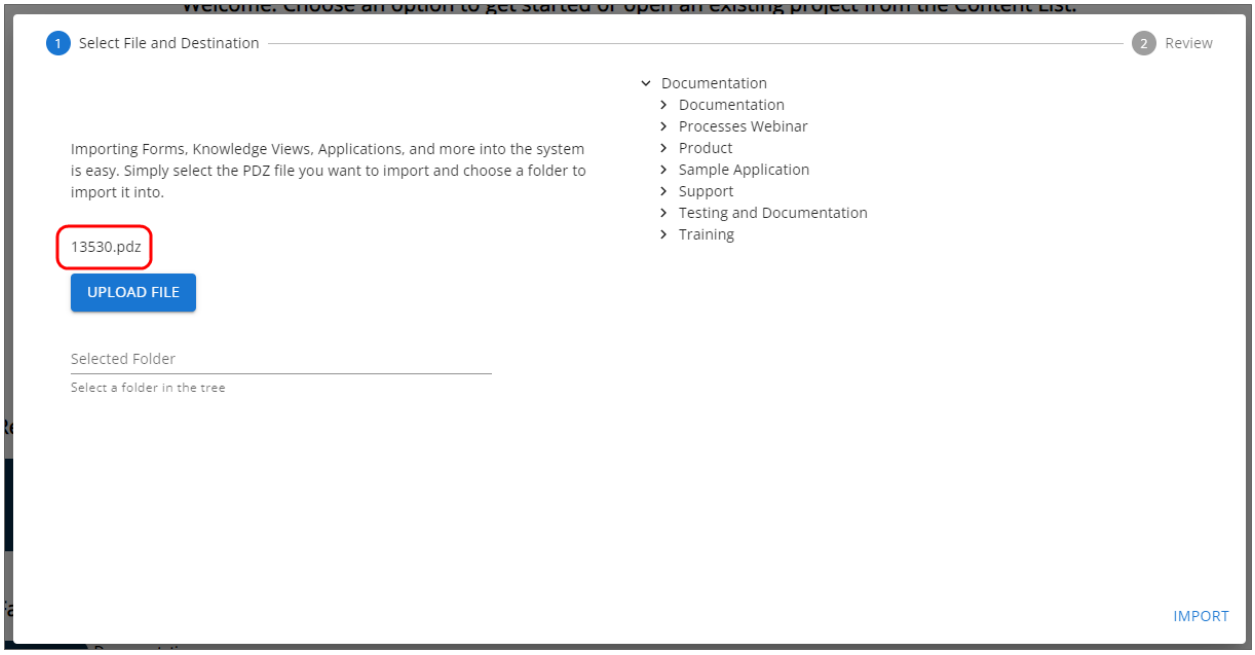
For users of Process Director v 6.0.300 and higher, Process Director objects can be imported onto the server directly from the Home page. A large panel at the top of the home page, labeled Import, displays an Import button.



When clicked, the Import button will open an import wizard that will guide you through the process of importing the PDZ file containing the Content List objects.



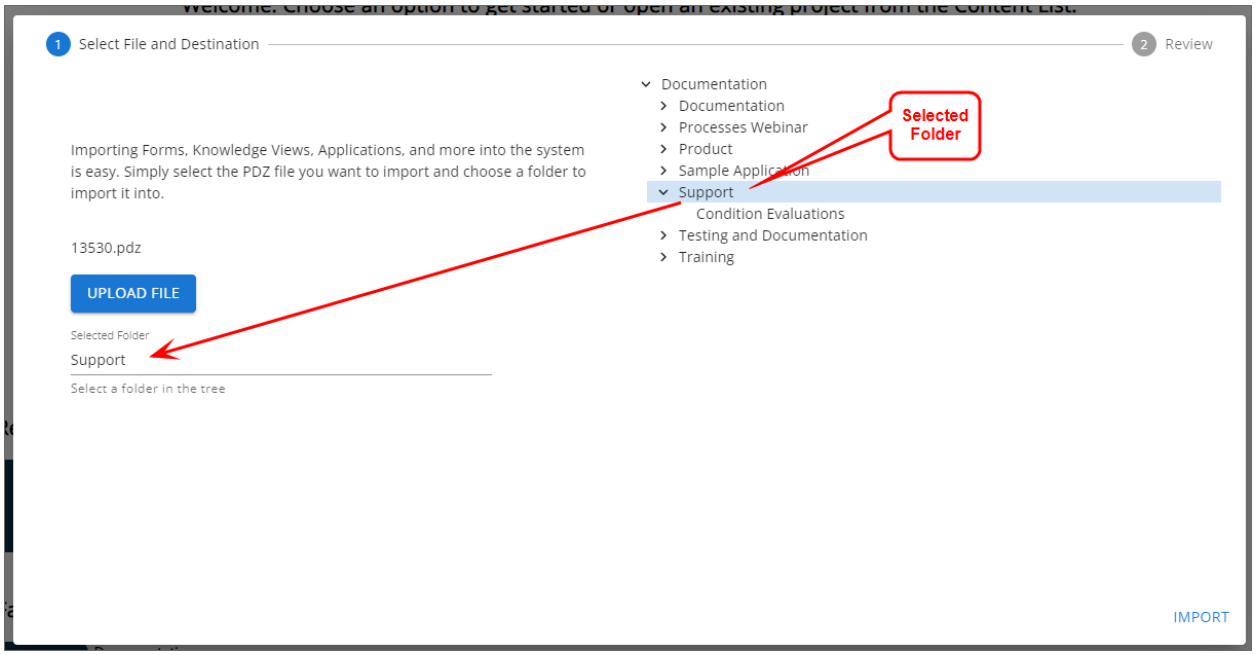
Te begin the import click the Upload File button, which will open the Windows File Open dialog box. You can use this dialog to navigate to the location of your PDZ file, select it, then click the Open button to specify the file to import. Once the file is selected, it will be displayed in the wizard screen above the Import File Button.



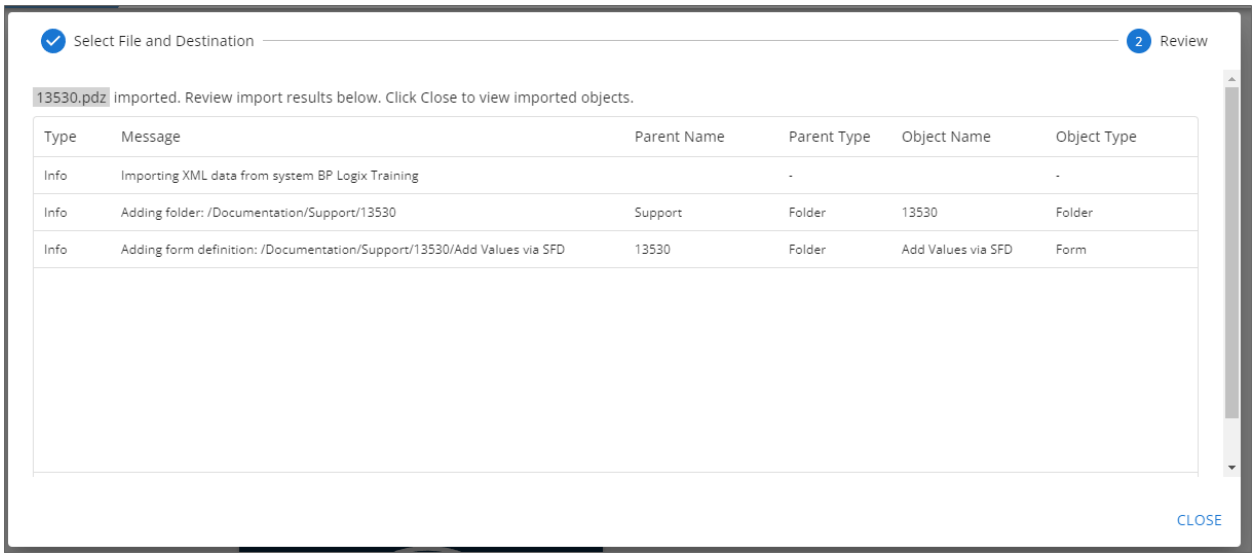
On the right side of the wizard, a list of folders will appear. Using this list navigate to the folder into which you wish to import the object. If the PDZ file contains a single object, the object will be imported into the selected location. For PDZ files

that contain applications inside a single parent folder, the parent folder will be created as a subfolder of the selected Content List folder.

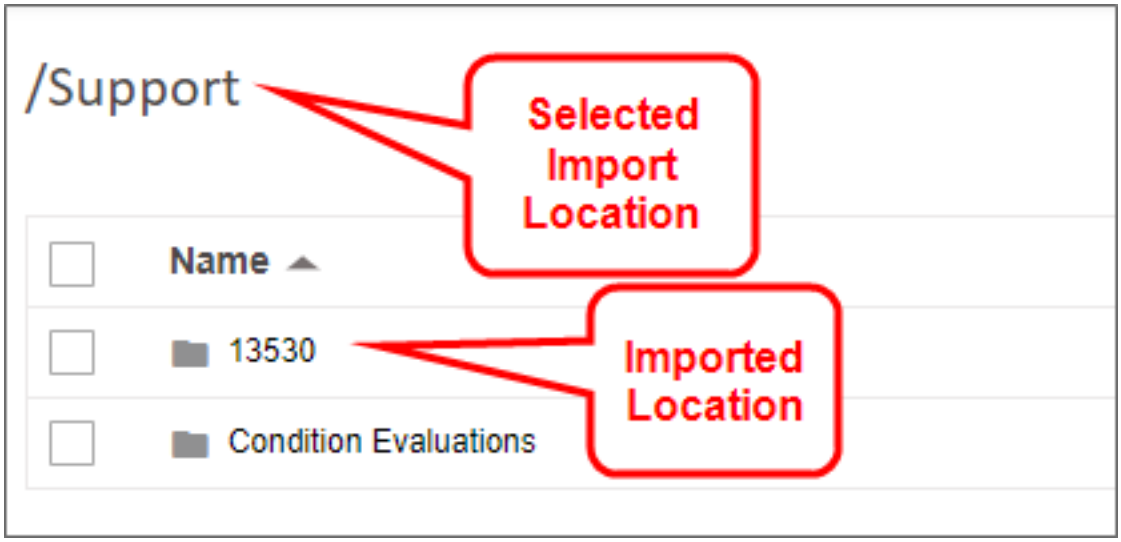
Once you select a folder, it will be highlighted on the tight, and the name of the selected folder will be displayed in the Selected Folder property displayed on the left.



You can now click the Import button to import the PDZ file into the selected folder. a Progress icon will appear while the import process occurs. After a successful import, the Import Report will be displayed in the wizard to enable you to ensure the import ran correctly. This import report may contain warning messages in gray banners, or error messages in yellow banners.

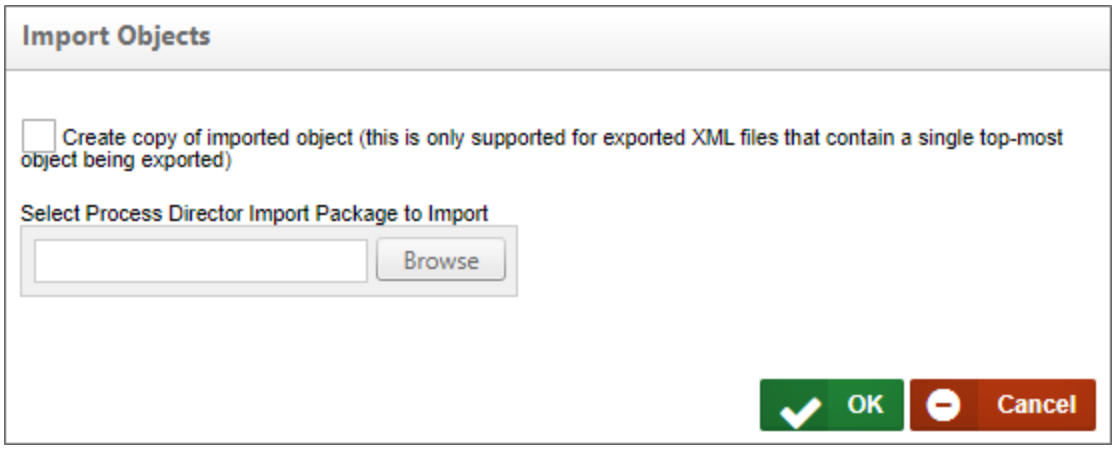


After reviewing the Import report, you can click the Close button to exit the wizard. Once the wizard exits, Process Director will automatically navigate you to the specified import location in the Content List.



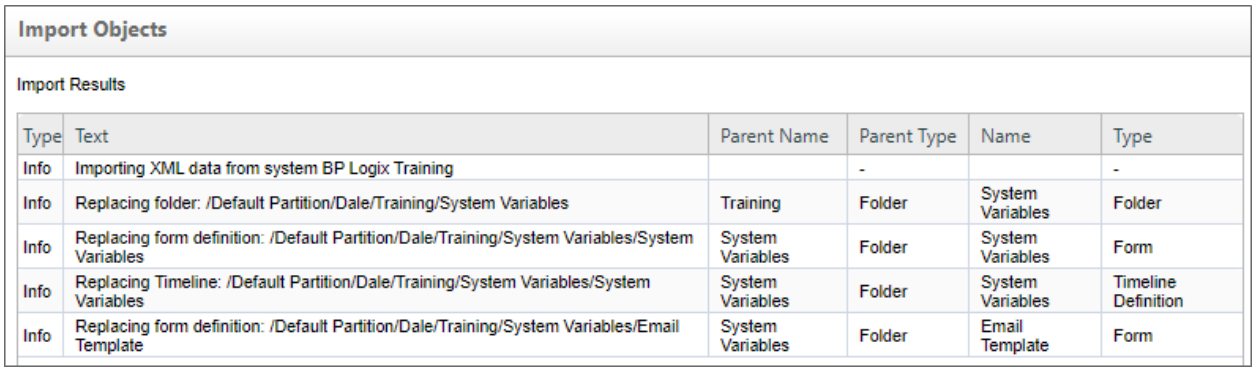
Importing Content List Objects from inside a Folder)

To import a file containing Content List objects, first navigate to the folder in the Content List where you want the imported item(s) to appear. Next, select Import Objects from the Create New dropdown located in the upper right corner of the screen. Selecting this option will open an import dialog box.



From this dialog, you can use the Browse button to browse to the location of the file, and select it. Once selected, clicking the OK button will begin the import process.

When the import process concludes, an import report will appear automatically.



Type	Text	Parent Name	Parent Type	Name	Type
Info	Importing XML data from system BP Logix Training		-		-
Info	Replacing folder: /Default Partition/Dale/Training/System Variables	Training	Folder	System Variables	Folder
Info	Replacing form definition: /Default Partition/Dale/Training/System Variables/System Variables	System Variables	Folder	System Variables	Form
Info	Replacing Timeline: /Default Partition/Dale/Training/System Variables/System Variables	System Variables	Folder	System Variables	Timeline Definition
Info	Replacing form definition: /Default Partition/Dale/Training/System Variables/Email Template	System Variables	Folder	Email Template	Form

You should peruse this report to ensure there are no warnings or errors for the import. If there are, you should correct the errors, then re-import the objects until the errors and/or warnings are resolved.

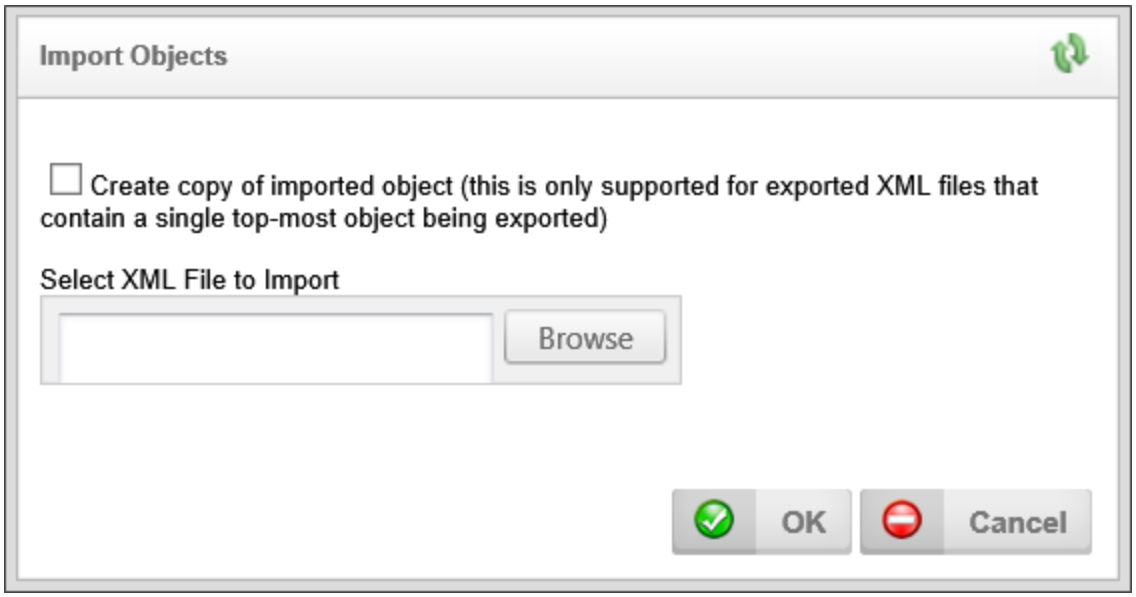
Other Functions

Copying Objects between Content List locations

To create a copy of a folder within the Content List, first export the folder using the process described above. Doing so will create an XML file containing the exported information.

Next, import the XML file that you just created. When the import screen appears, check the box labeled "Create copy of imported object (this is only supported for exported XML files that contain a single top-most object being exported)". This will

import the file as a copy of the existing item. If you don't select the checkbox, the import will overwrite the existing item.

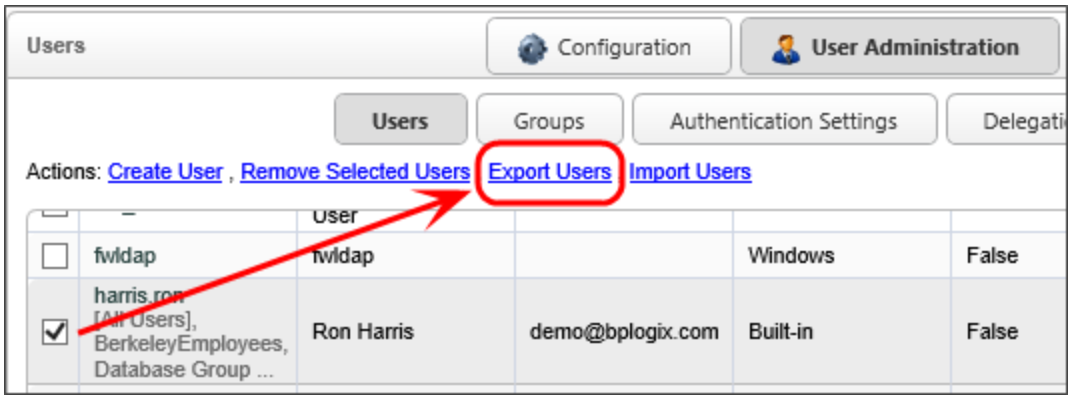


Please note that this "create copy" feature will only work when importing an XML (or PDZ) file that contains either a single folder as a top-level object, or a single Content List object. If you wish to import multiple Content List objects, therefore, they must be exported in a folder, not as a collection of individual objects. The exported folder can then be re-imported as a copy.

As a quick example, if your original folder is called "vacation request", the XML file you create through export will be called "vacation_request.pdz", and the new Content List folder you create through importing the XML file will be called "vacation request copy".

Importing / Exporting Users Using Excel

Users in Process Director can be exported to and imported from a Microsoft Excel spreadsheet. To export Process Director's users to an Excel file, first ensure that you are logged in as an administrator. Navigate to IT Admin > User Administration > Users, and click on the Export Users link. If you only wish to export specific users, you can select those users by clicking the checkboxes next to their names, and then clicking Export Users.



The resulting excel file should contain information about the users on this Process Director installation, and should look like this:

	A	B	C	D	E	F	G	H	I
1	UserID	UserName	EmailAddress	Culture	TimeZone	Disabled	AuthType	LastLogin	Groups
2	Administrator		administrator@bplogix.com			False	BuiltIn	5/17/2013	admin, group2
3	user1		user1@bplogix.com			False	BuiltIn	2/1/2013 2	
4	user2		user2@bplogix.com			False	BuiltIn	2/1/2013 2	

You can also include other columns that contain additional information about the user, such as contact information and information about the user's role in the company.

J	K	L	M	N	O	P	Q	R	S	T	U
Phone	Description	Title	Office	Company	Department	CustomString	CustomNumber	CustomDate	LegalEntity	BusinessUnit	Manager

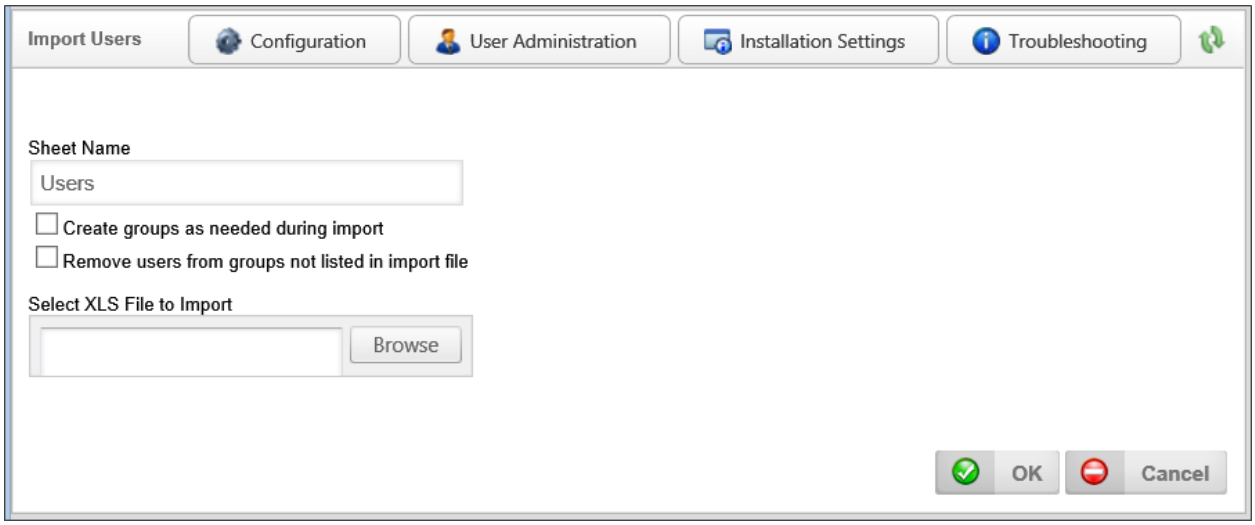
The Manager field can refer to the Manager's User ID, or Process Director's internal oUID for that user.

We can then edit much of the information on the Excel file. We can change the names of the users in the UserName column, or their email addresses in the EmailAddress column. The Culture column specifies which language and time format and number format the user will view, and the TimeZone column specifies the user's time zone, using .NET's list of time zones. The column values for some common time zones are as follows:

- Pacific Standard Time (US and Canada West Coast, GMT-08:00)
- US Eastern Standard Time (United States East Coast, GMT-05:00)
- Greenwich Standard Time (UK, GMT)
- Central Europe Standard Time (Central Europe, GMT+01:00)
- Tokyo Standard Time (Japan, GMT+09:00)
- AUS Eastern Standard Time (Australian East Coast, GMT+10:00)

You also disable users and/or specify the date and time of their most recent log in. AuthType determines how the user authenticates, whether it's through Windows, Process Director, or another method. The Groups column is a comma-separated list of columns to which the group belongs. UserIDs identify the users: they must be unique, as each different UserID specifies a different user.

After creating or editing an Excel file, we can import it into Process Director at the same page from which we exported the Excel file. After navigating to IT Admin > User Administration > Users, click on the Import Users link.



To import users, specify the name of the worksheet in the Excel file that contains information about the users. By default, the name of this worksheet is "Users", and an Excel file created by Process Director will store the user data in a worksheet called "Users".

Checking the "Create groups as needed during import" checkbox will automatically create groups for any group specified in the Excel file that doesn't currently exist in Process Director. If this checkbox is checked, and you add a user to a group that doesn't exist, that group will be created upon import.

Checking the "Remove users from groups not listed in import file" checkbox removes users from groups unless it is specified in the file that they belong to that group. So, if in Process Director I have a user, Administrator, belonging to groups "admin" and "group2", and I import a file that just specifies that the Administrator user belongs to the group "admin", then, when this checkbox is checked, the user Administrator will be removed from group2 on import. Nothing happens to the

group itself, but any user whose membership to that group isn't specified in the Excel file will be removed from that group.

Importing Custom Tasks

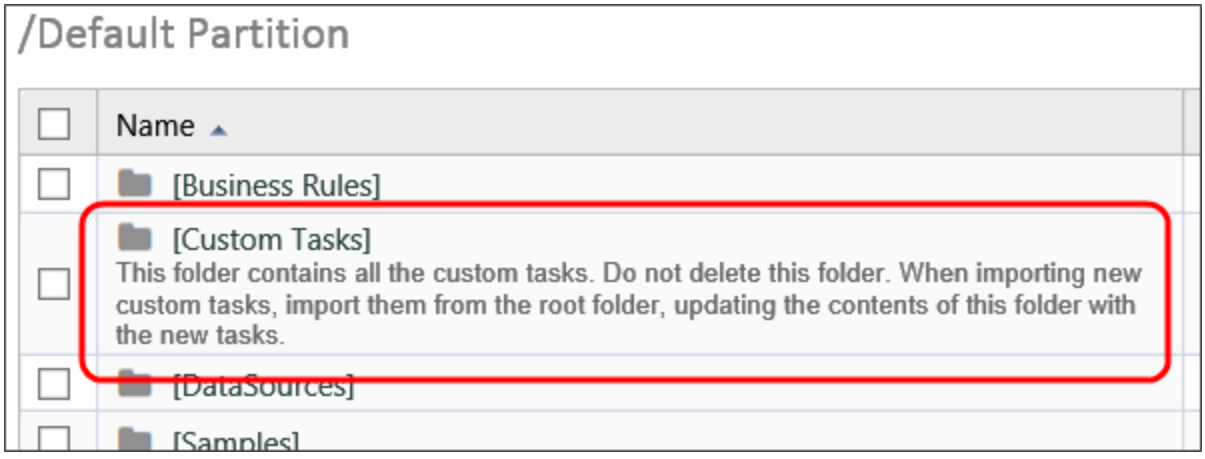
Custom Tasks may need to be periodically updated, as BP Logix updates or adds Custom Tasks on an ongoing basis. Custom Tasks are usually unrelated to any specific version of Process Director, though there are a few exceptions that require a specific product version to work properly. When upgrading the product to a new version, you may wish to upgrade the Custom Tasks after you complete the product upgrade.

To ensure proper installation and upgrade of the Custom Tasks review the following procedures:

Do not delete the current Custom Tasks unless you are sure you want to delete them. Deleting Custom Tasks will remove all Custom Task event mappings to any objects in the partition.

1. To import Custom Tasks, ensure that you aren't inside of the [Custom Tasks] folder, or any of its subfolders, unless you are importing an individual Custom Task to a specific location.

Navigate to the root of the location of the Partition, The [Custom Tasks] folder should always reside in the Partition root, as shown below.:



2. Continue to import the Custom Tasks by selecting Import Objects from the Create New dropdown. Once you've selected the PDZ file to import, click OK.

This will import and overwrite the current [Custom Tasks] folder with the new/updated Custom Tasks.

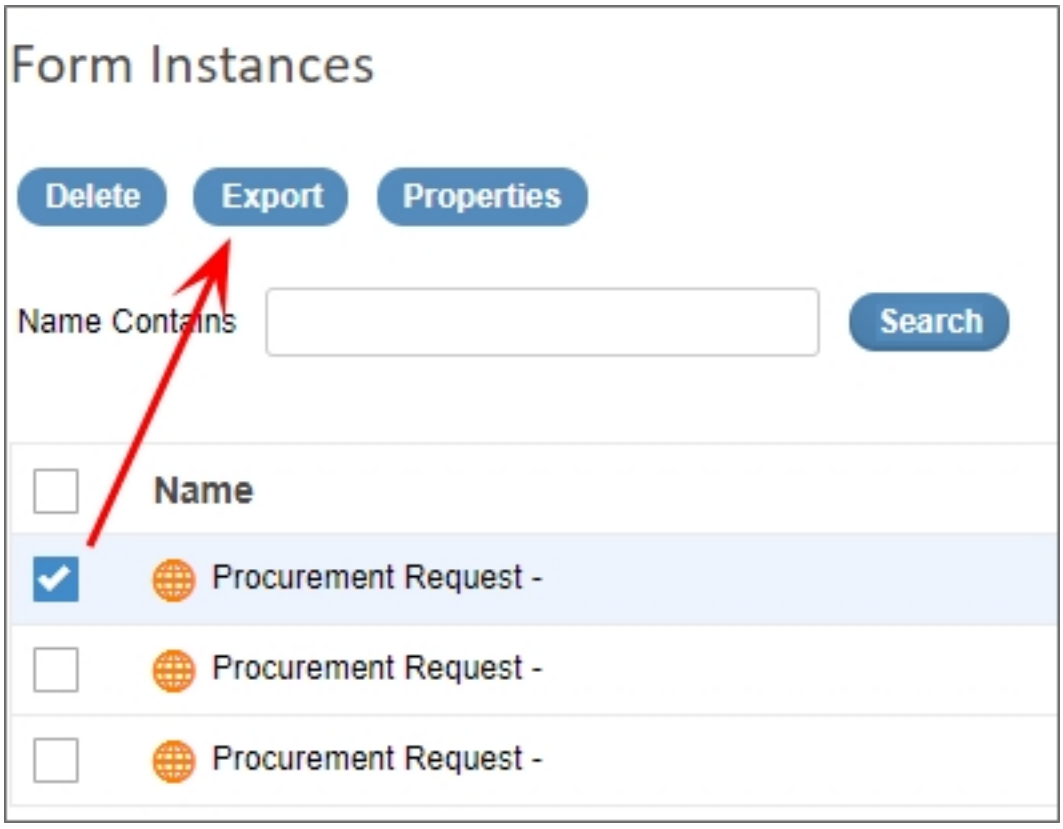
3. Ensure that there are no duplicate [Custom Tasks] folders. If not, then you've successfully updated your Custom Tasks.

Exporting Form Instances

For users of Process Director v4.54 and higher, you can export form instances. This is a feature that enables a relatively rare use case where you need to create Form instances to store configuration data, and you want to export the form instance(s) containing the configuration data between systems.

This feature is *not* designed to export regular form instances between systems, as it does *not* export the linkages to process instances or to any other objects. The export *only* exports the discrete form instances.

In the Content List, when viewing form instances, clicking the check box next to the form instance will show the action buttons, which now include an Export button.



Home Page Import Wizard

For Process Director v6.0.300 and higher, a wizard located on the Home page can be used to import a PDZ file containing Content List objects.

This is a documentation stub for an upcoming Process Director feature, and is subject to change without notice.

Complex Application Imports

When creating a new application, there may be other Process Director objects, like Workspaces and Meta Data categories, created as part of the application. Exporting other Process Director Objects, like Workspaces or Partition Meta Data, is done from the IT Admin area of the installation, on the appropriate page for administering those item types. For example, to export a workspace, you must go to the Workspace page of the Configuration section to perform the export.

The process for exporting these objects is very similar to the method for exporting Content List items. Each object type must be exported separately, and are exported into their own separate PDZ files.

These associated objects will need to be imported separately from the Application folder, and **the order in which these other objects are imported is critical** to avoid import errors. Process Director objects exist in a hierarchy, where lower-level objects can only import correctly if the associated higher-level objects already exist on the system. For instance, if a Form has a Meta Data Category assigned to the Form Definition, you can't import the Form correctly unless that Meta Data category already exists on the target system.

While the order in which you import objects to a target system is important, the order in which you export the objects from the source system is not.

To perform complex import operations like these, the Process Director objects should be imported in the following order.

1. **Partition Meta Data:** Meta data can be applied to any Process Director object, which means it sits at the highest level of the Hierarchy.
2. **Users/Groups:** Any new users or groups that have been created on the source system, and that are used in the application, must be imported. Exports of Groups and Users are, unlike other objects, exported and imported as Excel files rather than PDZ files.
3. **Partition-Level Content List Objects:** If you have created additional Business Values, Dropdown Objects, or Business Rules that reside in the partition's "system" folders (e.g., [Business Values], [Business Rules], etc.), import them to the appropriate place in the Content List before importing the application. BP Logix does **not** recommend importing Datasource objects between systems. Instead, they should be manually created on the Production system, using the same name as they have on the Development/Staging systems. You should have a [Data Sources] folder at the partition root of all systems, which is where all Datasource objects should reside.
4. **Application Content List Objects:** Once the higher-level objects have been imported, you can now import the PDZ file containing the application folder and all its contents.

5. **Workspaces:** Workspaces are usually configured with specific groups or users assigned to them. Similarly, a Workspace may display an application Dashboard or Knowledge View, or have navigation buttons that reference application Forms. Importing Workspaces before importing the required objects will result in an import warning. The Workspace will be imported, but any missing users or groups or application objects will be removed from the Workspace.

Remember, failing to import objects in the proper order will result in import errors that will necessitate re-importing the objects.

Managing Non-Importing Objects

Objects like Forms and Process Timelines are routinely imported into production from development or staging systems. Other objects, however, should not be considered routinely importable. Forms and Process Timelines usually have the same configuration on all systems, which makes them easy to import and export between systems. Some objects, however, might need to be configured quite differently on different installations.

As an example, let's take a look at a Datasource object. On a development or staging system, a datasource might be configured to connect to a test database that is populated with non-sensitive sample data. On a production system, however, the same datasource needs to be configured to connect to the database that stores production data.

On each server, therefore, you would have a Datasource with the same name, and which resides in the same location in the Content List, but each of the Datasource objects might be configured differently. Importing the same Datasource from a different server would overwrite the Datasource's proper configuration.

It's best to treat any object as non-importable if its configuration on a production system will differ from development or staging systems.

The first time you create a non-importing object on a development system, you might, for convenience, export it to staging or production initially. After the import, though, you would need to configure the object properly on each server onto which you import it. Once you have changed the configuration, any further changes to the object should be made manually on each server.

Global Objects

Non-importable objects also tend to be, though are not always, objects that are universally accessible. To return to the example of a Datasource object, each system should only have a single Datasource object that connects to a specific data store. Every Business Value or Custom task that needs to access that specific data store should access the same Datasource object. This datasource, therefore, is a global object that might be used across many applications. Moreover, by only having one global Datasource for each data store, any change you make to the Datasource is immediately reflected across all applications, making management much easier.

Another example might be a Goal. On a development system, a Goal may be configured to run evaluations on a very short time scale for testing purposes. On a production system, however, that Goal may only be evaluated once each day, week or month. Goals are global objects by design, because each Goal sets a universal system state when it is evaluated. So, you would usually not need to have two goals that make the same evaluation.

Similarly, some Business Rules might be global. Most of the time, a Business Rule makes an evaluation of some specific form or timeline data to return a value. This could be considered a "local" Business Rule, because it can't be used outside the context of the specific application in which it is used. On the other hand, a Business Rule might evaluate a Goal result, or a Business Value, and be used in many different applications, because the evaluation it makes is not specific to a Form or Process Timeline, allowing it to be run in any context.

Unlike non-importing objects, global objects may have the same configuration in production as they do on the staging or development servers. In those cases, importing them is not really an issue, but it might be wise to treat them in the same way you treat non-importing objects, in terms of how they are organized in the content list. Custom Tasks are an example of universal objects that you want to import when BP Logix updates the custom tasks. They are universal, but ***not*** non-importing, because you import them on a recurring basis, as updates are made available.

Organization for Non-Importing/Global Objects

Universal and non-importing objects should be organized into folders at the root level of the partition. These folders, and their contents, should be the same on each system. This rule means that if you have a Datasource named "ERP Data"

located in a root folder named [Datasources] on your development server, it should also be placed in the same location on the production server. When you import any objects that use the datasource, Process Director will, during the import process, expect to find /[Datasources]/ERP Data in the Content List. If it does not, the system will generate an import error.

The most common example of this organizational method can be seen by looking at the [Custom Tasks] folder. The majority of Process Director applications use at least one Custom Task, either in a Form on in a Process Timeline. Because the Custom Tasks have the same name, and are found in the same folder location on every server, importing applications that reference Custom Tasks are generally trouble free.

To emulate this practice, you might want to consider following the organizational method that we use at BP Logix, and creating the following root folders, complete with the square brackets in the name:

[Custom Tasks]

[Data Sources]

[Goals]

[Business Rules]

[Business Values]

Using the square brackets as part of the name means that they will be organized together in the Content List, in both the tree view and folder view. Moreover, they can be shown or hidden in the content list by clicking the Show/Hide System Folders icon at the top of the Content List. Organizing objects this way in the Content List makes them easy to find, both for the import process, and for your implementers when they need to make configuration changes. And of course, they can be hidden from the Content List when convenient.

Template Library

For customers who have a Cloud or Subscription license for Process Director v5.31 and higher, the Template Library enables you to easily replicate an existing Template as a new application.

Many organizations have specific design guidelines that they would like to see replicated in any application. For instance, a specified logo, color, or font might be

required to display on each form exposed to end users, along with some commonly required Form fields. Additionally, system administrators might want to provide one or more "shell" applications with an associated Form and Process Timeline already provided for each new application, simply to save on the time and tedium of creating the same objects from scratch. Templates enable you to achieve those objectives easily. Please see the Creating an Application Template topic for more information on how to create a template for this purpose.

A **Template** is a pre-made application, containing all of the design features and functionality you desire, to use as the basis of a new object. The Template can be as simple as a single form with some font and color styling, or as complicated as a fully functional application that you need to replicate. Any Process Director object, or collection of objects, can be used as a Template.

Once you have created a template, the Template Library feature enables you to select that template as the starting point for a new application by copying the existing template and placing the copy of it in any desired location on your Process Director installation. All of the existing configuration options of the template objects will be replicated in the new application.

Editing Templates

Another benefit of the template library is that you can customize any of the template applications. All changes you make to a template application will be reproduced automatically every time you use the template to create a new application. For instance, you might edit the forms in the different template applications to change your organization's logo or colors in the Form's design. When you create a new application from the template, your forms will reflect the form design changes you made.

Keep in mind that changing the form fields may affect how the application works, so you need to keep the application's architecture in mind. A field linked to a Dropdown object, if deleted, may require that the Dropdown object also be deleted as unnecessary. Similarly, a field evaluated by a Business Rule, if removed, will make the Business Rule non-functional and, in turn, cause any Needed When condition on a Process Timeline activity that uses that rule to fail when the Process Timeline runs.

If the application contains any Business Rules, you may need to alter them to reflect changes you've made to the Form or Process Timeline. You'll want to check

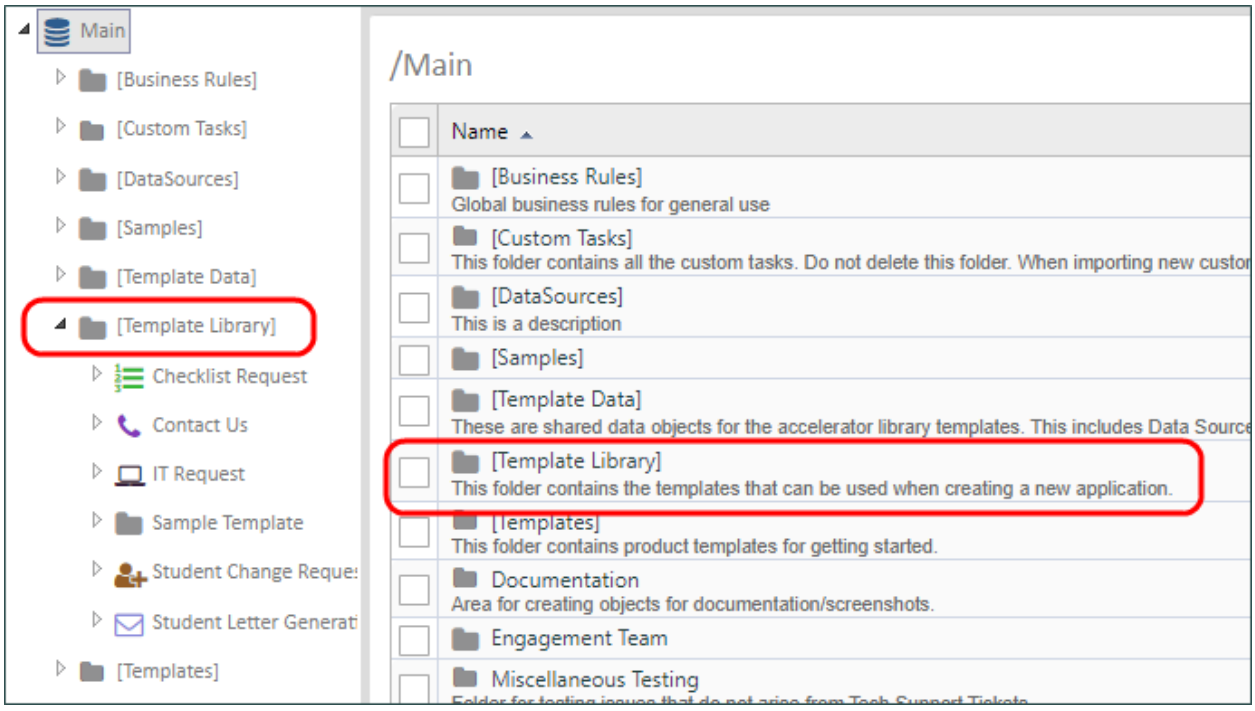
the Email Template to see what information is being sent in notifications and make whatever adjustments you find necessary. Check the Knowledge Views after making form changes. If you removed or added Form fields, you'll want to ensure that the Knowledge View is returning valid form fields and add any additional fields you'd like to display. Finally, check any Dropdown, Datasource, or Business Value objects to ensure the data to which they are connecting or returning is valid for your application.

Similarly, adding new activities or removing existing ones may affect the application's architecture. If you change the verbiage of User activity results, that may also affect the Looping or End Timeline activity conditions. Understanding the application's architecture before making changes or customizations will help you avoid creating unexpected behaviors, because you'll understand the implications of your changes before you make them.

Changing the Name of a Business Rule or other Process Director object will not affect the operation of the application. Process Director tracks object identities through an ID value, and the name is updated automatically in linked objects if you change it.

The [Template Library] Folder

For the Template Library feature to work properly, you must have a [Template Library] folder at the root level of the partition.



If you don't have this folder installed by default, you can simply create it at the root level of the partition. Once you have done so, you now have a template container folder that can be used for that partition. If you have multiple partitions on your installation, each partition will require its own [Template Library] folder.

For v6.1.100 and higher, installations without a [Template Library] Folder will display an error message to that effect on Staging and Development systems.

Inside the [Template Library] folder, each template application must reside in its own subfolder. If you do not have any template applications inside this folder, you will not, obviously, be able to replicate the template to create a new application. So, in addition to having the [Template Library] folder, you must have one or more application templates that reside inside the [Template Library] folder. Installing the BP Logix templates will, as mentioned, provide several template application for use. If, on the other hand, you manually create a [Template Library] folder, you'll need to supply at least one template application in a subfolder before you'll be able to create a new application from a template.

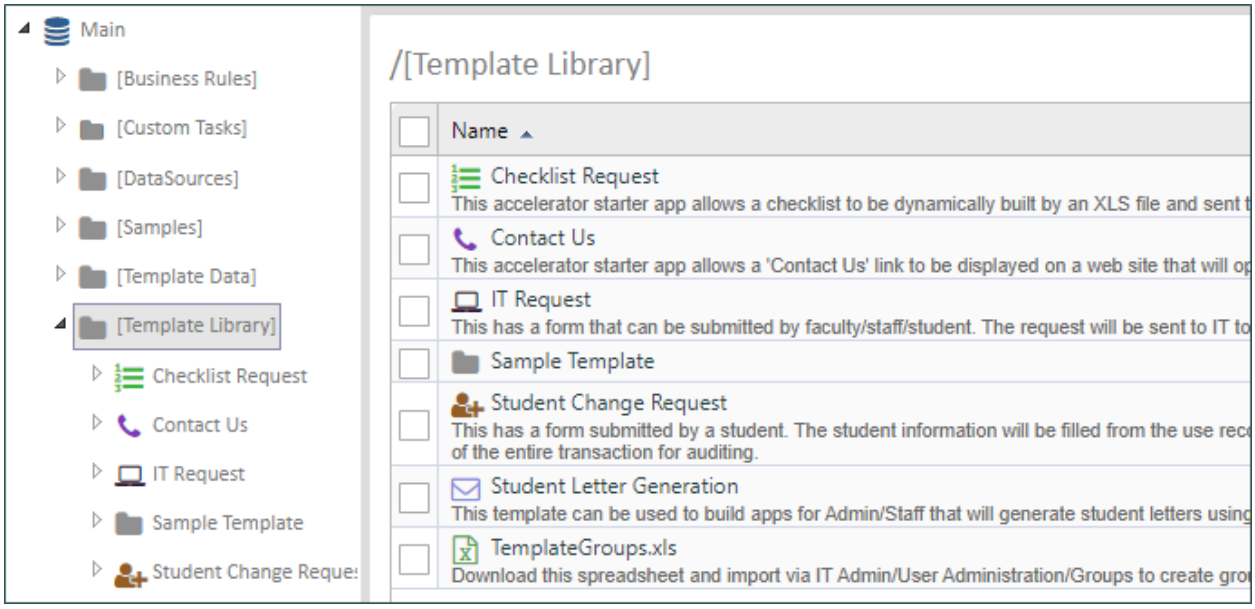
In general, a template application consists of:

- A pre-built form that contains some useful form fields.
- A Process Timeline that is linked to the form and configured to start when the form is submitted. There are usually some partially configured Timeline Activities in the Process Timeline.
- An email template that is used to send notifications from the Process Timeline.
- Additional objects such as Knowledge Views for reporting, and Dropdown objects for pre-filling the options in dropdown controls.

For Process Director v6.0.300 and higher, if you do not have a [Template Library] folder, or if the folder contains no application subfolders, the user interface will display a dialog box notifying you that no templates are available.

When creating the subfolder for a template application be sure to give the folder a logical Name for the application. Additionally, you should provide a description for the folder in the Description field. The description you write here will be displayed to the user when they choose a new template for creating an application, so provide a description that describes the purpose of the application.

Pre-Built BP Logix Templates



BP Logix provides some pre-built template applications that can be downloaded from the BP Logix Support site's Download page, in the section labeled, "BP Logix

Template Library". The download file is a zipped PDZ file that can imported i**nto the root of any partition**. When importing, the PDZ will create the [Template Library] folder at the Partition root. This folder will contain several template applications from BP Logix that you can use as starter templates. The Import will also create a [Template Data] folder that includes some Dropdown Objects, Business Values, and an Internal User Database data source for use in the pre-built template applications. This folder is required for the pre-built applications to work properly. There's also an Excel spreadsheet with administrative items like user Groups that must be edited and imported into your system prior to using some of the pre-made template applications. This Excel file is named TemplateGroups.xls.

In the example above, there are seven different template application folders, each of which contain the appropriate Forms, Process Timelines, and other objects. When using the Template Library feature, all of the objects in the selected [Template Library] subfolder will be replicated into the location you desire.

Requests applications handle a request with approval steps. Checklist applications enable you to dynamically build a list of actions that can be sent to users for completion. Letter Generation templates enable you to use Process Director to dynamically create and send letters. Feedback applications enable you to build applications to collect feedback.

For the pre-built BP Logix templates, an additional folder [Template Data], will be installed when you import the templates PDZ file. This folder contains some universal data objects, such as a Datasource object that connects to the Internal User Database to store and retrieve custom data, Dropdown objects used across multiple application forms, Business Values, and other data objects that the BP Logix template applications use. You can customize these items, and changes you make to the objects in this folder will be reflected automatically across all template-based applications. If you have data objects that you want to use for your own template applications, you can place your custom template data objects in this folder as well, though, for most purposes, placing them in the general Content List folders for global objects, such as [Data Sources], [Dropdown Objects], etc., is a better option. The [Template Data] folder is included in the import of the BP Logix template applications largely to avoid inadvertently overwriting your existing global object folders.

Some Template applications contain a form that uses the Bootstrap framework for the Form's layout, using controls from the Responsive Layout controls menu.

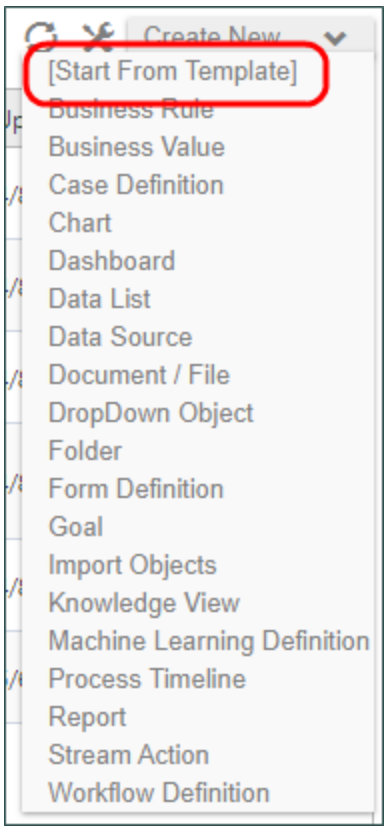
These controls are only available if your Process Director installation has enabled Bootstrap. If you have Bootstrap enabled on your system via the fIncludeBootstrap custom variable, you can use the Bootstrap Form in place of the standard form.

Creating an Application from a Template Library

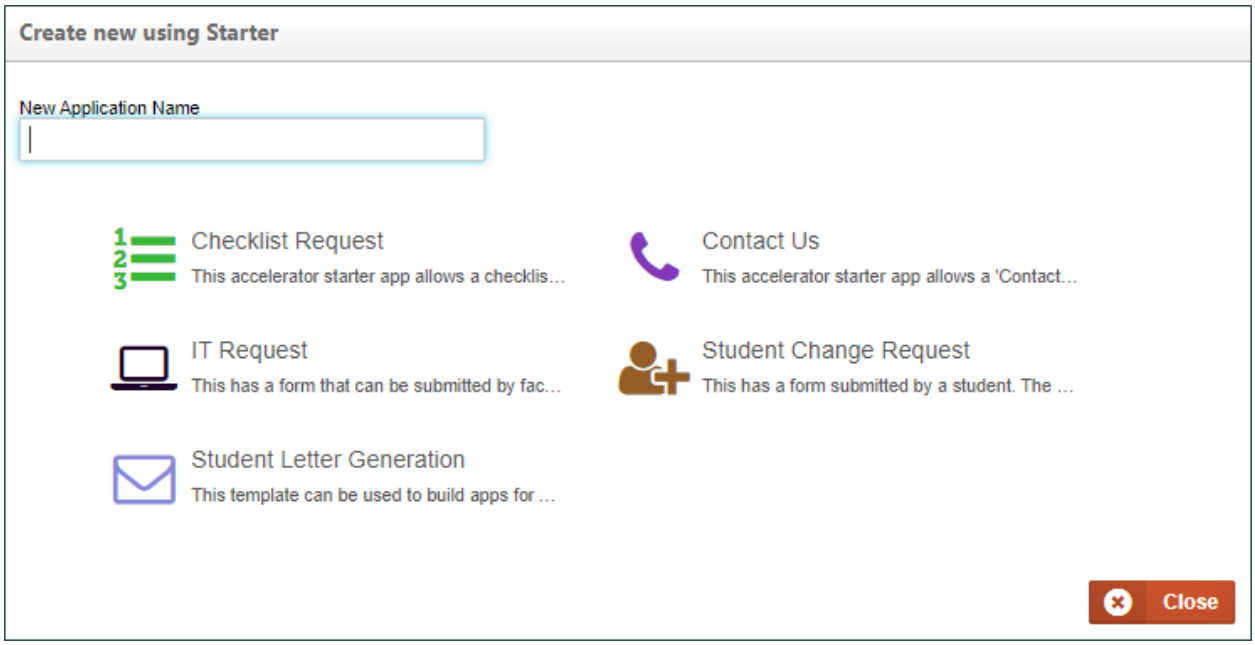
To create a new application from the Template Library, you can create it directly from a Content List folder, and, for Process Director v6.0.300 and higher, from the Home page.

Creating from the Content List

First, navigate to the folder location in which you'd like to create the application. Then, select the [Start from Template] option from the Create New... dropdown menu.



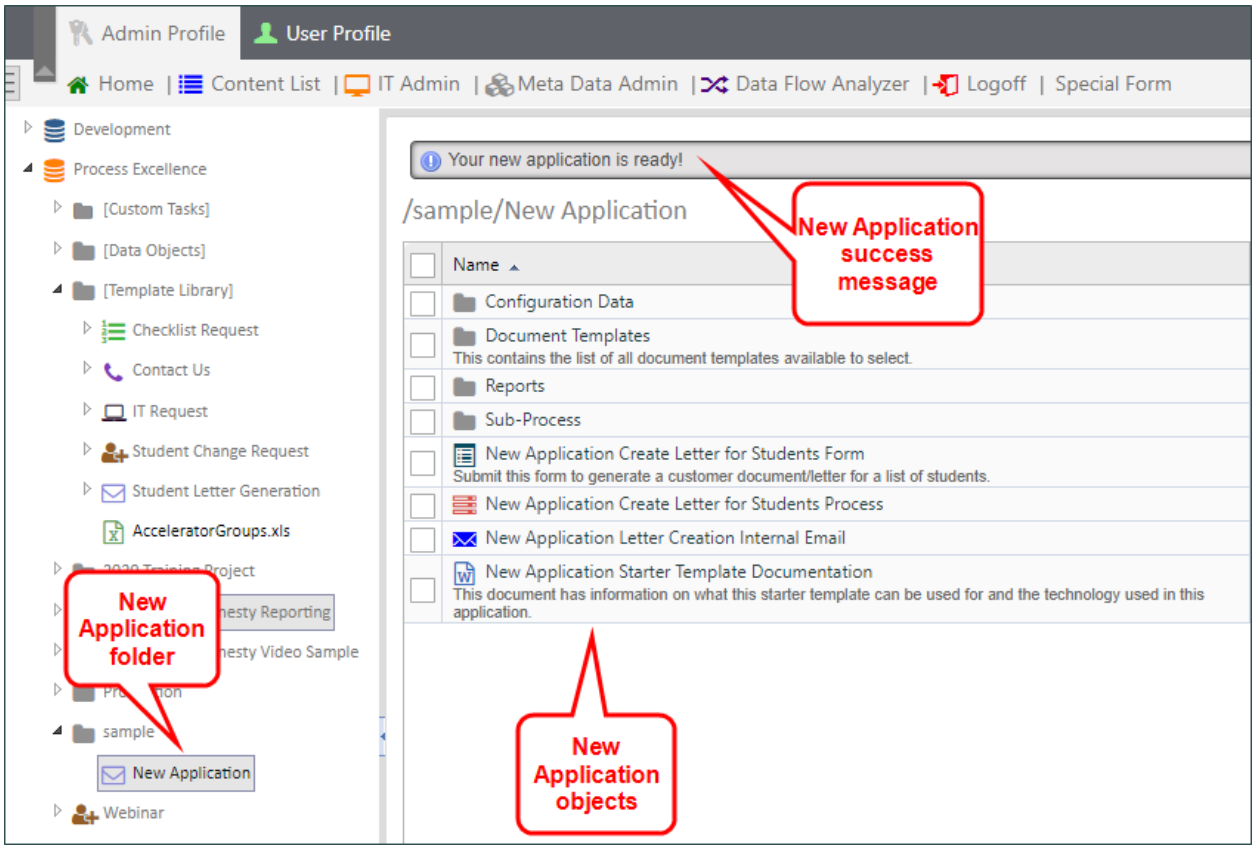
The Create new using Starter dialog box will open to display the list of [Template Library] subfolders from which you can select the desired template. All of the existing Template templates will be shown as clickable buttons.



In the New Application Name text box, type the name you want for your new application. The name you type here will be used to name the folder that will contain your new application. This name will also be appended to the names of all Process Director objects created in the new folder. Below the New Application Name text box, a list of all of the template folders contained in your [Template Library] folder will appear. In the case of this example, we have the Template folders we referred to above. Simply click on the desired template folder to select it. When you do so, a confirmation dialog box will appear, asking you to confirm this choice.

Once you confirm the choice in the dialog box, Process Director will automatically copy the template into a new folder in the location you chose.

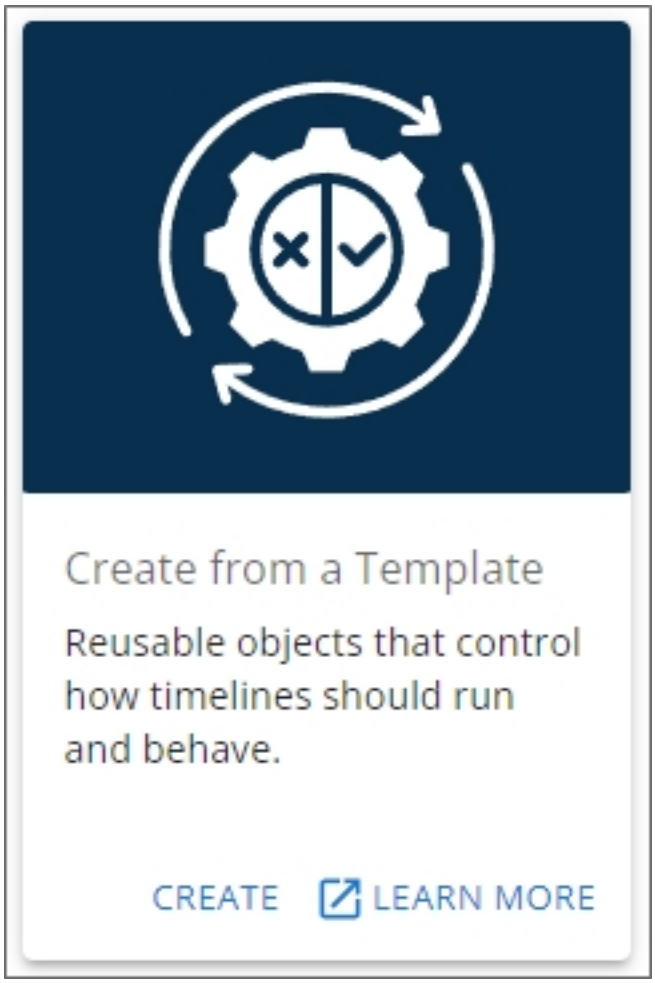
In this example, we used "New Application" as the name for our sample application.



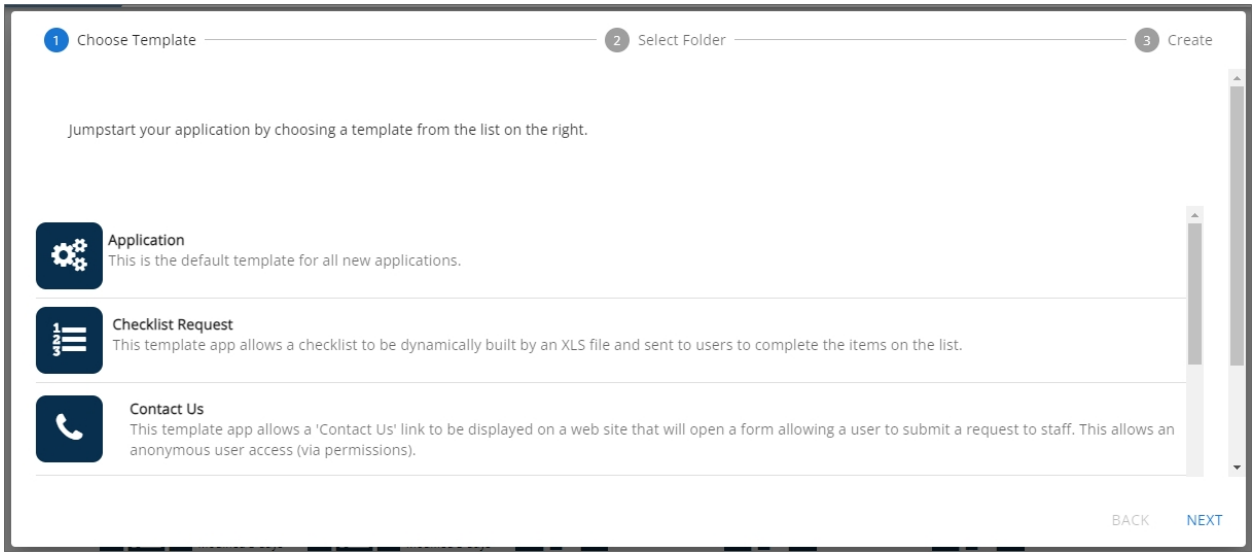
The New Application folder displays the name we provided in the New Application Name text box. Additionally, the names of the template objects will also display the same name, "New Application", appended to the original object name. Objects in subfolders will be similarly renamed. A success message will also appear, notifying you that the application is ready for editing. You can now configure the new objects as you desire.

Create from the Home Page

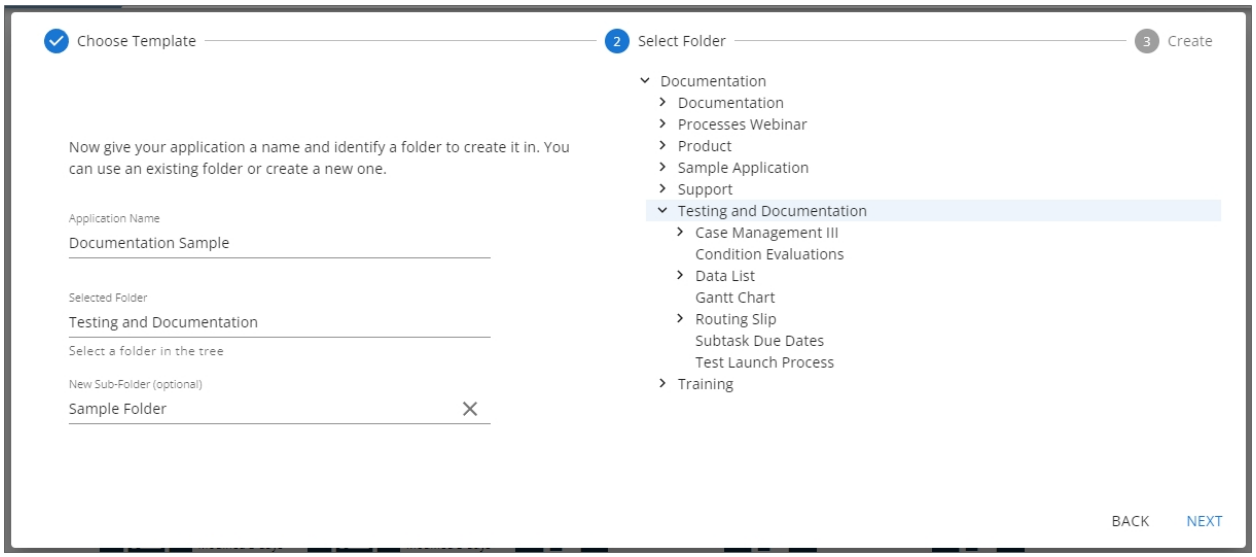
For Process Director v6.0.300 and higher, the Home page displays a panel labeled "Create from a Template", which enables you to create a new template-based application. A Learn More link will, when clicked, open this documentation topic in a new window.



The Create link will, when clicked, open a wizard that will guide you through the process of creating the new object you desire.



The first pane of the wizard is the Choose Template pane, which displays a list of all of the application templates that exist in the [Template Library] folder of your installation. Simply scroll to the template you wish to use, and click it to select it. For this example, we'll choose the Application template. Once you've selected a template, click the Next button at the bottom right corner of the wizard to advance to the Select Folder pane.



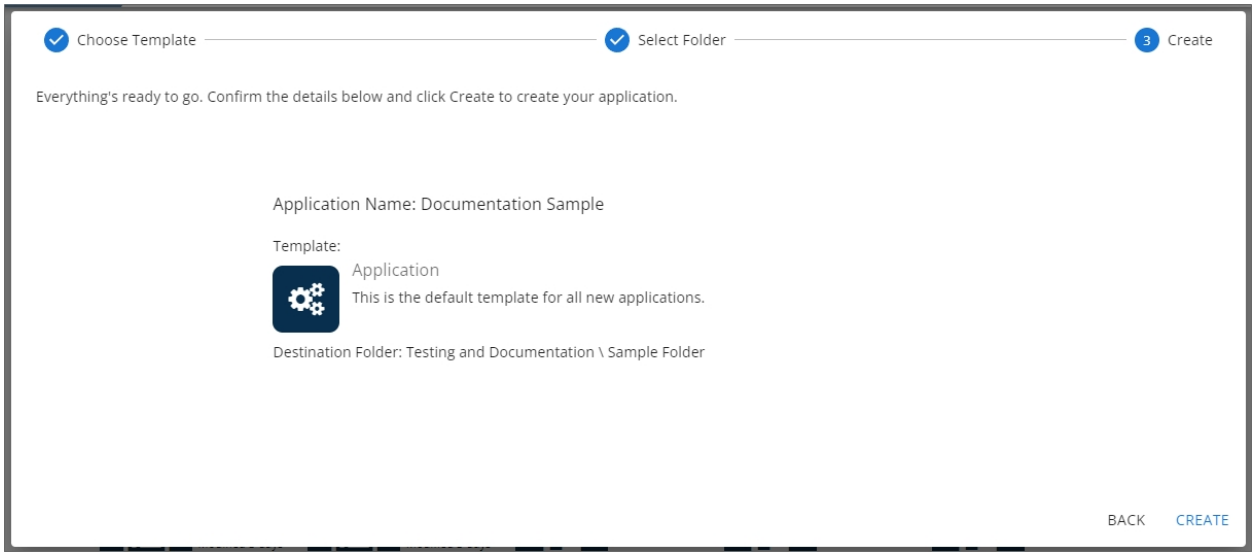
On the left side of the Select Folder pane, enter the name for the new application in the Application Name property. Once you've done so, you can navigate through the folder list displayed on the right side of the pane to find and select the folder into which you want to add the new application by clicking it (The folder must already exist in the Content List). When you do so, the folder name will appear automatically in the Selected Folder property, as shown above. If you'd like to place your application in a new folder that does not yet exist on the system, you can optionally enter the name of a new subfolder to create in the New Sub-Folder property. Keep in mind that the application itself will automatically be created in its own new folder, so if you add a New Sub-Folder, the application will still be created in a new folder inside the specified subfolder.

So, in the example show above, the new application, which will be named Documentation Sample, will be created with the following folder structure:

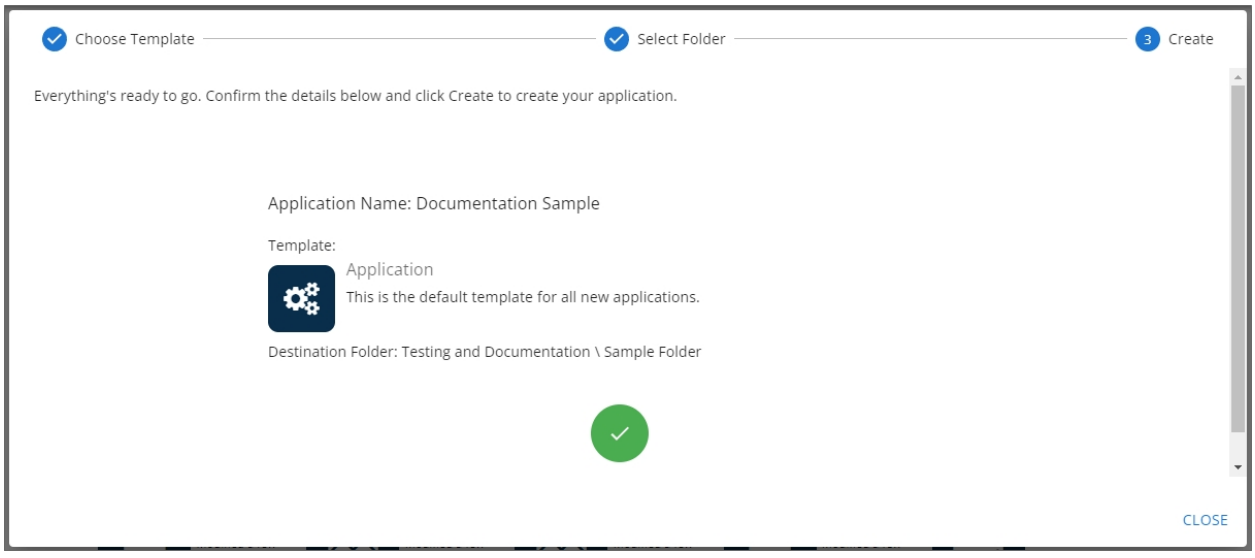
```
Documentation/Testing and Documentation/Sample Folder-
/Documentation Sample
```

All of the application objects will be located in the Documentation Sample folder upon creation.

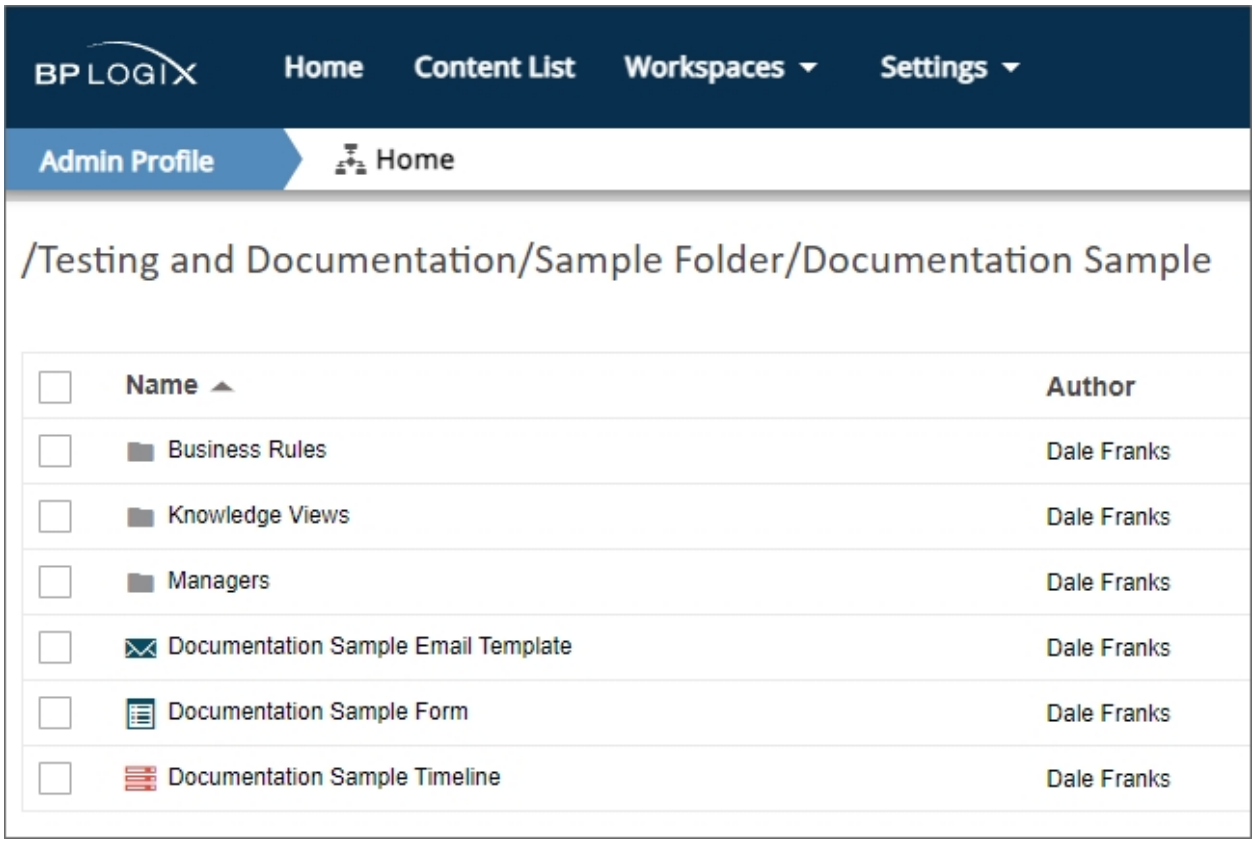
When you're done configuring this pane, click the Next button again.



The Create pane enables you to review the choices you've made in the previous two wizard panes. You can, at any time, click the Back button to return to a previous pane to edit your choices. Once you finished reviewing your choices, you can click the Create button to create the new application. When you click the Create button, a "working" animated icon will briefly appear, followed by a success icon when the application has been created.



Click the Close button to exit the wizard. Process Director will automatically navigate you to the new application folder, which is Documentation Sample in this example, and open it for editing, so that you can complete its configuration.



Importing Data

It's a very common requirement to use some internal organizational data in Process Director. For on-premise installations, this simply requires creating the appropriate Datasource that points to the data's repository, whether it's a CRM or ERP system, or some other SQL Server or Oracle database. For Cloud installations, however, this can be more difficult, as it can for organizations that don't wish to have even an on-premise installation of Process Director access the data directly.

Since Cloud installations are on the public Internet and most organization's have firewalls that prevent their organizational data from being accessed from outside the network boundary, access to the data is a bit more difficult. Moreover, most organizations are unwilling to open a hole in their firewall to enable a Cloud installation to access data from inside the boundary. So, for Cloud Customers, the data must be pushed to the Cloud Installation. For Process Director, the preferred method of sending data is via an Excel file, or series of files. Happily, importing the data is relatively straightforward.

First, create Excel exports of your organizational data. You can do this one time, or on a scheduled basis. When you do so, send the Excel files to a desired export folder. These files will need to be imported into Process Director, so be mindful of the technical requirements for Excel Datasources.

Next, in your Process Director installation, create a destination folder that will store the Excel files in the Content List. While you're at it, you should also create the Internal User Database Datasource that will be used to import the Excel files into tables in the Internal User Database.

Now, you can use the bpImport utility to transfer your Excel files from your file system, to the destination folder in your Process Director Installation. You can do this manually for the first import, but it's also useful to set up the Window's Schedule Utility to import the files on a scheduled basis by calling the bpImport utility to send the files on the schedule you desire. The bpImport utility enables you to save the import configuration as a command that you can paste into the Windows Schedule Utility to run on schedule. For on-premise installations, the bpImport utility is available in your installation folder for Process Director. Otherwise, contact BP Logix, and we'll be happy to provide you with a copy. The bpImport utility can be run on any machine, so you don't need Process Director installed locally for it to work.

Once you've imported the files the first time, you'll need to configure the Excel files to import the data into the Internal User Database, using the Datasource you configured previously. Again, see the Excel Datasources topic for the details of how to configure this. If you're going to import the files on a scheduled basis, be sure to select the Automatically import this Excel file after every check-in or import property so that, every time the import runs, the data automatically gets re-imported.

Once you're done, you should have the following actions taking place on a manual or scheduled basis:

1. The data gets exported from the data repository to an Excel file or files, with the files stored in a specified file system folder.
2. The bpImport utility runs, to transfer your Excel files from the specified file location to the specified destination folder in the Process Director Content List.

3. When properly configured, every time the old Excel files are replaced in the Content List by the newly imported files, the Excel data will be automatically imported into the Internal User Database.

Once the data is imported, it's immediately available for use in Process Director, via Datasources that use the Internal User database to access the data.

Obviously, you'll need to consider how fresh the data needs to be, in order to set up the appropriate schedule for the data import, as the data will only be as current as the most recent import.

Managing Users

The following User ID's are created on a new Process Director installation.

- Administrator
- User 1
- User 2

The User ID's contain no passwords and have been given full permission to all objects on the server by default.

As part of your implementation, you'll create new User ID's and groups, as well as setting the appropriate permissions for your environment

When users want to login to Process Director, they must navigate to the login page located at `https://ServerName.com/` where `ServerName.com` is the host name where the server is installed. This will redirect the user to the login.aspx page in that directory. Depending on the authentication model used, the page below may be displayed, prompting for a User ID and password. If Windows Authentication is enabled, the user may be prompted with a Windows login dialog box to enter the user's credentials.

User Profile

Each user has a profile that contains settings and information about their User ID. The values include their email address, an optional Display Name and an option to change their password. Please note that only built-in users can edit settings on this page. Active Directory users must contact their administrator for changing profile

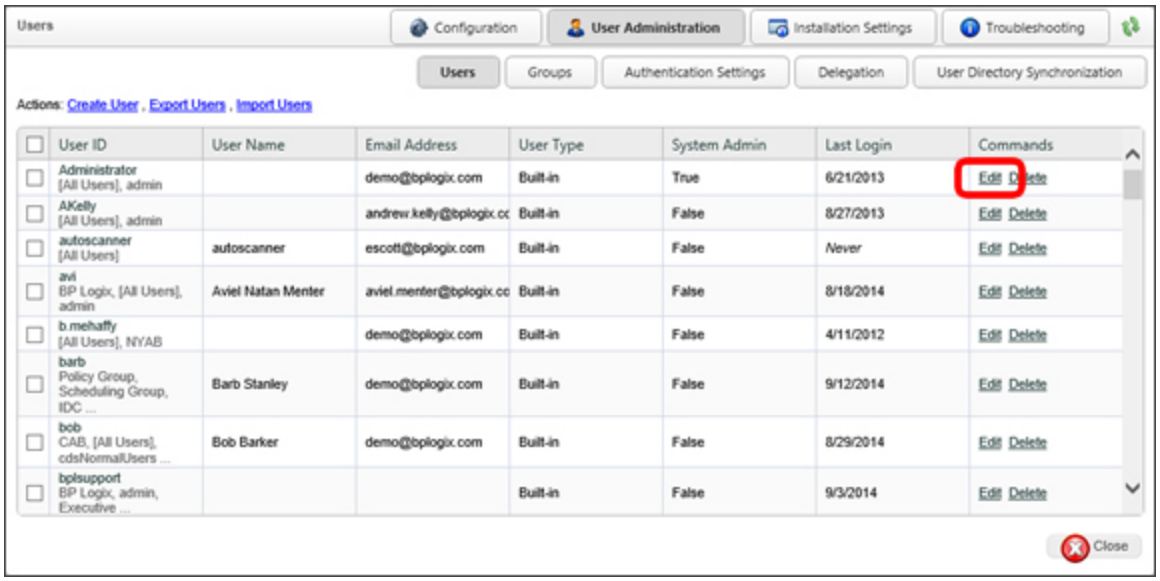
details. The administrator can also control these options from the Process Director IT Admin area, allowing the profile information to be configured for users. A user must have the appropriate authority to modify their own settings (by default all users can modify their profile) if the menu item is available.

Signature Image File

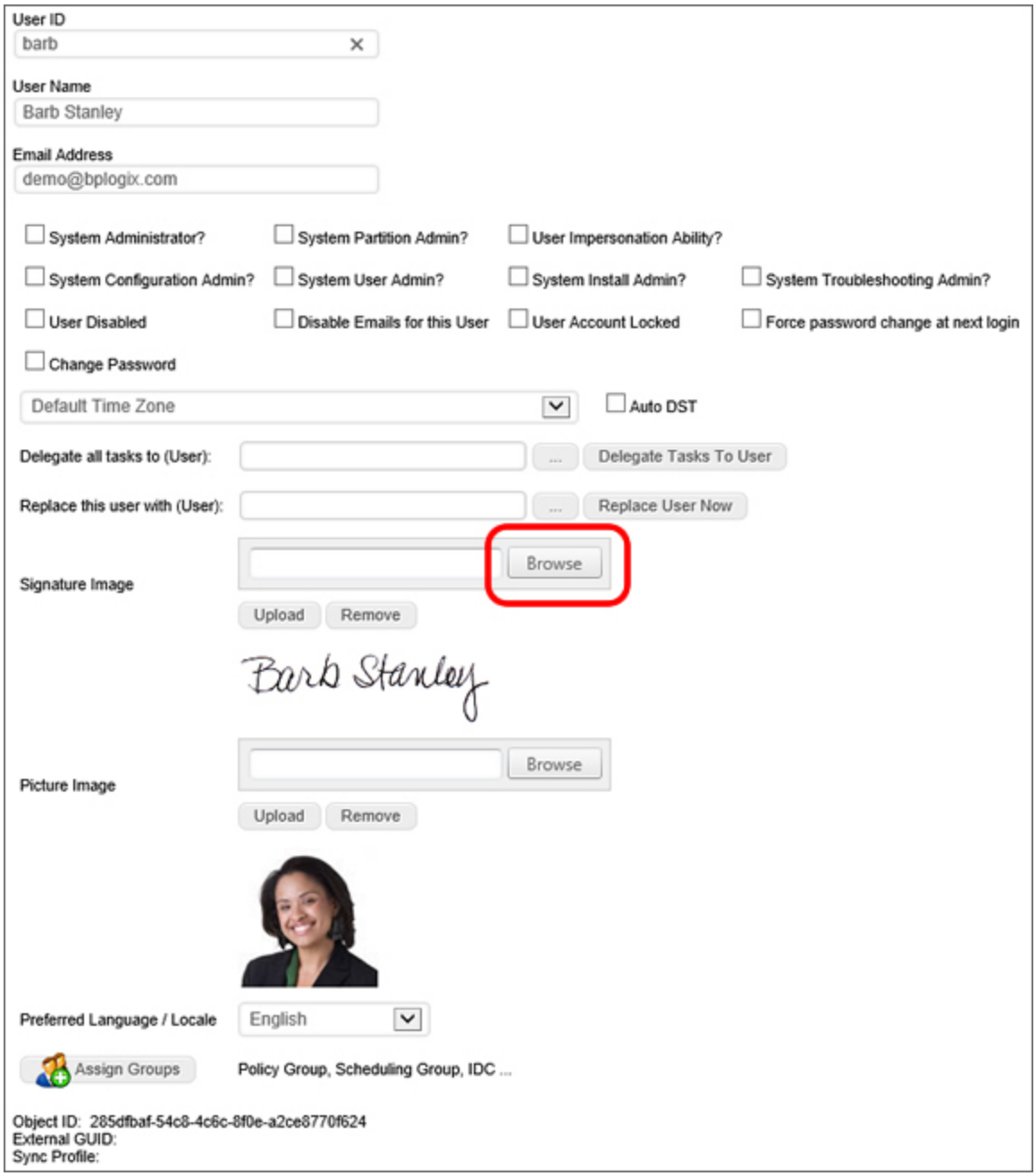
An optional signature image file can be associated with a user ID causing it to be displayed next to their name in the Routing Slip when they complete their task. You can upload a signature for each user in the User Administration section of the system. If a file exists with the same name as the UserID or UID in the \BP Logix\Process Director\website\custom\signatures\ directory it will use that for the signature (it must have a ".gif" or ".jpg" file extension such as Susan.gif). It is recommended that a ".gif" file be used with a transparent background so the image can be displayed on web pages with different color backgrounds.

To attach a signature to a user, log in as an administrator and go to the IT Admin area's User Administration section, which will open the Users page automatically.

Click on the Edit link for the user to which you want to attach the signature.



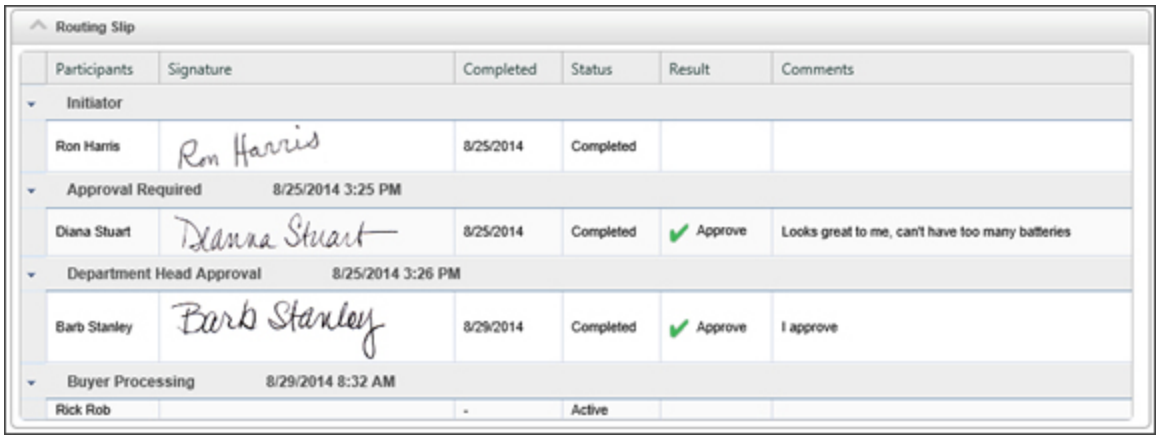
Click Browse on the Signature Image field, select the signature's image, and then click Upload.



Save your changes by clicking the OK button.

As an aside, you can follow the same process to add a user's image to the Picture Image field of the profile.

If a signature file is found it is automatically displayed anytime the Routing Slip is viewed. Below is an example of how the Routing Slip would appear with signature files uploaded to the system.



External Users

Process Director can be used by **external users**, which is to say, users from outside your organization. The most common use for external users is to include customers, suppliers, or other outside stakeholders as participants in a process.

In Process Director, **authenticated users** are those users who can be positively identified by Process Director, either through their internal Process Director user accounts, Active Directory, SAML, or Windows Single Sign-On. These users are considered to be internal users.

External users fall into to two broad categories. **Anonymous users** are those users who can't be identified, and have no identity within the system, as authenticated users do. Anonymous users are only able to submit forms, and can't be assigned tasks as a process participant. A **Non-Authenticated** user has no user account in the system, but does have a known email address that can be assigned a pseudo-identity. When an Anonymous user is identified with an email address, the user automatically becomes a Non-Authenticated user, and can be assigned tasks as a process participant, via email.

It is important to remember that the Anonymous and Non-Authenticated user designations don't describe two separate classes of external user, but rather describe two different roles that may be assigned, at various times, to the same user. An external user can be an Anonymous user in the context of one operation (e.g., anonymously submitting a public-facing form), while simultaneously being a Non-Authenticated user in the context of a different operation (e.g., being assigned a process task via email address). Whether an external user is considered Anonymous or Non-Authenticated is based entirely upon whether or not the user has been

identified by email address in the context of a specific operation. If so, the user is a Non-Authenticated user in the context of that operation, but is still an Anonymous user in all other contexts.

For systems still licensed under the legacy Tiered/Perpetual license models prior to 2017, access to Process Director for unauthenticated or anonymous users is enabled only by an optionally licensed component.

A **process participant**, is any user, authenticated or non-authenticated, who is assigned a task in a process. In the case of non-authenticated users, tasks are assigned by sending an email to the user's email address. The email contains a special link to open a Process Director object, such as a Form, to enable the non-authenticated user to perform a process task.

All external users who can't be specifically identified by Process Director (including both Anonymous and Non-Authenticated users) are considered part of the Anonymous Users group when setting Process Director object permissions.

As mentioned above, some Process Director licenses don't allow anonymous users. Additionally, for licenses that are granted on a per-user basis, any participants in a process—whether authenticated or anonymous—are counted as Process Director users for the purposes of licensing. Contact BP Logix if you are unsure whether your installation of Process Director is licensed to enable anonymous users.

Anonymous users can be granted permissions to access objects just like any normal user by assigning permissions for an object to the Anonymous Users group. For example, you'd assign Read permissions, and perhaps Modify Children permissions, to Anonymous Users on public-facing Forms and Knowledge Views that you'd like external users to be able to access.

If an object is given Anonymous User permissions, then all other Process Director users will also have those permissions.

If a non-authenticated user participates in a process, Process Director will automatically grant the required object permissions to the user to view or modify

objects, usually Forms, when the user needs to participate in or complete an assigned task in a process. Since non-authenticated users can't log into Process Director, their every action and notification is based on their email address. Process Director sends email messages to non-authenticated users that give them the appropriate links to Process Director objects. As long as Process Director has a valid email addresses for the non-authenticated users, they can do the following:

- Non-authenticated users can have aTask List, but they can only access their Task List from an email that assigns them to a task.
- A non-authenticated user can be assigned a task just like a normal user, and can even be assigned a task in the same step as a normal Process Director user.
- The Routing Slip will display the appropriate indication when an anonymous user completes a task, usually by showing their email address as the participant name.

When a Process Timeline Activity is assigned to a user with an invalid UID or user ID, Process Director will immediately stop the process, and place the Process Timeline Activity into an error state. This is also true for anonymous user assignments if the email address isn't provided in a valid format.

Process Director can't validate whether the email address is valid, only that the email address is provided in the proper format.

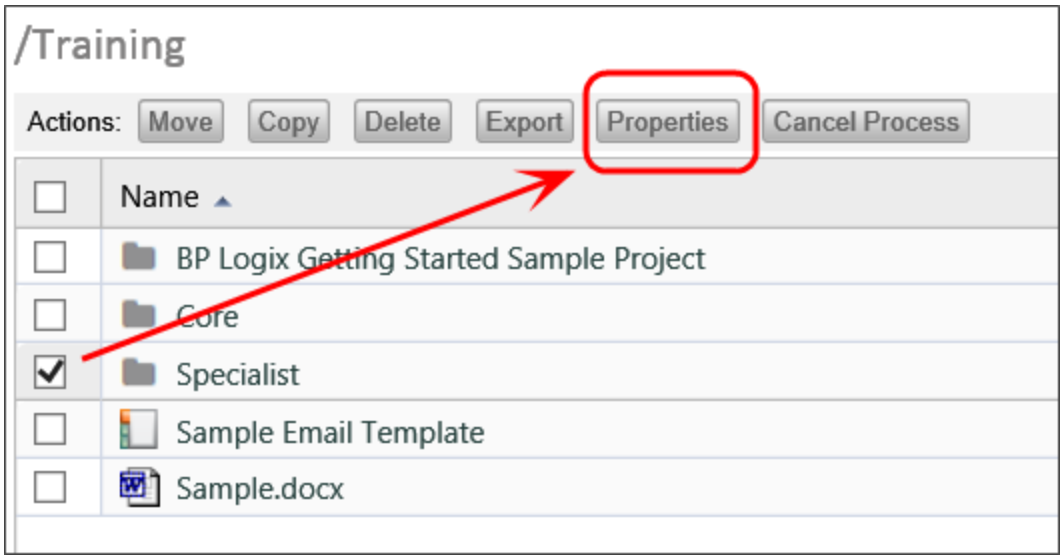
All user Timeline Activities have an option to enable non-authenticated users to complete a task based on the user's email address. This is a required option when using the Email Anonymous Task List system variable. In the vast majority of use cases, however, the email address will be taken from a Form field identified on the Participants tab.

In the case of user steps/activities that may be assigned to non-authenticated users, exercise care in using the “Automatically show user’s next task if it is in this process” setting. Process Director doesn’t distinguish among non-authenticated users when determining if the next task should be displayed, so if the current and subsequent tasks belong to different non-authenticated users, the subsequent task will still be displayed in the window of the current user. In the case of assignment to non-authenticated users, only use this option when you are certain that the same non-authenticated user will be assigned both tasks.

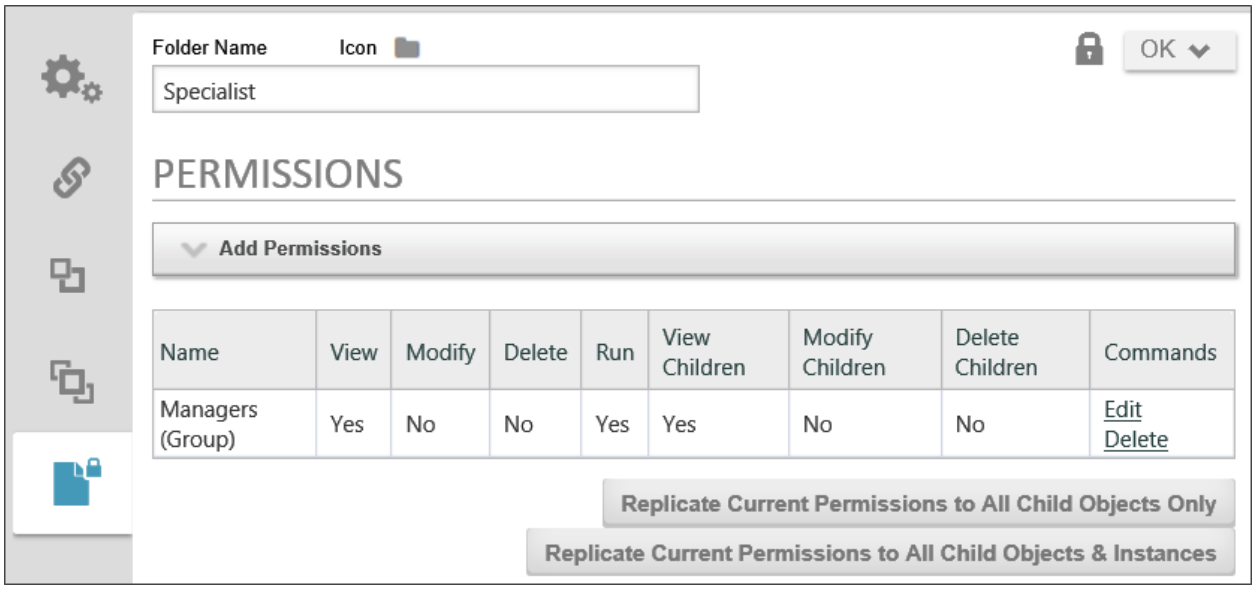
Permissions

Permissions control who has access to different objects (document, Form, folder, etc.) in the Process Director database. Sometimes referred to as Access Control Lists (ACLs), permissions provide the foundation for the securing of objects in the database. They control who can perform what functions on which objects. Permissions can also be used to provide a simpler interface to end-users, hiding the more advanced functionality from them.

Permissions can also be accessed from inside any Content List object by opening the object's definition. Open the definition by selecting the object, then clicking the Properties button.



Once the object is open, the permissions for the object can be configured on the object's Permissions tab.



See also:

Permissions Methodology on Production Systems

Setting Permissions

Permissions Methodology on Production Systems

In Process Director, permissions grant users access to Process Director objects in the content list. Every Content List Object can have permissions applied to it, though permissions should primarily be set on folders, rather than individual objects. Permissions are actually records that are stored in the Process Director database, and each record applies to a different user, group or class of users. Permissions records that are applied to a user or a group are static, which is to say that they always apply to those specific users. Records that apply to a class of users are dynamic, meaning that they can apply to different users at different times.

For instance, one class of users is Process Users. Any users who fall into this class, by virtue of participating in the process, receive the permissions for the class when they are accessing Process Director in the context of that process. At all other times, their permissions revert to the permissions of their group or their user account. So, a user might not have permission, under normal circumstances, to view a particular Form. But, when participating in the process, that same user might receive View and Modify permissions while they are participating. When they

are done participating in the process, their permissions automatically revert to the normal permissions applied to their user account.

By default, when Process Director is first installed, permissions on the system are granted extremely liberally. The default permissions are to grant All Authenticated Users full access to create, edit, and delete every Content List object. So, your first job will be to impose permissions that are appropriate to the server. Setting the default permissions should be a part of the initial installation or configuration of every Process Director server.

On a Production or Development server, the easiest method of setting initial permissions is to open the folder properties for the partition and set All Authenticated Users with Run and View Children permissions. While the View Children permissions are not, strictly speaking, necessary, users will not be able to view Form instances in Knowledge Views, or document attachments in Form instances without View Children permissions, so it's often easier to set them as the initial default. You can then replicate these permissions to all the existing child objects. These settings constitute the minimal hardening you should implement on Production systems. For maximal hardening, you may wish to remove all access for All Unauthenticated Users at the partition root and replicate the permissions to the rest of the partition. You can then provide different permissions, as necessary, to partition folders.

On a Development server, setup may be more complicated. In most cases, the users, groups, and folder organization of the Development server should mirror those of the Production server very closely. Yet, you'll need to set the permissions for implementers and designers on the Development system very expansively. Implementers usually need to create, edit, and delete folders and other objects. Adding your implementers to a specific group on the development server, then applying full permissions to members of that group, might be the best option.

Administrative Users

System or Partition Administration power should never be granted to a normal user account, including the normal user accounts of administrators. Each administrator should have a normal user account with the same access as any other authenticated user. If an administrator needs to take administrative action on a Production system, then the user should be required to sign out of their normal user account, and log into an individual administrative account of the Built-In

account type prior to taking any administrative action. Each administrator should have a unique administrator account, so that actions taken by an administrator can be associated with a specific person. You should not allow administrators to use a shared account to perform administrative tasks.

Users

Anyone who participates in a process, or anyone who has an identity in the system, is a user. Anyone who is identified in Process Director, either through having an internal Process Director user account, or identification through Active Directory, SAML, or Windows Single Sign-on authentication is considered an authenticated user. Anonymous users are users who participate in a process, but whose identity can't be positively authenticated. For instance, you can assign tasks to users who aren't part of your organization, by assigning them a task via email. These are anonymous users who access Process Director through a special URL provided in the email. Process Director can't authenticate these users, and assumes that anyone who accesses a task via the email link is the desired user.

Children

In Process Director, when a new child object is created, it automatically inherits the permissions of its parent. This inheritance is applied only when a new object is created. Changing the properties of a parent object won't change the permissions on existing objects.

There are different types of objects that are considered children. For instance, in the Content List, a folder is a parent object, while all of the objects contained in the folder, such as definitions for Forms, Process Timelines, Business Rules, etc., are the children objects for that folder. Similarly, every time a new Form is submitted, a new Form instance is created, and the Form instances are the children of the Form definition. Each Form instance, once created, now has its own permissions that apply only to it. Changing the permissions for a specific Form instance won't affect the permissions for any other Form instance.

Quite often, in Process Director, you attach documents to a Form or Process Timeline instance. When you do so, the attached document automatically inherits the permissions of the object instance to which it is attached. In essence, the attachment is treated as a sibling of the instance.

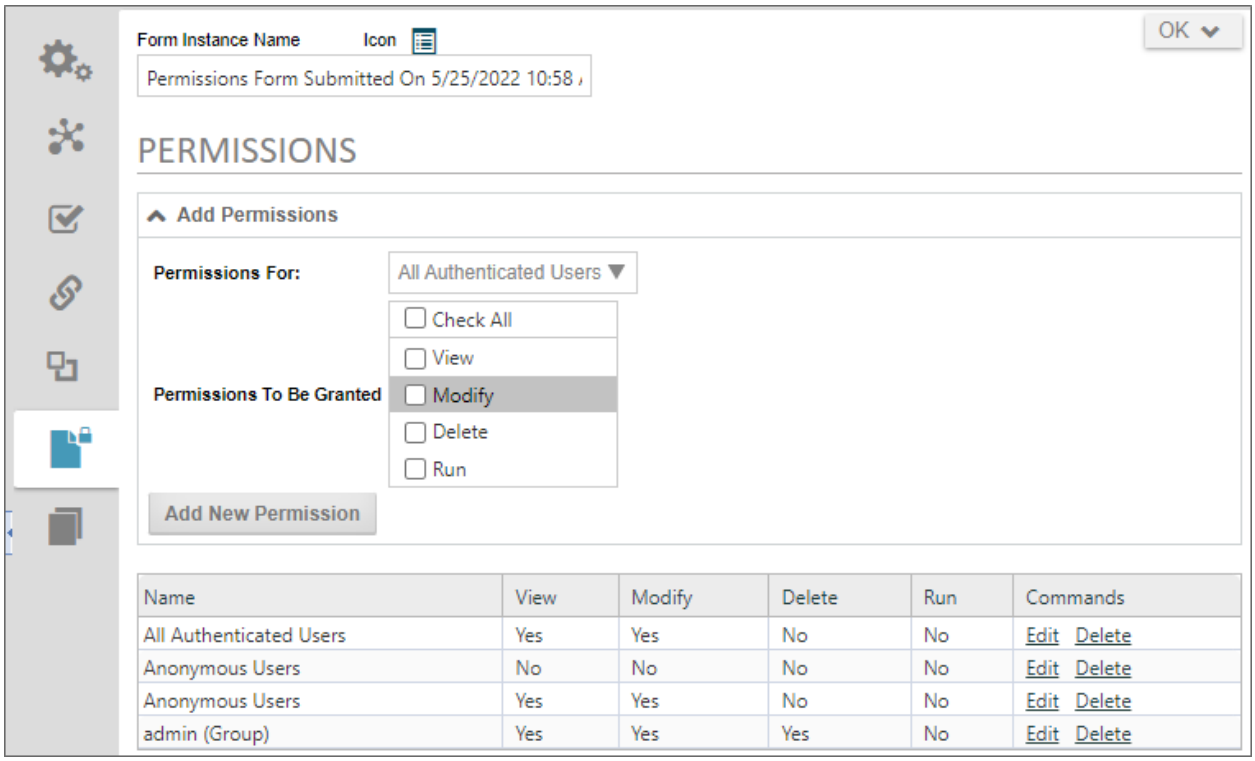
Permissions Methodology

Not all permissions are meaningful for all objects. For instance, there is a Run permission, which is applicable to process objects like Process Timelines, but isn't applicable to a document like a Word or PDF attachment. Setting a Word document, therefore, with Run permissions doesn't accomplish anything useful.

When setting permissions, some permissions automatically imply others. For instance, if you give someone Modify permission to an object, you are also granting View permission as well. Similarly, Delete permission also applies View and Modify permission. Applying a more destructive permission always implies—and grants—any less destructive permission. In Process Director, the implied permissions are made explicit, in that, when you check the box to select a more destructive permission, the less destructive permissions will automatically become checked as well, so that you can see the granting of any applicable implied permissions as soon as you select the desired permission.

As mentioned previously, objects automatically inherit the permissions of their parent object. A Form instance inherits the permissions of the Form definition, a document attachment inherits the permissions of the instance to which it is attached, and so forth. You may, however, wish permissions for child objects to be different than their inherited permissions. Process Director contains a Custom Task to address this issue, the Item Actions Custom Task. In the Item Actions Custom Task, you can modify the permissions of attached objects. This does add some extra logic to your process, however, in that you must select when the Custom Task will run, e.g., from a Form or Process Timeline. But, however you choose to do it, you must explicitly change the permissions for the attachment via the Item Actions Custom Task.

When changing permissions via the Custom Task, you may create a permissions rule that conflicts with an existing one. For example, look at the screen shot below:



In this example, Anonymous Users have two different permission rules. One of them grants View and Modify permissions to the form instance, while the second one, added via the Item Actions Custom Task, removes both permissions. Remember, the system will parse the entire permissions list and grant all permissions found. So, in this conflict, the most permissive rule will win, meaning that the View and Modify permissions will be granted, irrespective of the more restrictive permissions rule. So, you must think carefully about how permissions are applied, to avoid such conflicts.

Permissions should be set early on. If you have separate development and production systems, for example, you should have folder permissions already set on the production machine prior to importing any objects from the Development machine. Similarly, when you begin creating a project, you should create a project folder and set permissions on the folder prior to creating the project objects, since new objects always inherit the permissions of the parent object.

The question then arises, "What default permissions should be set on the parent objects?" Sadly there's no simple answer to this question, though there are general guidelines. You should always be sure to harden the permissions adequately to prevent unauthorized access to the objects, which means setting permissions so that as few people as necessary can access the objects. Commonly, hardening is

done on the basis of Lines of business. You may have a parent folder that contains all of your HR processes, for example, and permissions to that folder might be limited to only members of the Human Resources department. Subfolders containing individual projects might be hardened even further, so that only specific subgroups of the HR personnel can access a specific project. Line of Business hardening is, perhaps, the most common method of determining permissions.

As mentioned previously, permissions don't import to or export from a system. Indeed, permissions are specifically prevented from importing/exporting. Usually, the permissions on a development system are far more permissive than on a production system, if for no other reason than designers/developers shouldn't be able to access many processes, like sensitive financial or human resources processes, in a production environment. Preventing permissions from exporting or importing enables you to set up hardened permissions on the production server, while leaving the development server relatively open to support collaborative design efforts.

User administrators have the ability to replace one user with another. This is useful as people change jobs or leave the organization. When using the Replace User function, user replacement automatically replaces the old user in all of the permissions records with the new user. In other words, the permissions assigned to the old user will be assigned to the replacement user. This is a convenient way to replace a user without having to recreate all of the permissions records for the new user from scratch. User replacement is, essentially, a universal replacement with permissions, just as it is with process participation.

In cases where you'd like users to open and submit a form—even if they don't have permission to view or modify the Form in any other context—you can assign all users to have Run permissions for the Form. After the user submits the Form, all of the regular permissions are applied, so if the user doesn't have Read or Modify permission, they'll no longer be able to see the Form in most cases. The exception is if the user is assigned a task later in the process. Process users are always temporarily granted the minimal permissions level they need to complete a task, so, even users who don't normally have permission to see a Form can open the Form in the context of completing the task. Once the task is complete, regular permissions are, again, applied to limit the user's access.

In addition to permissions, Forms offer another level security, in that you can show or hide controls on a Form by using the visibility properties, or disable controls with the Readonly properties. In both cases you can set conditions the hide or

disable Form controls based on the user's identity, or any other evaluable condition. This enables you to hide sensitive data even for users who need to see other portions of the form. This isn't permissions-based, but it does add another level of security, in addition to controlling access to the Form itself.

When setting permissions, the question always arises of where to implement the desired permissions settings, since you can set permissions on any object. As a best practice, you should set permissions at the folder level for Content List objects. When you create or import objects in the folder, the new objects will automatically inherit the permissions of the folder. So, in the case of importing from development to production, if you set the permissions on the parent folder on the production system prior to the import, all of the permissions for the folder will be replicated to the imported children objects automatically.

Finally, you should exercise care when moving a Content List object from one folder to another. When you create an object in a folder, it inherits the folder permissions on creation. Process Director has a Move function that enables you to move objects from one location in the Content List to another. An existing object that is moved from one folder to another will bring its existing permissions with it, rather than overwriting them with the permissions of the folder to which it is moved. So, when moving objects to a different folder, you must go to the new folder's permissions list and click one of the Replicate Current Permissions buttons to overwrite the existing permissions of the moved object with the permissions of the new parent folder. Alternatively, instead of moving the object, you can export and reimport it to a new location, as exporting does remove all permissions from the object. In most cases, though, it is probably easier to simply move the object, then replicate the parent folder's permissions.

Delete Permissions

No user account in a production version of Process Director should have delete permissions on any object, nor should any user account in Production be designated as a partition administrator. The only account that should have delete permissions, or be designated as a partition administrator, is a built-in administrative account. No one should *ever* log into this account unless they are specifically doing so to delete items from the Content List.

Deleting an object definition from the Content List automatically deletes all instances of that objects. Deleting any Content List object is a permanent action.

There is no "undo". Objects deleted from the Process Director database are gone forever. The only remediation for deleting an object and/or its instances is to restore the database from the most recent backup.

For this reason, delete permission on production systems should be very tightly controlled. No one should be able to delete any Content List object when logged in using their regular identity. Force users to log into specific administrative accounts to delete any Content List object from a production system.

In general, Delete and Delete Children permissions should never be granted to any users in a Production system. System and Partition Administrators, by virtue of their inherent User Permissions, can still delete objects, but no user should be granted Object Permissions to do so

Permissive Access

By default, all permissions in Process Director are permit-style permissions, which means that users are denied access to objects unless they are specifically permitted to see the objects. The permit-style permissions model adequately hardens Process Director for nearly all purposes.

There are a limited number of use cases that might require the denial model. For example, you can’t deny a user who is otherwise included in a permitted group without deny permissions. So, in a law office, you might want to bar one of the attorneys from participating in a specific case because of a conflict of interest. Deny permissions would be required to prevent the attorney from viewing the objects that are included in that case instance. Use cases like this should be rare, however.

Process Director can implement the denial model by setting the fEnableDenyPermissions custom variable to true in the custom vars file. Implementing the denial model isn't cost free, however, because it requires the system to perform additional permissions checks every time a user opens an object. This may not be noticeable when opening a form, but it will make Knowledge Views run perceptibly slower. Unless you have a specific use case for implementing denial permissions, you shouldn't implement the denial model.

Other Considerations

Irrespective of the permissions assigned to regular Content List objects, there are some special consideration that must be given to global objects such as Custom

Tasks, Datasources, and Business Values. Similarly, Knowledge Views that return Content List objects have their own special considerations.

Global Objects

Custom Tasks, Business Values, and other global objects often require access by all implementers who create and/or configure Content List objects on the system. For example, disabling the View permission on Custom Tasks will make it impossible for implementers to select Custom Tasks when configuring a Form a Process Timeline. If the Custom Tasks aren't viewable, attempting to add a Custom Task to a Form or Process Timeline Activity will result in the Choose Custom Task dialog box displaying a blank page, with no Custom Tasks displayed.

While Run permission will enable any end user to invoke a Custom Task at run-time, implementers who do not have View permission will be unable to add the Custom Task to a Form or Process Timelines. Similar issues will arise when trying to configure a Business Value in the Set Form Data tab of a Form or Process Timeline Activity, or trying to choose a Datasource when attempting to configure a Business Value. Without View permission, implementers will be unable to see the objects in the lists from which they need to be selected. All implementers need to belong to a user group that has View permissions on these global objects in order to use them in configuring Forms, Process Timelines, etc.

Knowledge Views

In most cases, Knowledge Views displayed for end users will return a set of Form instances, from which the user can select and open the instance for viewing or editing. It's important to remember that the permissions applied to the Knowledge View itself are **not** applied to any of the objects the Knowledge View returns.

So, for example, if a Knowledge View is configured to return Form instances, the parent Form definition for those instances will need to have View Children permission. The Form instances, when created, will then be viewable to end users, as they will have View permissions applied them automatically, via inheritance of the View Children permission that was configured on the Form definition. Conversely, if the instances do not have permissions that enable them to be viewed by end users, then the Knowledge View will return no records when an end user runs it. A Knowledge View definition, irrespective of the permissions applied to it, will not return and display objects for which the end user does not have permission.

Technically, the system will operate normally as long as All Authenticated Users have Run permission on the Content List objects. They'll be able to create and submit new Form instances, start process instances, etc. As task assignees, they'll be able to edit the Form instance to complete their tasks. As a practical matter, however, this limited permission set will prevent Knowledge Views from accomplishing their intended purpose, which is usually to display Form or Process instances.

Similarly, if you'd like end users to be able not only to view the instances returned by the Knowledge View, but to edit them, the parent definition will need to have Edit Children permissions applied. Once again, the instances, when created, will inherit this permission as Edit permission on the instance. Of course, you should limit the grant of Edit Children permission to specific groups/users to whom you wish to have universal access to edit the child instances. For instance, authenticated users might have View Children permissions on the object definition, so that they can view the instances. A smaller group, such as process managers, might also have the Edit Children permission, so that only they can edit a Form instance returned by the Knowledge View. Of course, as mentioned in the Permissions Settings topic, assigning Edit Children permission to the managers automatically includes View Children Permissions.

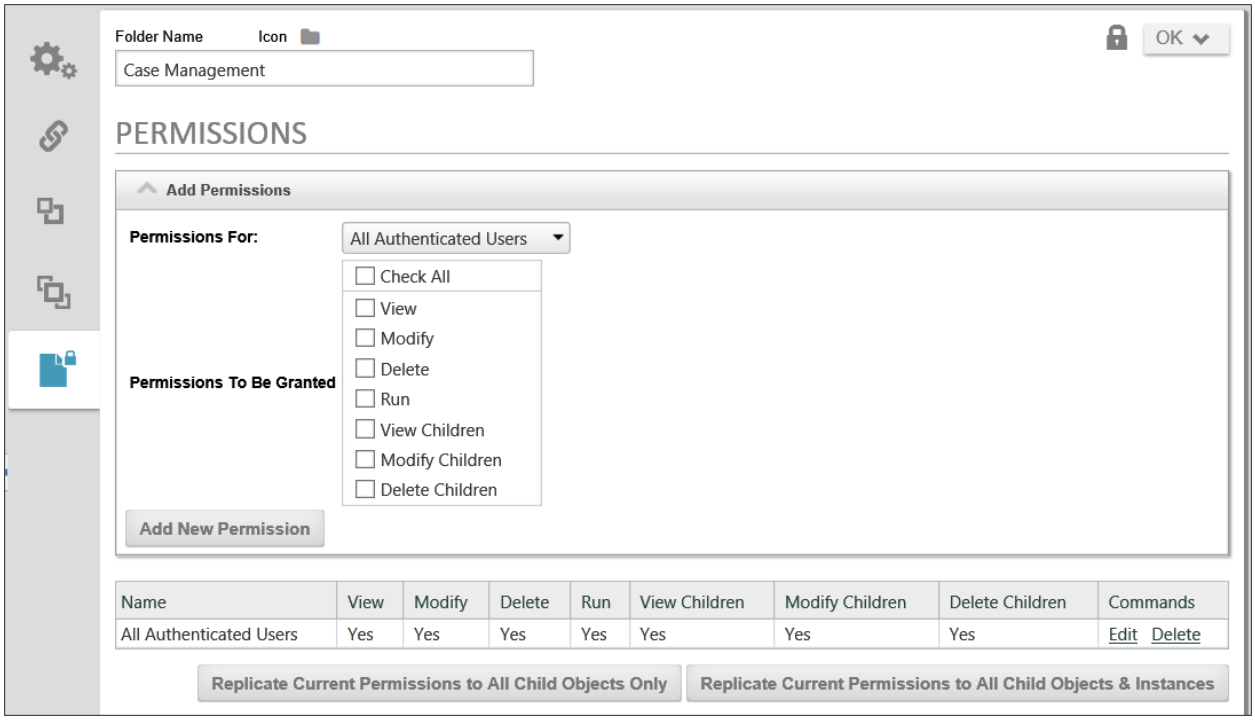
Keep in mind that, if a user has an assigned process task, and the Knowledge View is configured to open instances in task context, the end user will, if running in an active task, have the appropriate permissions to edit the object until their process task is complete. As a task assignee, they'll belong to the temporary class of Process Users, with all of the appropriate permissions they need to complete the process task, irrespective of the more limited permissions they may be assigned as a user or group member.

Setting Permissions

Any object in the Process Director database may contain a list of permission rules that determine who can access the object and what functions they can perform against it. A permission rule only grants a user access. The absence of a permission rule granting a user access to an object will automatically deny access to it. If there are conflicting rules which either grant or deny a user access, the user will be granted access. When the server attempts to determine whether a user has access to an object in the database, it will search through the object's entire permission list until it finds a permission rule that grants the user the requested access, or until

the end of the list. The system will apply the **highest level of access** that has been granted to that user in the list of permissions.

Different object types in the database have different permissions and options available to them. Objects that can be run (e.g. process definition, Form definition) will have more permissions options available to them, as will objects that can have children objects (e.g. folders, Form definitions, etc.).



The Permissions list is displayed in the lower portion of the permissions screen and will display the user and group names sorted alphabetically.

User Permission Types

Permissions are granted to a user. A permission record gives permission to something for a specific object in the Process Director database. Different types of user permissions can be granted.

Authenticated Users

An authenticated user is anyone who performs a login to access the system. Authenticated users have an identity (i.e. User ID) and exist in the Process Director database. They may be authenticated by Process Director, your Windows Domain, an LDAP server, or an external application.

Specific User

This identifies a specific authenticated user by name. All users known to Process Director have a User ID that is used to login. They may also have an optional display name or Alias. If an Alias exists for a user, that name is used to identify the user on all Process Director screens. If a user appears multiple times in a permission list, they'll be granted access according to the highest level of permission listed.

Specific Group

A group is a collection of users in the Process Director database. A user can belong to multiple groups. Only authenticated users can belong to a group. When access to an object is being determined by the server, it will give the user the highest level of permission found in any specific user record or any group the user is a member of.

Anonymous Users

Anonymous users aren't authenticated. They don't perform a login and they have no identity in the Process Director database. Only certain functions and objects are available to anonymous users. If an “anonymous” permission is given to an object, authenticated users will automatically be given that same permission.

Process Users

Any users who participate in a process as a task assignee receive the permissions for this class when they are accessing Process Director in the context of that process instance. At all other times, their permissions revert to the permissions of their group or user account.

Object Creator

This is a special permission type only available to objects that can have children (i.e. folders, process definitions, Form definitions). This will automatically give the permissions specified to the author/creator of any child objects that are created. For example, if a user starts a process definition that gives the Object Creator a permission of Modify; that will automatically be converted to the initiator users ID with Modify permission for that specific process instance the user is creating (starting).

For Process Director v6.1.400 and higher, when user permissions changes are applied, any affected user who is currently logged into an active session will be required to authenticate immediately, in order to apply the permissions changes to them.

Object Permission Types

Setting the permissions for a folder doesn't automatically set the permissions for the existing objects contained within it. You have the ability to replicate all permissions of the parent folder to all of the existing child objects in the folder. You have two options for doing so.

First, you can Replicate permissions to all child objects, which won't only overwrite the existing permissions on definition objects, it will also overwrite permissions on all of the form and process instances—including those instances that are active and in-progress. This may completely change the ability of users to complete active Process Timeline or Form instances, so you should exercise caution in using this option in the production environment.

You also have the option, though, to replicate the permissions to child all child objects except for form, process, and Process Timeline instances. This is probably the preferred option for replicating folder permissions in the production environment where there are active processes running. Future process and form instances will be created under the new permissions regime, but existing instances will still run under the permissions regime that was valid when they were created.

View

This permission gives a user read access to the folder. This doesn't grant the ability to modify the folder name or description. If a user only has View permission to a folder they won't be able to create any new objects below this folder.

Modify

This permission gives a user the ability to modify a folder, but not delete. A user with this permission to a folder can create new objects beneath this folder. A user must have Modify permission to view or set the permissions for the folder.

Delete

This permission allows a user to delete a folder. However, to delete a folder, the user must have Delete permission to all objects and sub-folders under this folder. If any object exists below this folder that the user doesn't have Delete permission to, the delete operation will be canceled and none of the objects will be deleted. A user must have Delete permission to be able to grant Delete permission to another user.

Document Permission Types

View

This permission gives a user the ability to view, read or download a document. This permission doesn't grant the ability to modify the document.

Modify

This permission gives a user the ability to modify the document. The document modifications will appear in the document history. A user must have Modify permission to view or set the permissions for a document.

Delete

This permission gives a user full control over a document. A user with this permission can delete the document. Delete permission is required for a user to perform a rollback to an older version of a document, or to delete any portion of the document history. A user must have Delete permission to be able to grant Delete permission to another user.

Process Permission Types

Permissions are defined for a process definition, e.g., a Process Timeline. When the process definition is run, a child is created as an instantiation of the process definition. This “running” process is located beneath the process definition in the Content List. This instantiated, running process will inherit the Child Permission settings from the process definition.

View

This permission gives a user the ability to view a process definition in the Content List. It doesn't grant the ability to run or modify the process definition, and doesn't grant any permission to the running processes instantiated from this definition.

Modify

This permission gives a user the ability to modify a process definition in the Content List. It doesn't grant the ability to run a process definition, and doesn't grant any permission to the running processes instantiated from this definition. A user must have Modify permission to view or set the permissions for a process definition.

Delete

This permission gives a user the ability to delete a process definition. This will also delete all running and completed process Instances beneath this process definition in the Content List. A user must have Delete permission to be able to grant Delete permission to another user.

Run

This permission allows a user to run or start this process definition. If a user doesn't have Run permission to a process definition, it won't be displayed on their home page, nor will they be given access to the Run button for that object in the Content List.

View Children

This permission gives a user the ability to view the children of a process definition (e.g. running processes). When a process is started, the running process will copy all of the View Children permission rules as View permissions. If a user is granted View Children permission on the process definition, they'll be able to see all running and completed processes instantiated from the process definition. They will also be able to see the process definition in the Content List to enable the navigation to the running and completed process children. Users won't be view the actual process definition properties without View permission.

Modify Children

This permission gives a user the ability to modify the children of a process definition (e.g. running processes). When a process is started, the running process will copy all of the Modify Children permission rules as Modify permissions. If a user is granted Modify Children permission on the process definition, they'll be able to modify running processes instantiated from the process definition. This gives the

user administrative control over the running process (e.g. add user, remove user, skip step, restart step, etc.).

Delete Children

This permission gives a user the ability to delete the children of a process definition (e.g. running processes). When a process is started, the running process will copy all of the Delete Children permission rules as Delete permissions. A user must have Delete permission to be able to grant Delete permission to another user.

Form Permission Types

Permissions are defined for a Form definition. When a Form is filled out and submitted, a child object is created as an instantiation of the completed Form. This completed form is located beneath the Form definition in the Content List. Completed forms will inherit the Child Permissions from the Form definition.

View

This permission gives a user the ability to view a Form definition in the Content List. It doesn't grant the ability to run or modify the Form definition, and doesn't grant any permission to the completed forms from this Form definition.

Modify

This permission gives a user the ability to modify a Form definition in the Content List. It doesn't grant the ability to fill out and submit a Form, and doesn't grant any permission to the completed forms. A user must have Modify permission to view or set the permissions for a Form definition.

Delete

This permission gives a user the ability to delete a Form definition. This will also delete all completed forms beneath this Form definition in the Content List. A user must have Delete permission to be able to grant Delete permission to another user.

Run

This permission allows a user to fill out and submit this Form definition. If a user doesn't have Run permission to a Form definition, it won't be displayed on their home page and they won't be able to open it in the Content List.

View Children

This permission gives a user the ability to view the children of a Form definition (e.g. completed forms). When a Form is submitted, the completed form data creates an entry in the Content List and copies all of the View Children permission rules as View permissions. If a user is granted View Children permission on the Form definition, they'll be able to see the completed forms for this Form definition. They will also be able to see the Form definition in the Content List to enable the navigation to the completed form children. Users won't be able to view the actual Form definition properties without View permission.

Modify Children

This permission gives a user the ability to modify the children of a Form definition (e.g. completed forms). When a Form is submitted, the completed form will copy all of the Modify Children permission rules as Modify permissions. If a user is granted Modify Children permission on the Form definition, they'll be able to modify the completed forms under this Form definition.

Delete Children

This permission gives a user the ability to delete the children of a Form definition (e.g. completed forms). When a Form is submitted, the completed form will copy all of the Delete Children permission rules as Delete permissions. A user must have Delete permission to be able to grant Delete permission to another user.

Knowledge View Permission Types

Knowledge Views display objects contained in the Process Director database. Authenticated and anonymous users can be given access to the objects displayed in a Knowledge View. A user must have at least View permission to an object for it to be displayed in the Knowledge View.

View

This permission gives a user the ability to see the Knowledge View definition in the Content List. This permission doesn't grant the ability to run the Knowledge View.

Modify

This permission gives a user the ability to modify the Knowledge View definition in the Content List. A user must have Modify permission to view or set the

permissions for a Knowledge View.

Delete

This permission allows a user to delete a Knowledge View definition. A user must have Delete permission to be able to grant Delete permission to another user.

Run

This permission is required for a user to be able to run the Knowledge View. If a user doesn't have Run permission to a Knowledge View definition, it won't be displayed on the user's home page or when they select the Knowledge navigation button.

Category Permission Types

The category permissions are set in the category schema. To view the category schema use the Administration navigation button and select Meta Data Administration.

View

This permission allows a user to view the category in the category schema. If a user doesn't have View permission to a category they won't see this category name in the schema definition.

Modify

This permission allows a user to view and modify the category in the category schema. It allows the user to add sub-categories below this category. A user must have Modify permission to view or set the permissions for a category.

Delete

This permission allows a user to delete the category in the category schema. It also allows the user to delete any sub-categories below this category. A user must have Delete permission to be able to grant Delete permission to another user.

Run

This permission is required for a user to be able to assign the category to an object in the database.

Default Permissions for New Objects

Any time a new object (e.g. document, Review Set, sub-folder) is added to a folder it will be given the same permissions as the parent folder. If an object is added to the top-most folder (i.e. “<top>”) it will be given the same permissions specified in the default Folder Permissions hotlink. If an object is copied or moved to a folder, it will keep its original permissions.

Administrators

Any user that is checked as the “System Administrator” is given full administrative privileges. A user that is a System Administrator has full permission (View, Modify, and Delete) to all objects in the Process Director database.

Similarly, any user that is designated as a Partition Administrator will, for the partition of which they are a member, have full permissions on every object in the partition, regardless of any Object Permissions. Partition Administrators don't have access to the IT Admin area, as System Administrators do.

Adding/Removing/Editing Permissions

Permission records can be modified for an object by selecting the Permissions button. Each permission record has to be removed or added individually. To remove a permission record, click on the checkbox for that entry and select the Delete hotlink.

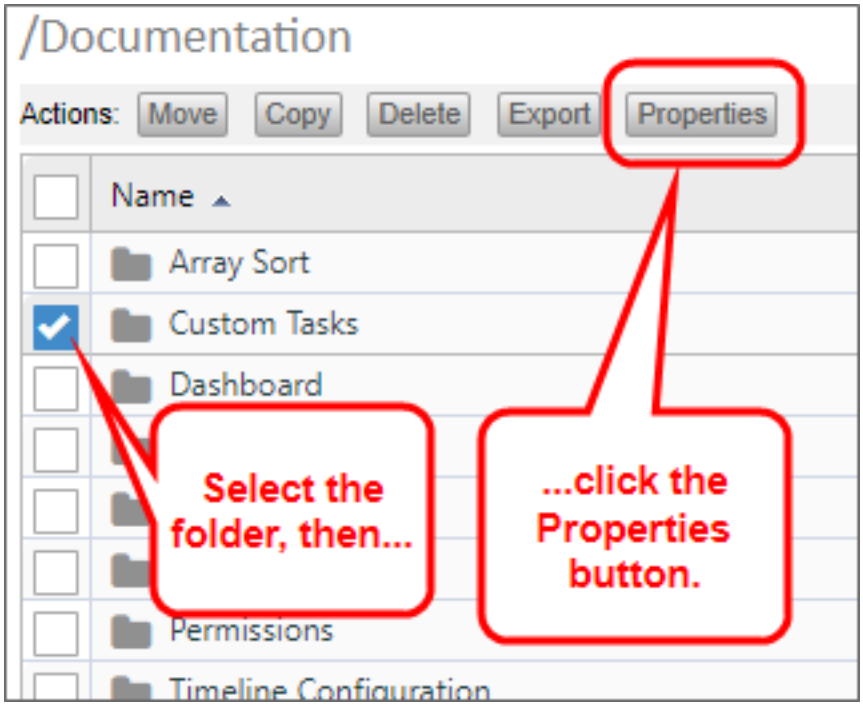
To add a new permission record, select the type of user, the type of permission and click on the Add New Permission button. You can't delete a permission record that would eliminate Write permission for yourself; the system will return an error message. If this occurs, add a new permission record granting your User ID Write permission, then delete the other permission record.

The Edit link enables you to edit an existing permission rule. Clicking this link will place the rule in a visible edit mode to enable you to reconfigure the selected Permissions Rule. Checking one of the permissions check boxes will add a permission to the rule, while unchecking a check box will remove the permission from the rule. Once you've configured the rule, you can click the Update button to save your changes or click the Cancel button to ignore them.

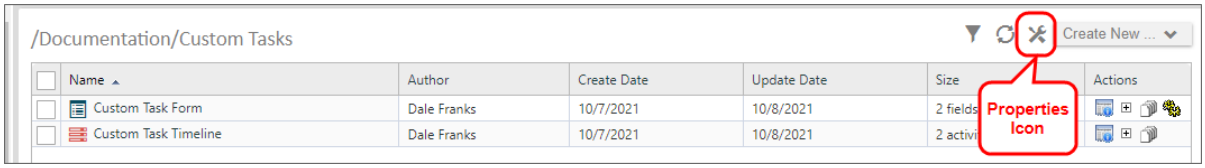
Folder Permissions

In the Content List, at the level of each folder, you may set permissions for that folder by

- Checking the checkbox next to the folder, then clicking on the Permissions link located at the top left of the Content List.



- Navigate to the folder, then click the properties icon when your inside the folder.



With either method, once the properties screen opens, you can navigate to the Permissions tab of the object definition.

You can add, edit, or delete folder permissions from this screen. A fully-detailed explanation of permissions is documented in the Permissions topic of the Implementer's Reference.

Replicating Permissions to Child Objects

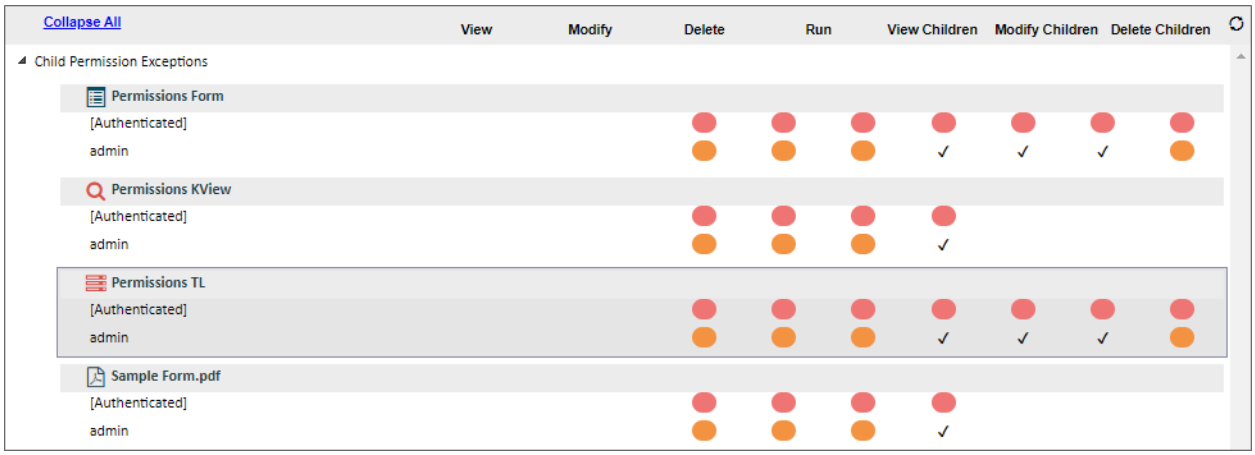
You have the ability to replicate all permissions of the parent folder to all of the child objects in the folder. You have two options for doing so.

First, you can Replicate permissions to all child objects, which won't only overwrite the existing permissions on definition objects, it will also overwrite permissions on all of the form and process instances—including those instances that are active and in-progress. This may completely change the ability of users to complete active process or Form instances, so you should exercise caution in using this option in the production environment.

You also have the option, though, to replicate the permissions to child all child objects except for Form, Workflow, and Process Timeline instances. This is probably the preferred option for replicating folder permissions in the production environment where there are active processes running. Future process and form instances will be created under the new permissions regime, but existing instances will still run under the permissions regime that was valid when they were created.

Permission Exceptions

At the bottom of the Permissions tab a button labeled, Show Permission Exceptions, will, when clicked, display a chart of all child objects, if any, whose settings for the same assigned permissions class are different from the object you are viewing.



In the case of this example, the parent object is the application's parent folder. A form contained inside this folder, named Permissions Form, has different permissions from the folder in which it is contained. This chart can be used to determine which child objects may need to have permissions replicated.

See also:

Permissions Methodology on Production Systems

Setting Permissions

Securing Process Director

Like any system that is exposed publicly, whether it's to the Internet or to a corporate intranet, security should be a primary administrative concern. There are a number of security practices that can be implemented to protect a Process Director installation from unauthorized access. This especially important for systems that may contain sensitive data such as Personally Identifiable Information (PII) or data that is covered under HIPAA.

Process Director includes a number of features and settings that assist you with tightening security to the product. To properly secure your Process Director installation, consider implementing the practices below.

Cross Site Scripting Protection

For users of Internet Explorer, Chrome, or Safari, Process Director v3.54 and higher implements cross-site scripting protection by implementing the XSS filter in the HTTP header, with the setting:

```
X-XSS-Protection: 1; mode=block
```

When the browser detects a cross-scripting attempt, it will stop rendering the page completely, rather than attempting to sanitize the attack and render the page.

The Firefox browser does not support cross-site scripting protection via the XSS filter.

Starting with Process Director v3.78, Cross-Site Request Forgery (CSRF) protection is enabled by setting the fEnableReferrerProtection custom variable. When set to "true," this variable enables Process Director to screen for CSRF by ensuring that the HTTP Referrer header is valid.

HTTP Strict Transport Security

For Process Director v4.54 and higher, HTTP Strict Transport Security (HSTS) has been implemented to protect against "man in the middle" attacks, like protocol downgrade or cookie hijacking attacks. The HSTS setting is sent to the browser via an HTTPS response header field named "Strict-Transport-Security", and specifies a

time period during which the browser can only access the server via a secure transport layer.

Properties Settings

On the Properties page of the Installation Settings section of the IT Admin area, there are two properties to which you should pay special attention.

Interface URL

Consider configuring your Process Director installation to use SSL. SSL encrypts all data moving between the client and the server. To do so, you'll need to acquire an SSL certificate for the server, and configure the certificate via IIS. Once you have properly configured SSL, you can set the Interface URL setting to use the HTTPS URL for your Process Director installation, preventing anyone from capturing clear-text data being sent in transit to or from your Process Director installation.

Local IP Addresses

The Local IP Addresses setting enables you to add a comma separated list of IP addresses that have permission to access the IT Admin area of Process Director as if they were a local user. Ensure that this setting only includes IP addresses from which you want to grant full administrative permissions without authentication.

By default, this setting is empty, which only allows localhost users to access the IT admin area of the product without authenticating (e.g. http://localhost/admin/). Adding additional IP addresses widens the range of users who might be granted unrestricted administrative access, thus reducing security.

If Process Director is running behind a firewall, additional care should be taken setting this value. If you add the IP address of the firewall, this effectively treats all access through the firewall as a local access.

Certain functions that would normally require system administrator privileges (for example: enabling debug mode, editing users) are available for users logged in to the server directly and connecting to the Process Director server using the “localhost” convention.

Web Service Enabled

If you don't need to make any calls to Process Director's web services, this property should be set to its default value of "False". The web service API calls can be

used to enable administrative functionality, so if the web services aren't needed, they should be turned off completely. If you do require web services to run, then you should restrict access to the web services to only the IP addresses that requires access to them by setting the Web Service Restrictions setting appropriately, as described below.

Web Service Restrictions

This property contains a comma-separated list of IP Addresses, and will restrict access to web services to only the IP Addresses entered. Leaving this field blank will allow universal access to web services with authentication.

Ensure that you are restricting web service calls only to IP addresses to which you want to give full administrative permission. Again, web service API calls can enable administrative functionality, so access should be restricted to only those trusted servers. To lock the system down more securely than the default, set this value to 127.0.0.1, which will allow only local access to the Web Services API, which is useful if you have no other server applications that use Web Services.

Custom Variable Settings

When using Cloud installations or on-Premise installations with the Compliance Edition, Process Director will keep an external audit trail of all activity. There are configuration options that should understand so that the settings you are using are implemented by design, rather than simply using the product defaults.

These values are configured in the /custom/vars.cs.asmx file in the PreSetSystemVars() function. Detailed information on how to set custom variables, with the appropriate syntax, are available in the Developer's Guide.

Cloud customers who have licensed access to their custom variables file can modify settings in the file via secure FTP. Other cloud customers can submit a support ticket to BP Logix to make any desired changes to the custom variables file.

Auditing Variables

fAuditLogFileOnly

This variable determines whether Process Director will write audit log information only to the text log files. The variable defaults to "True", meaning that it won't write any audit logging information to the Process Director database. Setting this

variable to "False" will cause the audit data to be written to the Process Director database, in addition to the log files.

Documentation: fAuditLogFileOnly

nArchiveLogDays

This variable sets the number of days to keep the zipped audit log files. The default value is 30 days. Log files are stored in the /website/App_Data/logs_archive/ folder. Any log files older than the number of days specified by this setting will be deleted.

Documentation: nArchiveLogDays

nAuditLogDays

This is the number of days to keep the audit log data in the Process Director database. The default value is 30 days. Any audit data older than the specified value will be removed from the database.

Documentation: nAuditLogDays

fAuditFormAPICalls

This variable determines whether Process Director will write an audit log for changes made by APIs. The default value is "True", which means that Process Director will log an audit record of API calls resulting in a change to form data. While we generally recommend that this value remain set to the default value of "True", please note that this setting can have an effect on server performance for applications rapidly performing large numbers of API-based form field changes.

Documentation: fAuditFormAPICalls

fAuditFormViews

This variable determines whether Process Director will write an audit log for all form views. This can result in a tremendous amount of audit logging when the value is set to "True". The default for this variable is "False".

Documentation: fAuditFormViews

User Authentication Variables

If you are using built-in user authentication with Cloud installations or on-Premise installations with the Compliance Edition, BP Logix recommends that you use the

password enforcement options. This will require that any user passwords conform to the enforcement options.

PwdStrength

This variable controls password complexity. It should be configured to use at least PasswordStrength.Medium. The default value for this variable is PasswordStrength.None. See the topic for the PwdStrenth custom variable for more information.

Testing Variables

fTestMode

This variable sets the Process Director installation to Test Mode when set to "True", and will allow a user to login as any other user without a password. In this mode, Windows Integrated authentication is disabled. The default value for this variable is "False". Ensure that this custom variable is always set to "False" in your production system. The variable should also be set to "False" on your development and test systems, except for those times when it is explicitly needed, after which, it should be set back to "False".

Documentation: fTestMode

Administration Variables

While these variables aren't specifically security-related, you should set them explicitly on your system to ensure the server load and process behaviors don't change when upgrading to newer versions of the product.

Activity Checking Custom Variables

Process Director uses background timer processes to check Process Timelines and Workflows for time-based operations:

- **TimerSecondsCheckProjAdvance** - Checks Process Timelines to advance activities when their start conditions are met or when delays expire.
- **TimerSecondsCheckProjReminders** - Checks Process Timelines to send scheduled reminder emails.

BP Logix recommends that you set these variables explicitly, even if only to set them to the default values. Please see the Activity Checking Custom Variables topic for more a detailed explanation of this recommendation.

User Settings

Users can be given administrative privileges in the User Administration section of the IT Admin area. There are some privileges that are very sensitive, and so should be used very sparingly, and only with an understanding of the capabilities these settings will give the user. Also, remember that these privileges can be given indirectly by granting them to a group to which the user is also a member.

On production systems, the best practice is to give no administrative rights to any "normal" user. Instead, build in a privileged user identity for each administrator, to which they should explicitly log in for the purpose of executing administrative tasks, and for no other reason.

System Administrator

This permissions setting gives the user access to the functions in the IT Admin area. With this privilege, a user can give any user any type of permission. Essentially, this permissions setting gives full control of the Process Director installation to the user, or any other user they choose to designate.

System Partition Admin

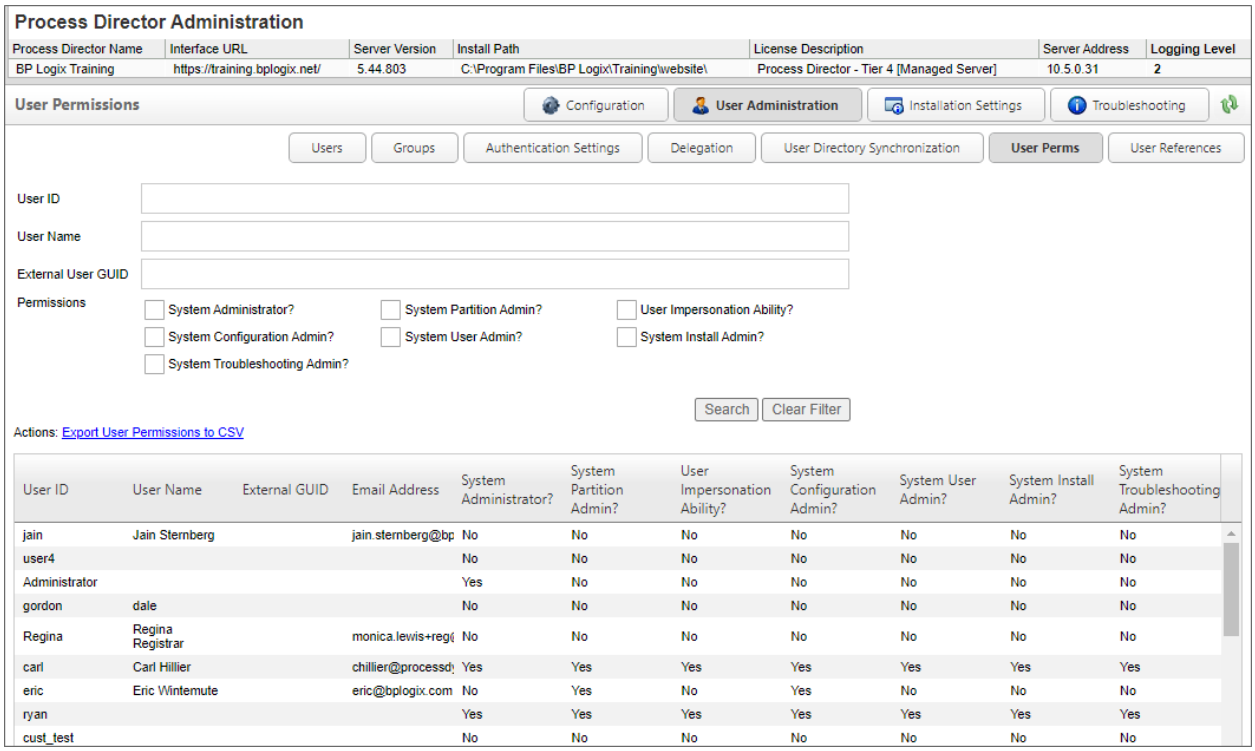
This permissions setting gives the user full permission to every object in the Content List of the partition for which administrator access is granted. All other permissions settings in the Content List of the specified partition are ignored for this user.

User Impersonation Ability

This permissions setting allows a user to impersonate any other user and take on their complete identity and privileges. Administrators should be extremely careful about who is granted user impersonation permission, though Process Director v5.13 and higher prevents non-administrators from assuming any administrative permissions via impersonation.

User Permissions Page

Administrators of Cloud installations or on-Premise installations with the Compliance Edition will notice an additional User Perms page in the User Administration section of the IT Admin area.



The User Perms page enables you to find users and view a report of permissions that are applicable to that user.

The top section of the user permissions screen provides a number of search fields that you can fill out to filter the users returned by the permissions search. The filter criteria will accept all or part of a User ID or User Name, or the GUID of an external user. Once you have entered the filter criteria you desire, click the Search button to see a list of all the users that match your filter criteria, as well as the permissions granted to the users.

The User Perms page is very useful for manually auditing the permissions granted to any user.

Built-In User Authentication Options

When using Cloud installations or on-Premise installations with the Compliance Edition, there are several callout points in the /custom/vars.cs.asmx file that allow you to control access to the system, password changes, etc.

More detail on these functions can be found in the Developer's Guide.

CanLogin()

This function is called when a user attempts to login, and can be used to reject a login, to redirect the user to another page, or to limit certain users to specific IP addresses.

LoginComplete()

This function is called when the login is successful.

ForgotPassword()

This function is called when the user clicks on the “I forgot my password” link on the login page. This is only for built-in users.

ValidatePassword()

This function is called when a user attempts to set or change their password, allowing for custom complexity requirements.

Audit Log Monitoring

When using Cloud installations or on-Premise installations with the Compliance Edition, and when storing audit logs in the database, you can search for various events. In the Troubleshooting section of the IT Admin area, there is an Audit Logs page containing a search function that allows you to find specific audit log types. BP Logix recommends that you periodically check audit logs for certain privilege changes to ensure that these privileges are the ones expected on the production system.

- AdminPageAccess
- PermissionAdded
- PermissionRemoved
- PermissionUpdated
- PermissionReplicated
- UpdateUser
- AdminPageAccess
- ImpersonateOn

Content List Permissions

When setting up permissions in the Content List on a production system you want to grant only the minimum permissions required. For instance, users don't need to

be explicitly given View or Modify permission to form instances to participate in a process task. The system will automatically grant them the appropriate level of permission on the form instance to complete a task that is assigned to them. When the user completes the task, the configured permissions are then used for that form instance.

Permissions shouldn't generally be configured on individual objects in the Content List, except in special cases that require that level of granularity. Normally, permissions should only be configured at the folder level. Preparation and thought is necessary during the initial implementation to ensure that you structure the objects across folders based on object types and how permissions will be established.

For example, don't create a folder that contains all Knowledge Views and Reports and then configure the permissions on each individual object to identify who is able to run it. Instead create multiple folders and group the objects by the type of access required. For instance, you might create an “Admin Knowledge Views” folder that only administrators can access, and a “Knowledge Views” folder than can be accessed by other end users. Segregating the access by folder will assist in the deployment of new objects from a non-production system to your production system, because:

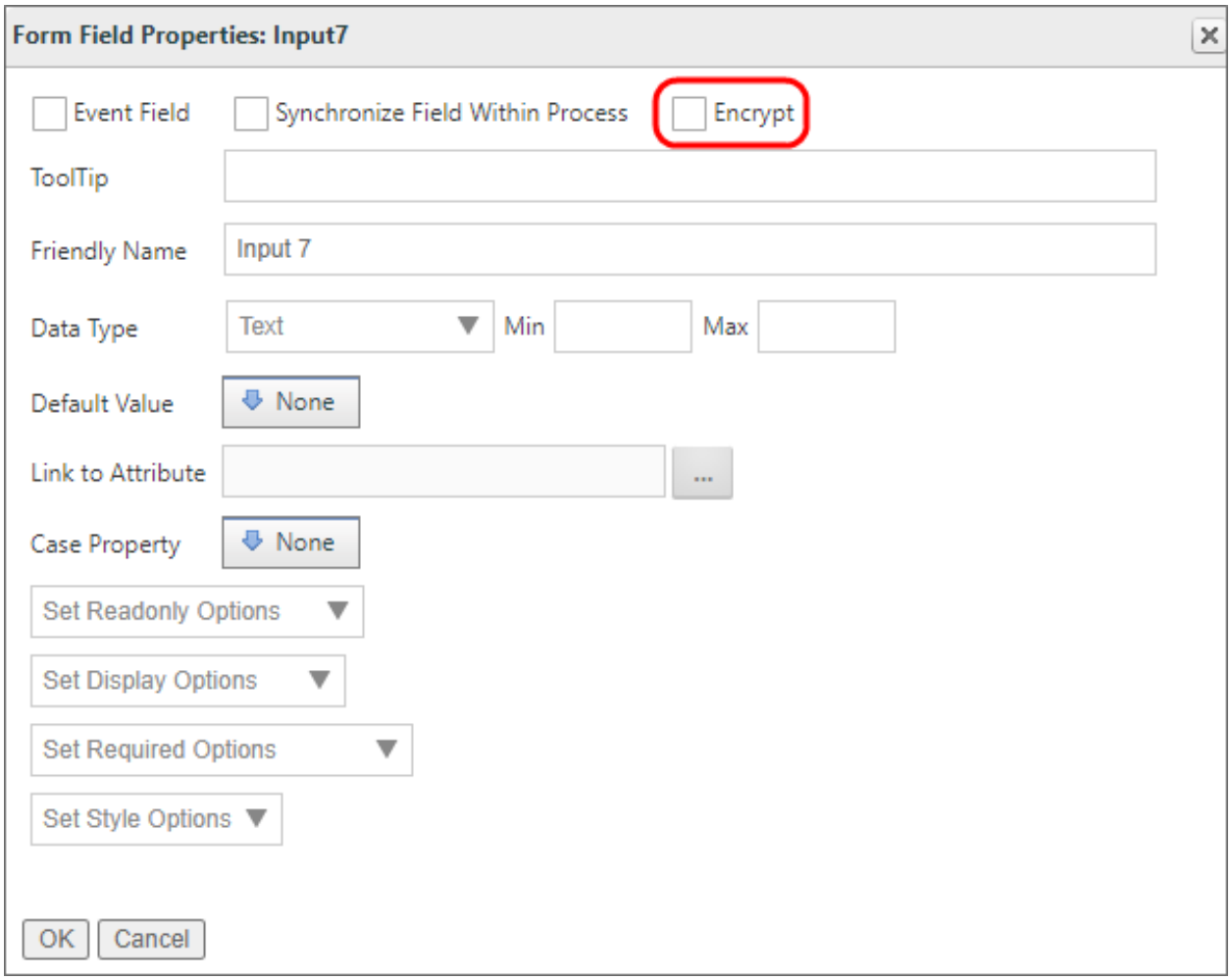
1. Imported objects inherit the permission of the parent folder into which objects are being imported, and
2. Permissions aren't carried across in the XML export.

Your production system shouldn't grant delete permission to any users. Instead, have a limited set of partition administrator users, as partition administrators can perform functions in the Content List regardless of its configured permissions. Users should only login with these partition administrator user IDs when performing specific administrative functions, like deleting objects or importing XML files from a non-production system.

Data Encryption

Process Director supports the use of Transparent Data Encryption (TDE) in Microsoft SQL Server to encrypt all data in the Process Director database. BP Logix recommends the use of this SQL Server option as the most effective method for encrypting data at rest.

When using Cloud installations or on-Premise installations with the Compliance Edition, you have the option to encrypt fields using the Form Field Properties settings of a control, which is located in the Form Controls tab of every Form definition. Simply select the Encrypt check box. This option enables you to encrypt individual field values for installations that do not implement TDE.



Checking the Encrypted Field check box will cause Process Director to save the field value in an encrypted format in the database. You may wish to consider this for fields that contain passwords, PII, HIPAA, or other sensitive data. Keep in mind, however, that you may not be able to search for data stored in encrypted form.

Storing Attachments

You have two choices when it comes to storing documents, such as file attachments, in Process Director: internal database or file system storage.

The default method of document storage is to store the documents in the Process Director internal database, in a binary field. For most users, this is the recommended option, for a number of reasons.

- Backing up the Process Director database automatically backs up the stored documents, so the number of required backups is reduced.
- Outside users can't inadvertently access the documents outside of Process Director to alter, move, or delete them.
- Performance, in most cases, will be superior, because there is no need to access files on separate network locations, and drag them across the network.
- Document administration can be performed from a single location.

For some users, however, there are use cases where storing documents in a separate file system makes more sense, such as:

- You store very large files, such as high-definition video files. It is more efficient to store these files on a File System, rather than to transform them to binary database storage and back, which imposes performance delays that file system storage doesn't.
- You have a need for the files to be accessed from outside of Process Director.
- You have a large number, i.e., in the hundreds of thousands, of documents, which greatly increases the time needed for a database backup.

Obviously, when documents are stored on a separate file system, you must perform two backups, one of the database, and one of the file system. And, of course, it may be possible for malicious users to get into the file system and alter or delete the documents, so extra security considerations apply. There are configuration options in the administrative settings that allow you to control the size of documents that should be stored on the file system. This is useful if you only want to store very large documents external to the database but maintain the advantages of using the database storage for all other documents. However, it is recommended to decide on one approach or the other, and if storing files on the file system, set the size to `0` so that all documents binaries are stored as files.

Ultimately, the decision whether to store documents in process Director or a file system depends on your use case, the size and number of documents you expect to store, your network bandwidth, and the backup requirements that will be imposed under each alternative. This is decision best made in consultation with your internal IT professionals.

Whichever choice you make, Process Director will handle the stored documents in a way that is transparent to your users.

There is a built-in Process Director function that enables documents to be moved according to the current configuration of the system. You have to go into “debug mode” in the client browser and navigate to the IT Admin area's Troubleshooting section. There will be a link that will move documents back/forth from the file system to the database. This can take quite some time, and if canceled in the middle, it can be restarted and it will continue safely where is left off.

Some internal documents will always be stored in the database regardless of the installation settings, for example, form definitions.

Test Server Methodology

Process Director can be installed on a development or test/staging server to create and test applications without affecting your production environment. Applications can be exported from a test server and imported to the production server. See the Importing/Exporting Content topic in the Implementer's Reference.

The safest option is to create a new, blank database for your test installation. You'll, once the test server is set up properly, have to manually export all of your existing object definitions from the Production server to the Test server.

When you want to set up a Test server, follow these steps:

1. Create a new, blank Test database
    a. Create a blank database on SQL Server to host your test installation.
2. Ensure that the database settings for Process Director are using the connection string for the test database and not the production database.
3. Run the test server diagnostics using the online Administrative functions under Troubleshooting.
4. Enter the test license key and token again on the test server in the online Administrative section under the Installation > Licensing.
5. Navigate to the installation tab in the Administration section and ensure the Interface URL in the Installation Settings is pointing to the test server (if any values were configured).
6. Turn off email notifications on the test server using the Administrative section and changing the Email Notification option in the Installation tab.
   This setting is important because emails sent to users from the test system may be confusing. The other option is to replace all email addresses in the database with another email address for testing. This can be done by setting the TestUserEmails custom variable in the installation's vars.cs.ascx file. This variable allows you to route all process emails (task list emails, notifications, etc) to a single user. Use this setting with caution. It should only be used on non-production systems to test processes and Forms.
7. To test the system by logging in as different users, turn off security for all users on the TEST system at the database layer, and set the `fTestMode` variable to "true" in the vars.cs file. In this mode, Windows Integrated authentication is disabled. This mode allows anyone to log into the server without a

password. Use this setting with caution. It should only be used on non-production systems to test processes and Forms.

8. You can now export Content List items from the Production server, and import them into the Test server.

For test servers, if you enable users to recover passwords, you should be aware that, when using the TestUserEmails custom variable, password reset emails will ***not*** be sent to the configured Test User Email address. Exempting the Test User Email address is done for security purposes, because anyone with access to the test email account would gain the ability to access the account of any user that submits a "Forgot password" request. Password reset emails are, instead, sent directly to the requesting user.

Custom Variables Code Sample

```
public override void SetSystemVars(BPLo-
gix.WorkflowDirector.SDK.bp bp)
{
    bp.Vars.fTestMode = true;
    bp.Vars.TestUserEmails = User.GetUserByUserID(bp, "my_
test_id");
}
```

After the above steps are completed, you can modify existing Processes on the test system and export them. The exported Processes can be then imported into the production server.

Using Production Data

When you are ready, you have an additional option to use production data on your Test server by importing your existing Production database into the Test database. This will provide an exact mirror of your Production installation, however, it may expose sensitive data to users in the test environment.

Exercise extreme caution when selecting this option, to avoid a breach of sensitive data.

To reproduce your production data on the Test database for Process Director **v5.44.700 and higher** follow the steps below. For versions **below v5.44.700**, steps 1, 3, 6, and 7 are not needed, as those version do not implement AES encryption.

1. Be sure web access to the test system is disabled during this process to prevent any users from accessing the system during this process.
2. Backup the database on the test server and save Process Director license key and token.
3. Backup your existing bpProperties.xml file in the App_Data folder.
    a. Save the value for Database Connection String from /admin/db.aspx. Using the UI allows you to view the un-encrypted value.
4. Turn off email notifications on the test server by setting the TestUserEmails flag in the installation's vars.cs.ascx file. This variable allows you to route all process emails (task list emails, notifications, etc) to a single user. Do this prior to copying over the Production database to ensure that users don't receive a plethora of notifications from the Test database when the import is complete.
5. Copy the Process Director production database to the test server. This WILL overwrite your existing test database, if applicable. All of your existing process director objects in the Production database will be available in the Test database.
6. Copy the sEncryptionKeyDoNotModify and sCheckValue XML elements from the production bpProperties.xml to the test server bpProperties.xml.
    a. You will need to set all the following values to empty (if used) and then reconfigure them in the administrative UI because you are changing the system's encryption key: sSMTPPassword2, sSMTPSecret, sSMTPClientID, sSMTPTenantID. Once configured, these values can be restored using the IT Admin pages AFTER saving bpProperties.xml.
7. After saving bpProperties.xml, restore the value of the Database Connection String from 3(a) using /admin/db.aspx page.
8. If you are storing documents on a Windows file system instead of the Process Director database, ensure you make a copy of that file system to the appropriate location for the new system.
9. Copy any other Custom variables you are using in the Production installation's custom vars file to the Test server's custom vars file. Do NOT copy the vars file itself, just the text of the vars settings from the Production vars file to the Test vars file.

10. Copy any other customized process scripts or other file-based customizations from these production directories to the test server:
    `\Program Files\BP Logix\Process Director\website\custom\*.*`
11. Re-enter your license key information on the Test Server.

You may wish to re-import the production data on a recurring basis, like once a year, or once every six months, so that changes to existing Processes can be tested based on a recent copy of the Production environment.

Working with BP Logix Technical Support

BP Logix maintains a technical support system that operates primarily through the BP Logix support site. All technical support tickets should be submitted through the support web site.

There are a some best practices to keep in mind when submitting a support ticket. These practices will make it easier for our support specialists to help you, and reduce the amount of time it takes to resolve your issue.

- Process Director is updated on a regular basis, with both formal releases, and patch updates. These patches may contain resolutions to the issue you are experiencing, and updating the product will eliminate the issue. If you are on an older version of Process Director, then first upgrade to the most recent version, which is available on the Downloads page of the BP Logix support site, and see if the upgrade fixes the issue you're experiencing. In general, you should always be using the most recent version of the product.
- When describing your issue, please remember that our support specialists don't know how your project is configured, the nature of your processes, or what you are trying to accomplish. The only thing they know is what you tell them in the Description field, so please provide them with adequate information to understand the issue. Doing so will help minimize the amount of back and forth it will take for the support specialist to understand your issue.
- Be sure to attach any relevant screen shots or, if the issue generates a Process Director error, attach your log files to the ticket, as described in the Logs topic.
- It is extremely helpful if you can create a small sample project that replicates your issue, export it from Process Director, and attach the exported XML file to the ticket. There is a good chance that the support specialist will ask you for such a project, so attaching it when you first submit the ticket will eliminate that round of troubleshooting conversation. Also, if you can't replicate the issue in a small project, then it may not be a Process Director issue, but rather an issue of improperly configuring the project where you are experiencing the problem.
- While we do wish to help you, please remember that the tech support ticket is for actual errors or other problems with Process Director itself. It isn't designed for use as a means of learning how to perform a configuration task. We are happy to schedule a Direct Assistance session with you to provide you with

instruction in how to use a particular feature or perform a complex configuration, but please reserve support tickets for actual issues with the product. Also, if you don't know how to perform a particular task, remember that we have fairly extensive documentation on how to administer Process Director and implement projects, so there's an excellent chance that your answer can be found there.

Registering at the Support Site

For most customers, two Support users are allowed to register at the support web site. You must contact your sales representative to submit names to be added to the list of Support Site users. Once BP Logix registers your user account, you'll be able to fully access the support site.

Submitting a Support Ticket

From the home page of the support site, you can create a new ticket by clicking the Submit Support Ticket button to open the Submit a Support Request form.

This button won't display unless you are logged into the Support Site.

The Submit a Support Request form contains the following fields to fill out:

Subject: A brief title to describe the nature of the issue.

Description: Please provide as detailed a description of the issue as possible.

Selected Severity: A dropdown control containing various severity levels, based on the issue's impact on your production environment. Please don't overstate the severity of the issue.

Vaccine Tracker Issue: A check box to indicate this is an issue with the pre-built Vaccine Tracker application.

Process Director Version: A dropdown from which you can select the version of Process Director you're running.

Attachments: This is a good place for you to provide additional information, such as screen shots, log files, etc.

Once you've filled out the support ticket, click the Submit button to submit it. Submitting the ticket instantly sends it to the support queue, making it visible to all of the support specialists.

This page intentionally left blank.

Index

I

L

M

N

O

P

Q

R

S

T

U

V

W

This page intentionally left blank.