

THE BUSINESS AUTOMATION TUTORIAL

BUILD MODERN PROCESSES WITH
LASERFICHE FORMS

by
EGE ERSOZ



COVERS LASERFICHE FORMS 10.2

Business Automation Tutorial

Build Modern Processes with Laserfiche Forms

Ege Ersoz

Contents

| | |
|--|------------|
| Acknowledgments | v |
| About the author | vii |
| Copyright and license | ix |
| 1 Beginning | 1 |
| 1.1 Introduction | 2 |
| 1.1.1 Prerequisites | 3 |
| 1.1.2 Conventions and terminology | 5 |
| 2 Up and Running | 7 |
| 2.1 Preparing our training environment | 7 |
| 2.1.1 Laserfiche Server | 8 |
| 2.1.2 Web Administration Console | 10 |
| 2.1.3 Web Access | 14 |
| 2.1.4 Laserfiche Forms | 21 |
| 2.1.5 Testing Our Configuration | 34 |
| 2.2 Chapter Summary | 46 |
| 3 Business Process: Customer Satisfaction Surveys | 47 |
| 3.1 Creating a customer satisfaction survey form | 49 |
| 3.1.1 Editing form theme | 56 |
| 3.1.2 Field rules | 60 |
| 3.2 Storing forms and sending emails | 63 |

| | | |
|----------|---|------------|
| 3.2.1 | Metadata templates and fields | 63 |
| 3.2.2 | Save to Repository service task | 67 |
| 3.2.3 | Exclusive Gateway | 72 |
| 3.2.4 | Email service task | 75 |
| 3.3 | Chapter summary | 79 |
| 4 | Business Process: Purchase Orders | 81 |
| 4.1 | Creating the Purchase Request form | 84 |
| 4.1.1 | Automatic calculations in Laserfiche Forms | 90 |
| 4.1.2 | Database lookups | 90 |
| 4.2 | Designing the Process Diagram | 103 |
| 4.2.1 | Purchase Approval Task | 103 |
| 4.2.2 | Submitter Revision Task | 107 |
| 4.3 | User accounts and security | 110 |
| 4.3.1 | User accounts | 110 |
| 4.3.2 | System Security | 113 |
| 4.3.3 | Business Process access rights | 114 |
| 4.4 | Publishing and testing | 116 |
| 4.4.1 | Publishing the Business Process | 116 |
| 4.4.2 | Testing the Business Process | 117 |
| 4.5 | Optimizing the Business Process | 120 |
| 4.5.1 | Preventing purchase requests for closed projects | 121 |
| 4.5.2 | Generating a unique purchase request number for each form | 122 |
| 4.5.3 | Routing the forms to the correct crew leader | 128 |
| 4.6 | Chapter summary | 128 |
| 5 | Business Process: Job Applications | 131 |
| 5.1 | Creating the job application form | 132 |
| 5.1.1 | Standardizing user input | 133 |
| 5.1.2 | Form pagination | 134 |
| 5.1.3 | Displaying a warning for closed positions | 136 |
| 5.2 | Creating the Recruiter Checklist form | 139 |
| 5.2.1 | Applicant's Basic Information | 140 |

| | | |
|----------|---|------------|
| 5.2.2 | Recruiter Screening | 143 |
| 5.2.3 | Multi-select drop-down fields | 144 |
| 5.2.4 | Advanced conditional routing | 147 |
| 5.2.5 | Merging gateways | 151 |
| 5.3 | Testing the Job Application Process | 152 |
| 5.4 | Chapter summary | 154 |
| 6 | Reporting and Optimization | 155 |
| 6.1 | Process Stages | 155 |
| 6.2 | Auto Reporting | 158 |
| 6.2.1 | Operational Dashboard | 159 |
| 6.2.2 | Performance Dashboard | 161 |
| 6.2.3 | Instances and Tasks | 163 |
| 6.3 | Forms Reporting | 165 |
| 6.4 | Process Optimization | 166 |
| 6.4.1 | Deadlines using Timer Events | 166 |
| 6.4.2 | Signal events | 169 |
| 6.5 | Conclusion | 172 |
| 6.5.1 | Guide to further resources | 172 |

Acknowledgments

Business Automation Tutorial owes a lot to the Laserfiche development team, without whom the software, and therefore this book, would not exist.

I would like to acknowledge a long list of ‘fichers who have inspired me over the years: Justin Pava, Miruna Babatie, Karl Chan, Kurt Rapelje, Alexander Huang, James Shearer, Jereb Cheatham, Alan Chan, Jonathan Kim, Jonathan Powers, Andy Wang, Cora Anderson, Steve Hackney, Trent Strong, Melissa Henley, Chris Wacker, Jeanie Conner, Eric Cressey, Tammy Kaehler, Ed Heaney, Evelyn Crofts, Sierra Ashley, Elizabeth Cunningham, Brandyn Campbell, Andrew Brown, and anyone else whose name I am forgetting due to my old age.

I would also like to thank my technical reviewer for going through the content of this book in meticulous detail and making sure everything is both correct and easy to understand.

About the author

Ege Ersoz is the author of Business Automation Tutorial. His prior experience includes a long stint at Laserfiche, first as a Software Support Engineer, then as a Sales Engineer. During this time, Ege taught numerous classes on Laserfiche at regional training seminars and conferences, and helped countless customers and Value Added Resellers get the most out of the platform. After leaving Laserfiche, Ege joined Zeno as a Senior Solutions Architect. In 2017, two of Ege's Laserfiche clients won Run Smarter awards and were recognized at [Laserfiche Empower](#). Ege is a [University of Washington](#) alumni and has a Bachelor of Science degree in [Informatics](#).

Copyright and license

Business Automation Tutorial: Building Modern Processes with Laserfiche Forms. Copyright © 2017 by Ege Ersoz. Last updated 2017-05-21 16:17:08 CT.

Everything in The Business Automation Tutorial is available jointly under the MIT License and the Beerware License.

The MIT License

Copyright (c) 2017 Ege Ersoz

Permission is hereby granted, free of charge, to any person obtaining a copy of this book and associated documentation files (the "Book"), to deal in the Book without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Book, and to permit persons to whom the Book is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Book.

THE BOOK IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE BOOK OR THE USE OR OTHER DEALINGS IN THE BOOK.

THE BEERWARE LICENSE

Ege Ersoz wrote this book. As long as you retain this notice you can do whatever you want with this stuff. If we meet some day, and you think reading it was worth the time investment, you can buy me a beer in return.

Chapter 1

Beginning

Welcome to [Business Automation Tutorial: Build Modern Processes with Laserfiche Forms](#). The purpose of this book is to teach you how to automate and streamline business processes, and our tool of choice is [Laserfiche Forms](#), which is a commercial, enterprise-grade automation framework built on the popular [Laserfiche](#) platform. Introduced in 2013, Laserfiche Forms is a relatively new edition to the Laserfiche family, but has matured very quickly and today can be found in most Laserfiche systems around the world.

This book is designed to give you a thorough introduction to business process automation, sufficient to launch you on a career as a Laserfiche expert. If you already work in the Laserfiche ecosystem, you will learn the essentials of Laserfiche Forms, including Forms Designer, Process Modeler, field, lookup and error messaging rules, and deeper customization of your forms using both out-of-the-box and custom tools. In any case, when you finish the *Business Automation Tutorial*, you will be in a position to develop fairly complex systems that are not only robust and user-friendly, but also have a noticeable impact on your organization's bottom line.

The *Business Automation Tutorial* takes an integrated approach to business process automation by building three example applications of increasing sophistication. These start with an introductory Customer Satisfaction Survey, a mid-level Purchase Request and Approval process, and a fairly complex process for the submission and review of job applications. Even though these are fairly specific processes, the emphasis throughout the book will be on general

principles, so you will have a solid foundation no matter what kinds of processes you want to automate and streamline.

In [Chapter 2](#), we will prepare our training environment by installing and configuring several Laserfiche modules and their requisite tools and applications. This chapter will lay the foundation for the rest of the book, so we will make sure to go over each configuration step in detail and explain how it fits into the bigger picture.

In [Chapter 3](#), we will create our first “production-grade” Business Process. This will contain a Customer Satisfaction Survey form that will allow users to rate the quality of the service they received while doing business with our fictional company, Acme Industries. After we get the form working, we will improve its user-experience using field rules and database lookups, and also make it look fancier in order to increase the survey response rate. Everyone loves pretty-looking surveys!

In [Chapter 4](#), we will dive into the Process Diagram, which allows specifying what happens to submitted forms. We will build a Purchase Order process with a manager approval step, configure database lookups to make it easy to use, and will also add some intro-level styling and behavior customizations using CSS and JavaScript.

In [Chapter 5](#), we will build a Job Application process. This process will have a fairly complex Process Model, including the ability to route a form to multiple departments, and the forms contained in it will have everything we will have learned up to that point, in addition to more CSS and JavaScript customizations to make sure the form is as user-friendly as possible.

In [Chapter 6](#), which will be the final chapter, we will make improvements to our Job Application process by making it analytics-friendly, learn about the reporting capabilities in Laserfiche Forms, and optimize the process further by utilizing some of the advanced features available in the Process Diagram.

1.1 Introduction

Laserfiche is an Enterprise Content Management (ECM) platform. Since its debut back in 1987, it has rapidly become one of the most powerful and popular

software systems for building rich, automated and streamlined business processes. Laserfiche is used by organizations of all sizes, some of the larger ones being Amazon, Texas A&M University, Avis Fleet Management, Emirates National Oil Company and the City of Long Beach, California. There are also many Laserfiche Value Added Resellers around the world that provide consulting, implementation and related professional services work on Laserfiche systems, making this a truly vibrant ecosystem.

One of the things that make Laserfiche so great is its ease of use. It is very common for users to be able to use a newly deployed Laserfiche system with little or even no training. The content inside the repository is organized using a nested folder structure most people feel at ease with, and the automation capabilities of the system are both transparent and powerful.

That said, quite a lot of work goes into making sure the barrier of entry of a Laserfiche system is as low as possible. Sure, tools included in the Laserfiche platform are very user-friendly, but it is how one uses those tools that determines whether a given project is successful. Over the course of my career, I have seen many Laserfiche systems that were poorly designed or not fully thought-out. Sometimes this is because the the person mapping out the process does not fully understand the requirements of each stakeholder. Sometimes it is because the implementer is inexperienced. And sometimes it is because the underlying process itself is way too complex to properly automate and streamline using software. Whatever the case, it's an all-too-common scenario to take a set of powerful and robust tools like Laserfiche and still end up with a "rat's nest".

1.1.1 Prerequisites

In order to make the most out of this book, you will need a Laserfiche system that is on version 10.2 or later and is licensed for Laserfiche Forms. Specifically, we will be using Laserfiche Forms 10.2 throughout the tutorial, and it requires Laserfiche Server 10.2. Prior versions do have many of the features we will cover, but not all. And even when the features are there, they may work slightly differently, or be labeled differently. To prevent any confusion, you should double-check your Laserfiche version and upgrade if necessary. I

realize the latter is easier said than done, but it is for the best.

You will also need Microsoft SQL 2012 or later. While Laserfiche still supports MSSQL 2008 R2, there are a few important features in MSSQL 2012 and later that we will make use of in our exercises. If your organization does not have Microsoft SQL Server, don't worry: there is a free version, called the Express Edition, available. Regardless of which route you take, make sure to also install the corresponding version of Microsoft SQL Management Studio.

There are no skill-related prerequisites. If you are a daily Laserfiche user, great. You will feel at home with many of the repository-related concepts and features we will use. Other than that, familiarity with HTML, CSS and JavaScript would certainly be helpful, but it is not necessary. An understanding of relational database concepts will make certain parts easier, but that, too, is optional. Essentially, this book makes no assumptions about your level of knowledge. Everything will be explained from the ground up.

Box 1.1. Laserfiche Forms Basic

As of this writing, Laserfiche is preparing to release version 10.2.1, which will have a “Basic” edition of Laserfiche Forms available in all licensing platforms. Please note that, while a good start, Laserfiche Forms Basic does not have some of the more advanced features we will make ample use of in this tutorial. You are definitely more than welcome to read along however. At the very least, it might help you decide if you need to upgrade to Laserfiche Forms Professional.

Training Environment

While not a prerequisite per se, it is strongly recommended to follow this tutorial in a training environment that is separate from your organization's production Laserfiche system. This gives you maximum flexibility and ensures that you don't accidentally publish your Funny Cat Picture Submission Form to the entire organization, or “break” something and make your boss angry.

The good news is that setting up a training environment, while a bit time-consuming, is not very difficult. We are going to cover installation and configuration in [Chapter 2](#), but to give a quick summary, you will need a fresh instance of Laserfiche Forms as well as a separate Laserfiche repository. The latter can be on the same Laserfiche Server as your production repository. Mainly, we want to make sure we don't clutter up the production repository with the forms we build and submit throughout the tutorial.

Box 1.2. Using your production environment for training

If for some reason you are not able to set up an isolated training environment, that's okay. You just have to make sure you don't accidentally edit any existing Forms Business Processes, and also be a bit careful about where in the production repository your submitted forms are stored.

1.1.2 Conventions and terminology

Most conventions and terminology used in this book are self-explanatory, but this section will cover some that may not be.

Processes in Laserfiche Forms are called **Business Processes**. These are different from Laserfiche Workflow's Business Processes, which are workflows that can be initiated manually by users. When we say Business Process, we mean the ones in Laserfiche Forms. Laserfiche Workflow is not covered in this book.

When we are in the Form Designer, we need to make sure we are adding or modifying fields correctly. You will often see tables containing field specifications, where each column represents a field option, like this:

| Field label | Type | Regular Expression |
|-------------|-------------|--------------------|
| Name | Single Line | cell3 |
| Phone | Single line | 3-3-4 |
| Department | Drop-down | cell3 |

If a field's option is not specified in the table, you can assume that it should be left at its default. For example, the above table does not have a Required or Read-only column, so both of those should be off for all three fields.

Formatting conventions

- Links, buttons and option names are represented using a green monospaced font, for example: `Administration`, or `Required`
- Specific values, as well as names of Field and Process Variables, are encapsulated in quotations, for example: `"localhost"`. In these cases, ignore the quotation marks and just enter the value they encapsulate.
- Longer inputs are represented using text boxes, for example:

```
Please fill out the following form and route it to the appropriate department.
```

- Field names are displayed in bold, for example: **First Name**
- Various terminology may be capitalized as appropriate

We will also be writing a little bit of HTML, CSS, JavaScript and SQL as we go through the tutorial, as well as some SQL. This code is included in code listings, such as the JavaScript snippet below:

```
function helloLaserfiche() {  
  console.log("Hello, Laserfiche!")  
}
```

Chapter 2

Up and Running

Even for experienced Laserfiche administrators and implementers, installing and configuring Laserfiche requires attention to detail. The software is very well-documented, but every environment in which it is deployed is different. Subtle differences in configuration can result in big discrepancies in software behavior, and sometimes even a single typo can lead to an inoperational system.

In this chapter, we are going to do two things. First, we will install and configure a Laserfiche system from scratch. This system will consist of three components: Laserfiche Server, Web Administration Console, Web Access, and Forms. Once everything is installed and configured, we will create a very basic Business Process in Laserfiche and make sure we are able to submit forms through it. By the end of the chapter, we will be in a good place to dive deeper into Laserfiche Forms and create more complex processes.

2.1 Preparing our training environment

To perform the installations in this chapter, I will be using the Laserfiche Rio 10.2 package, which contains everything we need.

Box 2.1. Licensing your installed modules

Laserfiche Server and Laserfiche Forms installations will prompt for a license. If your organization is on Laserfiche Rio, you can activate them against your Laserfiche License Manager or Laserfiche Directory Service. If you have Laserfiche Avante, or if you are a Laserfiche VAR, you will need the product's activation key, which you can [acquire from the Laserfiche Support site](#). Speak to your Laserfiche administrator if you don't have a Laserfiche Support Site account or if it does not have the necessary permissions to view product activation keys.

2.1.1 Laserfiche Server

We will begin by installing Laserfiche Server. As you may already know, Laserfiche Server is the central component of Laserfiche, and is usually the first piece that is installed.

Laserfiche Server has several different components, but we are not going to go over them, as they are not pertinent to this book. We will perform the installation using the default options.

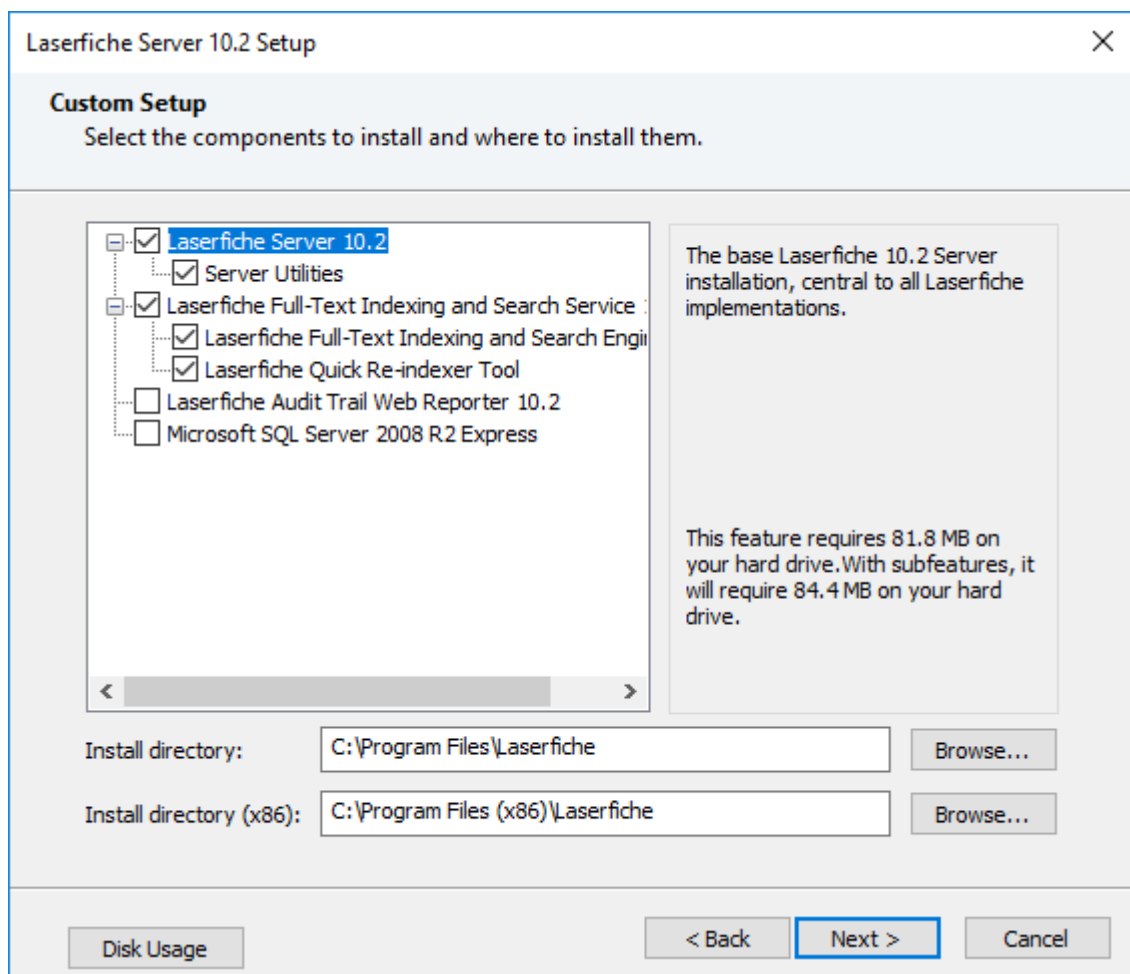


Figure 2.1: The components we will be installing.

Box 2.2. Using SQL Express

If your organization does not have Microsoft SQL Server, you will need to install the Express Edition. Laserfiche Server comes bundled with Microsoft SQL 2008 R2 Express, which is available as a component when you go through the installation wizard, but you will need Microsoft SQL 2012 Express or later. You can download that from [here](#). Click the Download button, select

“ENU\x64\SQLXPRT_x64_ENU.exe” and click Next. Your download will begin shortly after.

Box 2.3. Laserfiche Service Settings

The installation will ask which Windows account you want the Laserfiche Server service to use. If you are installing everything on the same computer (such as a virtual machine), then you can keep it as Local System. But if your Laserfiche Server will be accessing a SQL Server that is on another server on the domain, you will need to change it to a domain account that has sufficient privileges to the SQL Server. You can find more information [here](#).

2.1.2 Web Administration Console

Next, we need to install the Web Administration Console so that we can create and configure our Laserfiche repository.

The installation is quite straight-forward. The only thing you may need is to make sure you have Internet Information Services (IIS) installed. You will need that for Laserfiche Forms and Laserfiche Web Access anyway, so you might as well install it now if it is not already.

Once installed, it is time to navigate to localhost/webadmin. The Web Administration Console allows you to administer Laserfiche Servers, Quick Fields Agents and Distributed Computing Clusters. We aren't worried about the latter two for now, so click Laserfiche Servers.

You should see a mostly empty page. Since it is a fresh install, the Web Administration Console is not yet “aware” of any Laserfiche Servers. Let's fix that. First, we need to sign in as a System Manager.

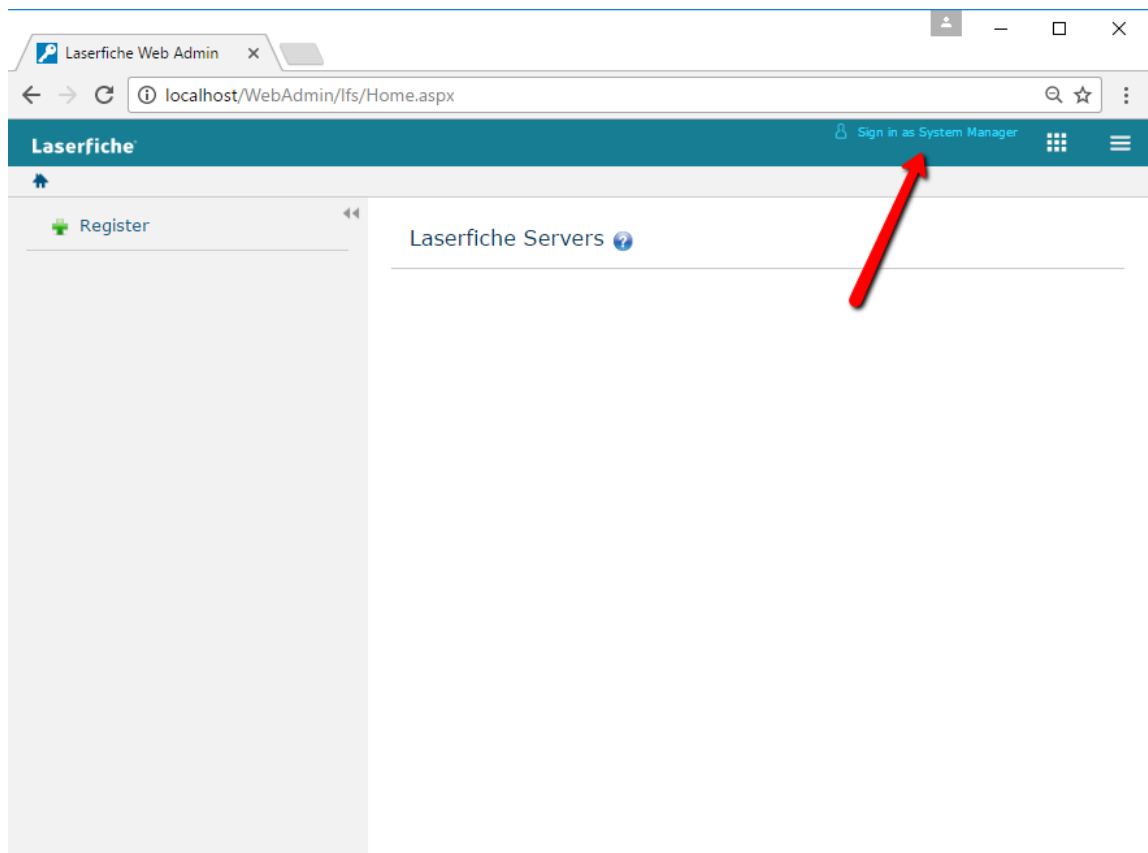


Figure 2.2: Signing in as a System Manager.

A System Manager is a Windows user that has administrative privileges on the machine on which Laserfiche Server is installed. In our case, the Laserfiche Server is installed locally, so we can use the credentials of our local admin user and sign in.

Next, we will click the Register button to register our Laserfiche Server. Type “127.0.0.1” into the Server field and click OK. That’s the IP address of our local machine.

Since it’s a brand new Laserfiche Server, we don’t have any repositories hosted on it yet. We are going to create one now. Mouse over the server node on the left pane, then click the small down arrow icon. A menu will appear. Select “Create new repository”.

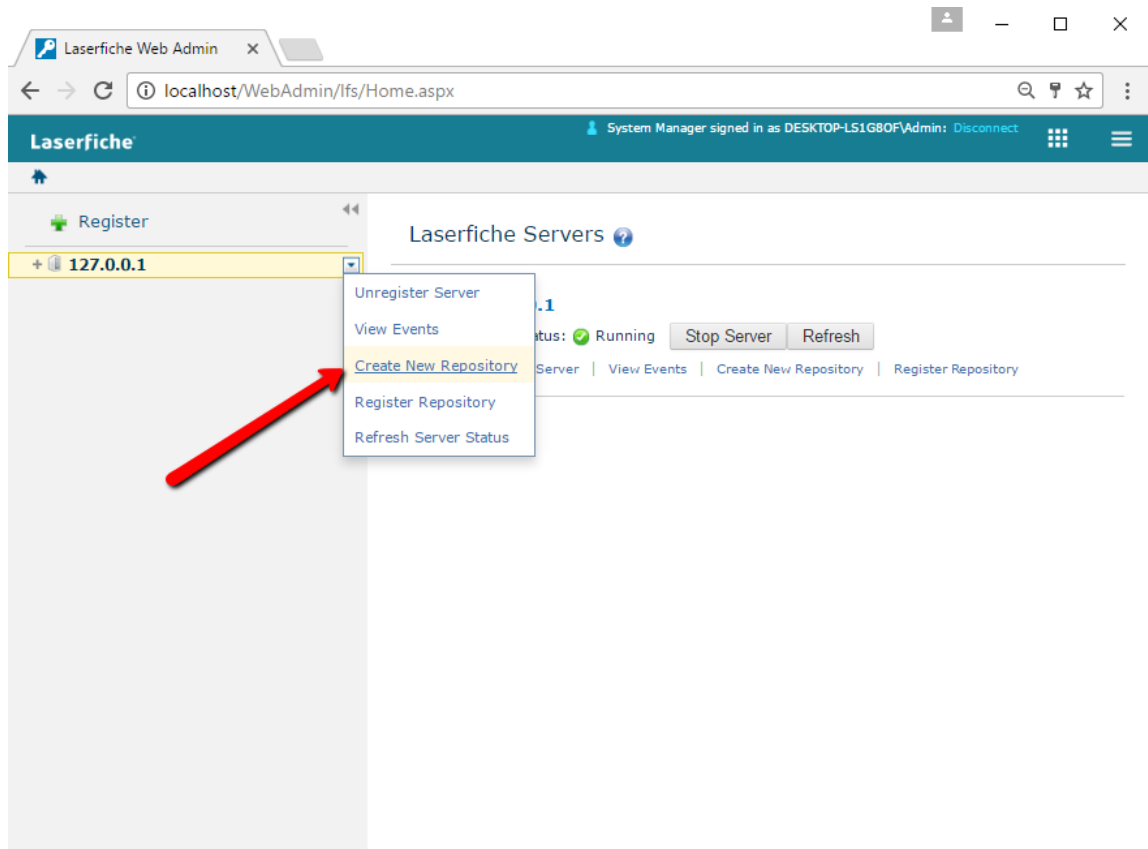


Figure 2.3: Creating a new repository.

A dialog box will open and it will walk us through the repository creation process.

Repository Settings

For our Repository Name, we will use “LFTraining”. The Repository Path can be located anywhere on the machine, but to keep things organized I generally store mine under “C:\Repositories\LFTraining”. This is where Laserfiche Server will store the newly created repository’s various components.

Server and DBMS

In the second step, we will select Microsoft SQL Server from the DBMS drop-down, then enter our SQL Server settings. If we installed Microsoft SQL Express on this computer, the Server field should say “localhost\sqlexpress”. If you have a Microsoft SQL Server installed elsewhere that you can use, enter its information instead.

Box 2.4. SQL Instances

SQL instances provide a method of compartmentalizing a SQL Server’s various databases. They are often used to optimize resource usage on a database server. “Full” editions of Microsoft SQL Server sometimes don’t have instances, in which case we would simply enter the name or the IP address of the SQL Server machine. SQL Express always installs a SQL instance though, and the default instance name is “sqlexpress”, which is why we are using the “localhost\sqlexpress” nomenclature.

Then, we need to specify which authentication method we want Laserfiche Server to use in order to access the SQL Server instance. Everything is installed locally, so we will leave this at “Use Windows Authentication”, which will make Laserfiche Server use the “Local System” user.

Finally, we need to provide a name for our repository database. It’s a good idea to keep the name same as that of the repository, which makes it easy to identify. So let’s enter “LFTraining” and click Next. When it asks us if we want to create the database, we will click OK.

Confirmation

We are going to skip the “Search Settings” step, since this tutorial does not cover the Laserfiche Full Text Search Service. In the final step, we will verify our settings before clicking Finish.

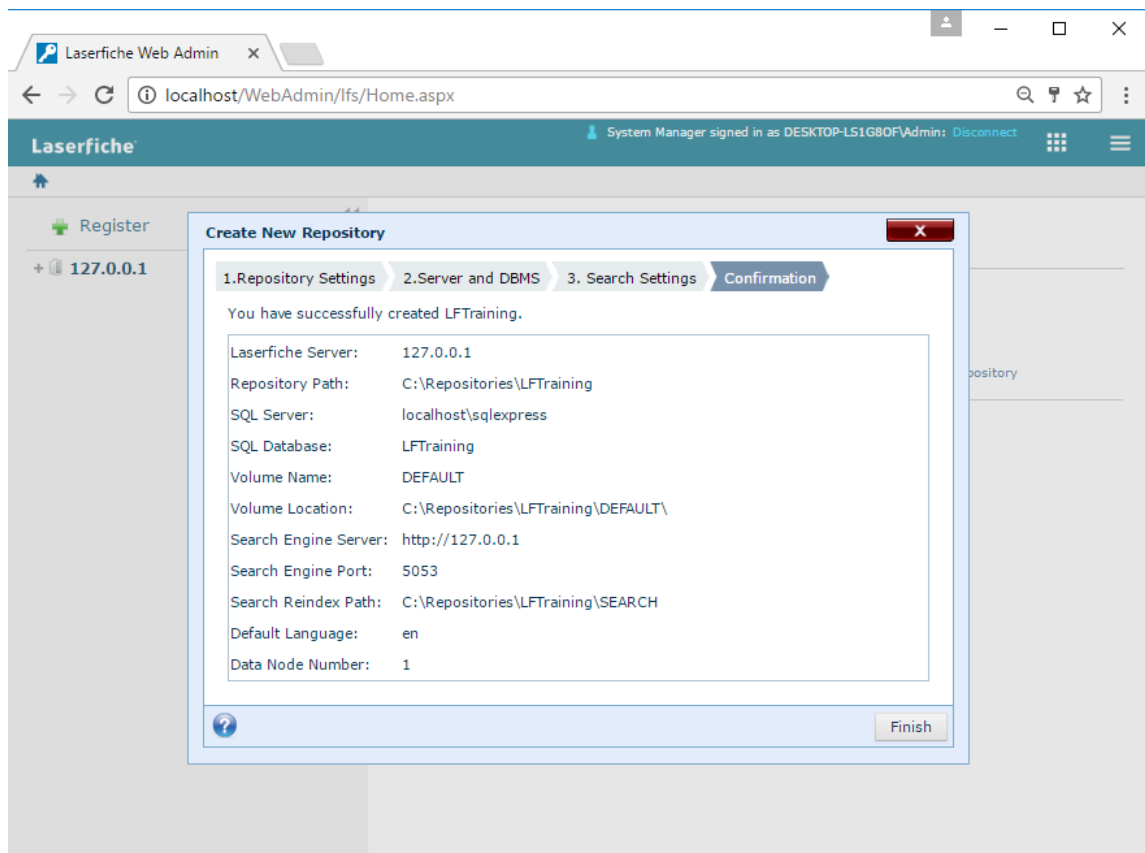


Figure 2.4: Final settings for our new repository.

Once the repository is created, we should be able to expand the server node, click the repository and log in with the default Admin user (no password).

2.1.3 Web Access

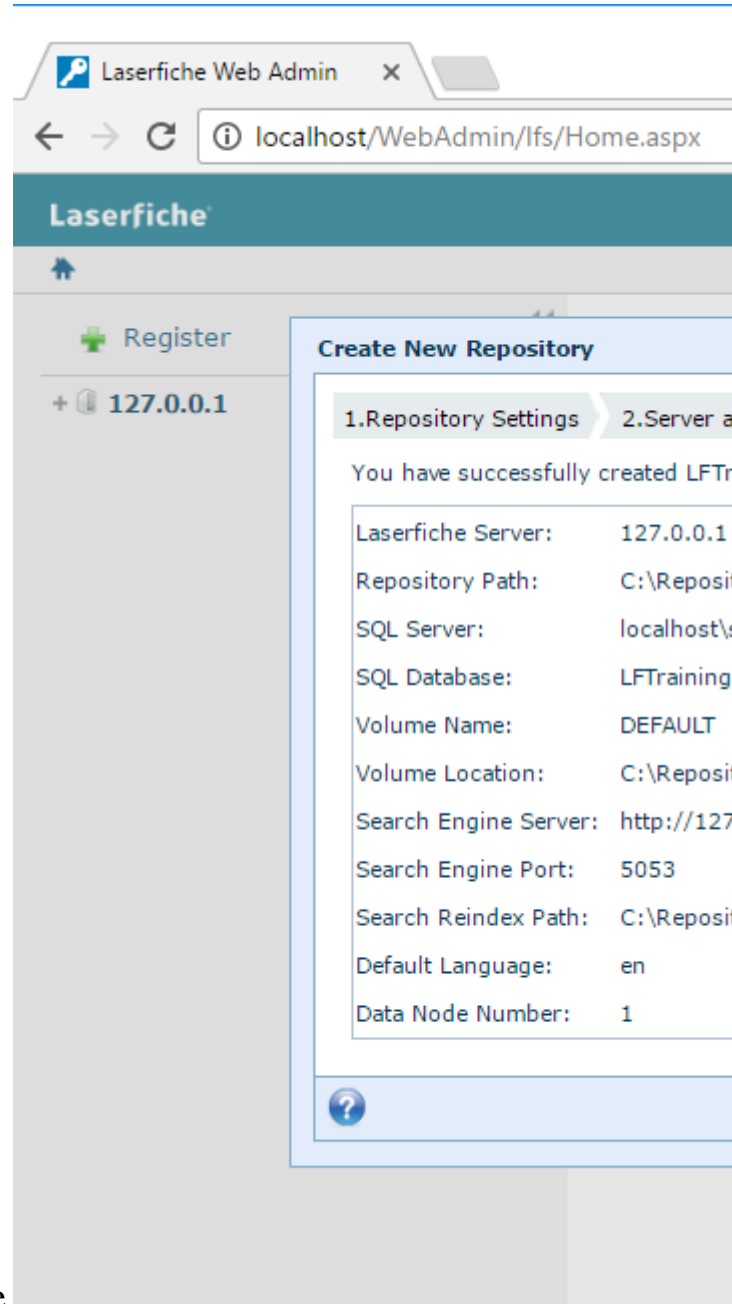
We are done with repository creation. Now, we will install Web Access, which will give us the means to log into the repository as a user, rather than an administrator. With Web Access, we will be able to browse around the repository (which will be initially empty), we well as perform more specific administrative actions.

Box 2.5. Repository Administration vs. Management

That last sentence might have been confusing. You may be thinking, “Hold on, perform administrative actions? I thought that’s why we installed the Web Administration Console.” You are right. As of this writing, Laserfiche repository administrative actions are split between Web Administration Console and Web Access. We won’t cover the details here, but from this point forward all of our administrative work will be performed inside the latter.

Let’s go ahead and run the Web Access installation. Note that Web Access is referred to as “Web Client” in the installation menu.

Similar to the Web Administration Console, Web Access also has a fairly straightforward installation. You can leave all settings on their defaults and simply click through the wizard. You will, however, need to provide an activa-



tion key, so have it handy. Once done, the checkbox will be checked, and it will launch when you click Finish.

Once in the Configuration page, we will click “Add new repository” and add the LFTraining repository, as shown below.

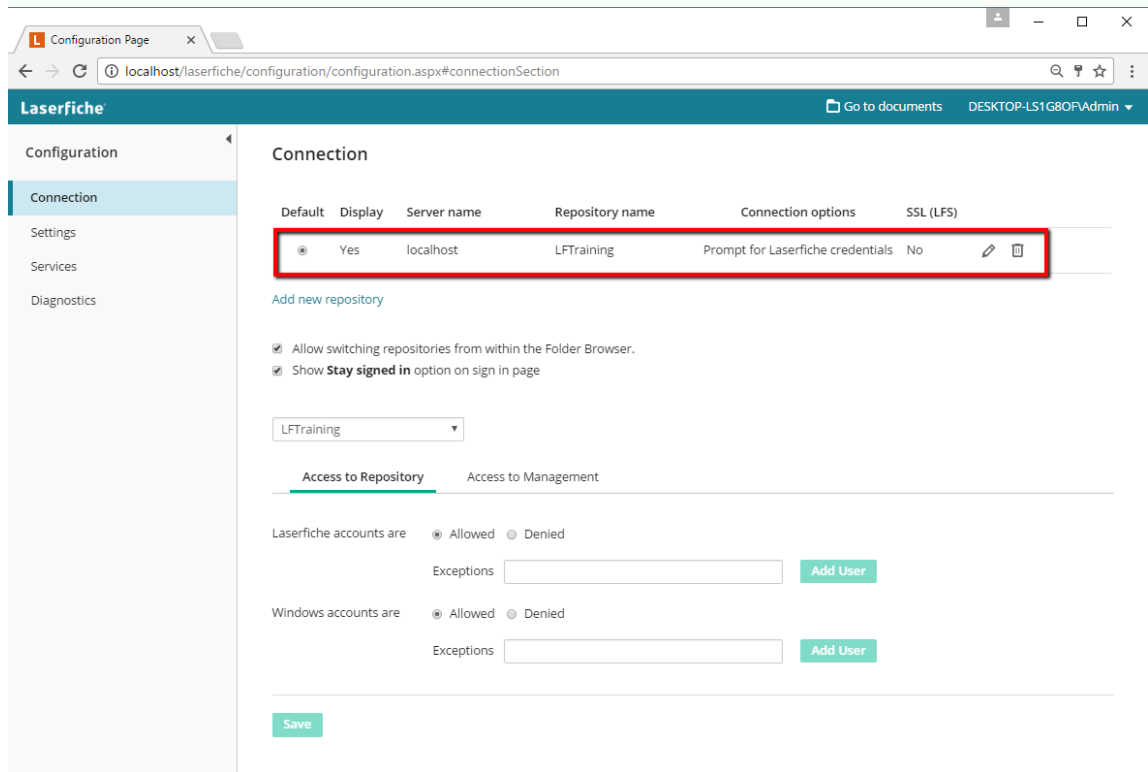


Figure 2.5: Adding a new repository to Web Access Configuration.

That's it! Now when we click on the "Go to documents" link on the navigation bar, we can log in (as admin and with no password) and see a fresh new repository.

Repository Management

Before we proceed to installing Laserfiche Forms, we will create a new user account that it can use to log into the repository. On the top right corner, click on the username (ADMIN) and select Management.

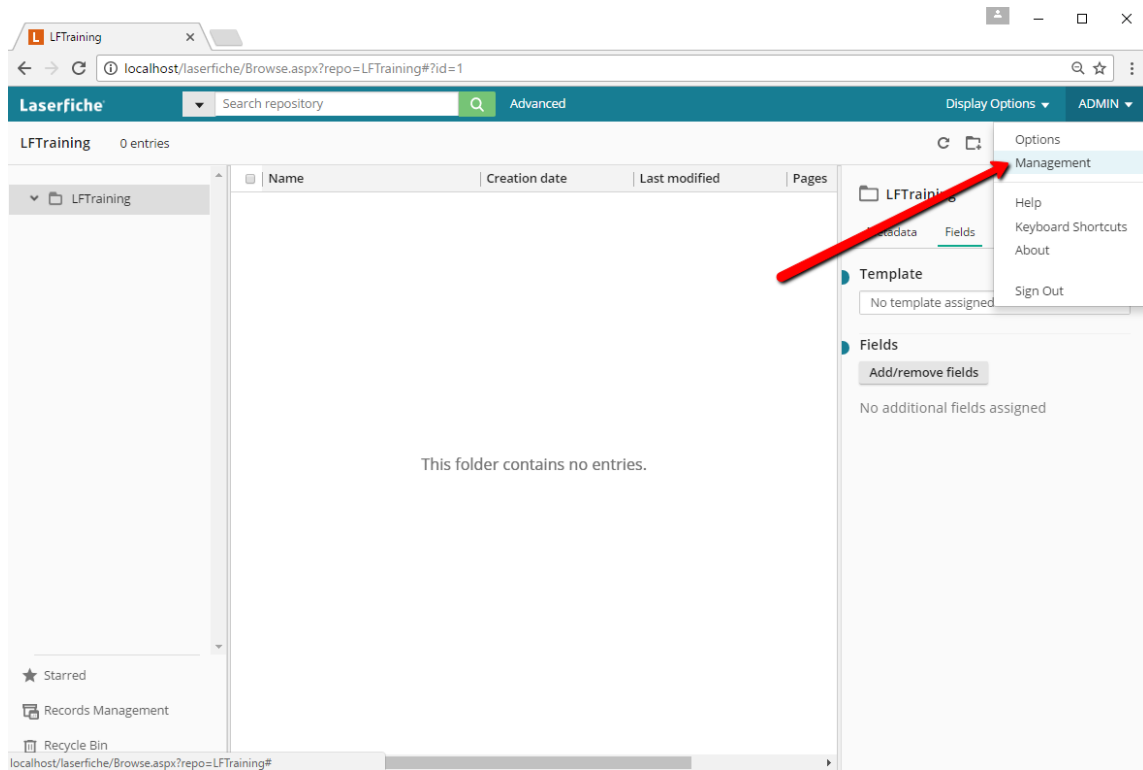


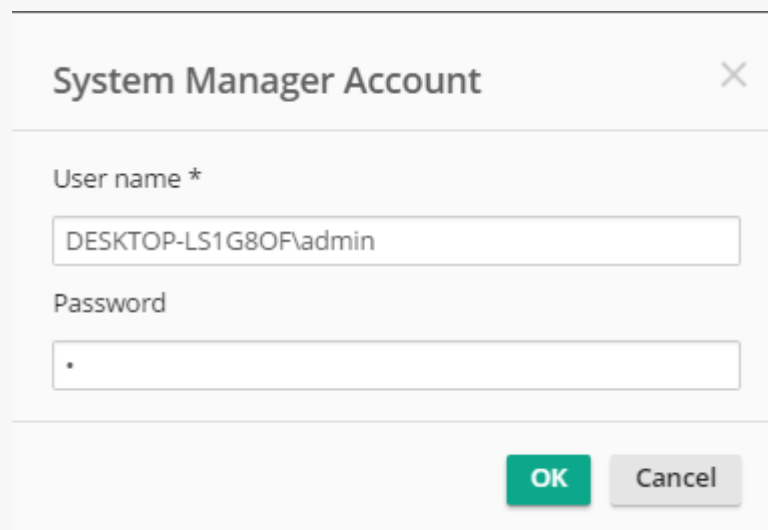
Figure 2.6: Accessing the repository management page.

The Repository Management section of Web Access allows us to perform several critical administrative actions related to the repository. We can create new users and groups, manage metadata, change record management settings, configure LDAP Server Profiles and Microsoft Outlook integrations and more. For now though, let's start by creating a new user account by clicking the Users item on the left menu.

Currently, the only user account we have is the Admin account, which is a special built-in account. It can do a lot of things, but it does not have a Named License by default, which means it cannot be used by other Laserfiche services (such as Laserfiche Forms) to connect to the repository. So we will create a new user account. First though, we need to sign in as a System Manager again using the navigation bar link that says "Sign in as System Manager".

Box 2.6. System Manager user name

The username in the System Manager Account dialog needs to be in the “computername\username” format. You can find your computer name by opening File Explorer, right-clicking on This PC, and selecting Properties. For example, my computer is named DESKTOP-LS1G8OF and I am logged into Windows as a user called “admin”, so I need to enter my username as “DESKTOP-LS1G8OF\admin”:



The image shows a dialog box titled "System Manager Account". It has a close button (X) in the top right corner. Below the title bar, there are two input fields. The first is labeled "User name *" and contains the text "DESKTOP-LS1G8OF\admin". The second is labeled "Password" and contains a single asterisk "*". At the bottom of the dialog, there are two buttons: "OK" (highlighted in green) and "Cancel".

Figure 2.7

Now let’s create our new user by clicking the plus icon on the top right corner. From the dropdown menu, select Repository. This will be a Repository user, meaning it will authenticate with a username and password.

Let’s name our user something descriptive: we will call it “Forms”. We will not assign it to any groups (since we don’t have any yet), but we will make it a Full repository named user, and give it a password. For training environment, I make passwords the first letter of the username so they are easy

to remember. In this case, it will be a lower-case “f”.

After creating the user, we need to do one more thing, which is provide it with the Manage Trustees privilege. This will allow Laserfiche Forms to use this user account to synchronize its list of user accounts. Let’s go ahead and select the newly created Forms user. Then, on the right-hand pane that opens, switch to the Rights tab, check the Manage Trustees checkbox and save.

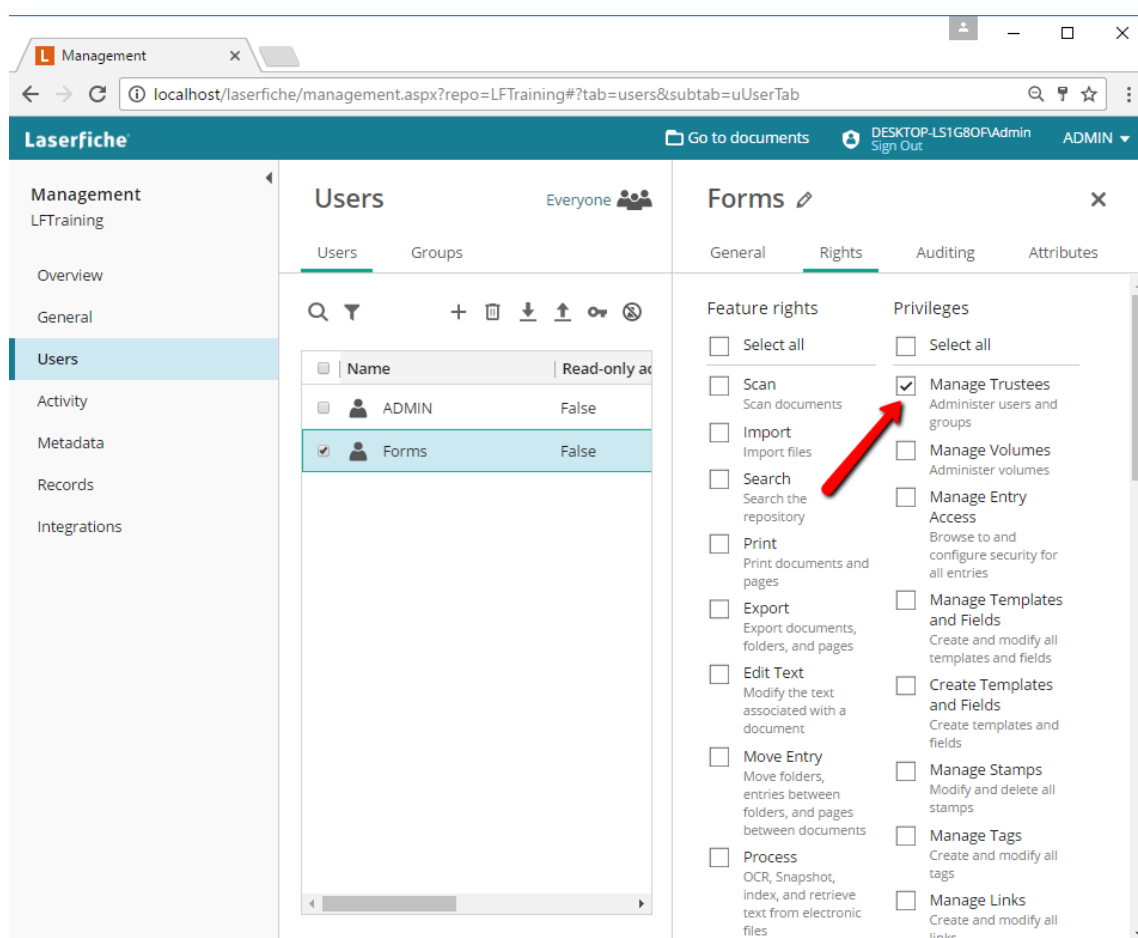


Figure 2.8: Granting the Manage Trustees privilege to the Forms user.

Great! We are finally ready to install and configure Laserfiche Forms.

2.1.4 Laserfiche Forms

This section is going to be fairly long. Laserfiche Forms has a lot of configuration options, and in order to be thorough we are going to cover most of them in detail. We will also install and configure an email server, and create a new account on Braintree, which is an online payment processor.

Let's go ahead and start by running the Forms 10.2 installer. Like the previous two modules, the installation is easy. However, note that the information on [Box 2.3](#) applies here as well for the Laserfiche Forms System Settings step. At the end of the installation, we will launch the Laserfiche Forms Configuration.

Database

First, we need to configure the database that Forms will use. This is similar (and in fact identical) to how we configured the repository database in the Server and DBMS part of the Web Administration Console section. As of this writing, Laserfiche Forms only supports Microsoft SQL Server. We will enter our server name as "localhost\squlexpress", keep the authentication setting at Windows authentication, and enter "Forms" as our database name. Then we will click Save.

Uh oh! If you performed the same steps that I did, you may have received an error message, shown in red at the top of the configuration page:

```
CREATE DATABASE permission denied in database 'master'.
```

What this means is that the IIS application pool that Laserfiche Forms uses does not have sufficient permissions. But why is that? After all, we entered Local System as the service account when we performed the installation, and the Local System account should not have any permission problems when it comes to local resources.

Here is what is going on. Laserfiche Forms has two different components. The first component is the web application pool, which makes up the primary component of the Laserfiche Forms application. The second component is the Laserfiche Forms Routing Service, which is responsible for determining what

happens to forms that are submitted. The service account we configured during installation was for the latter. In order to configure the account for the former, we will go into Internet Information Services (IIS) Manager. We can find that by opening the Start menu and typing “IIS”.

Once IIS Manager launches, we will expand the computer name node on the Connections pane (left side), and select Application Pools. Then, we will select the FormsAppPool and go to the Advanced Settings section on the right side.

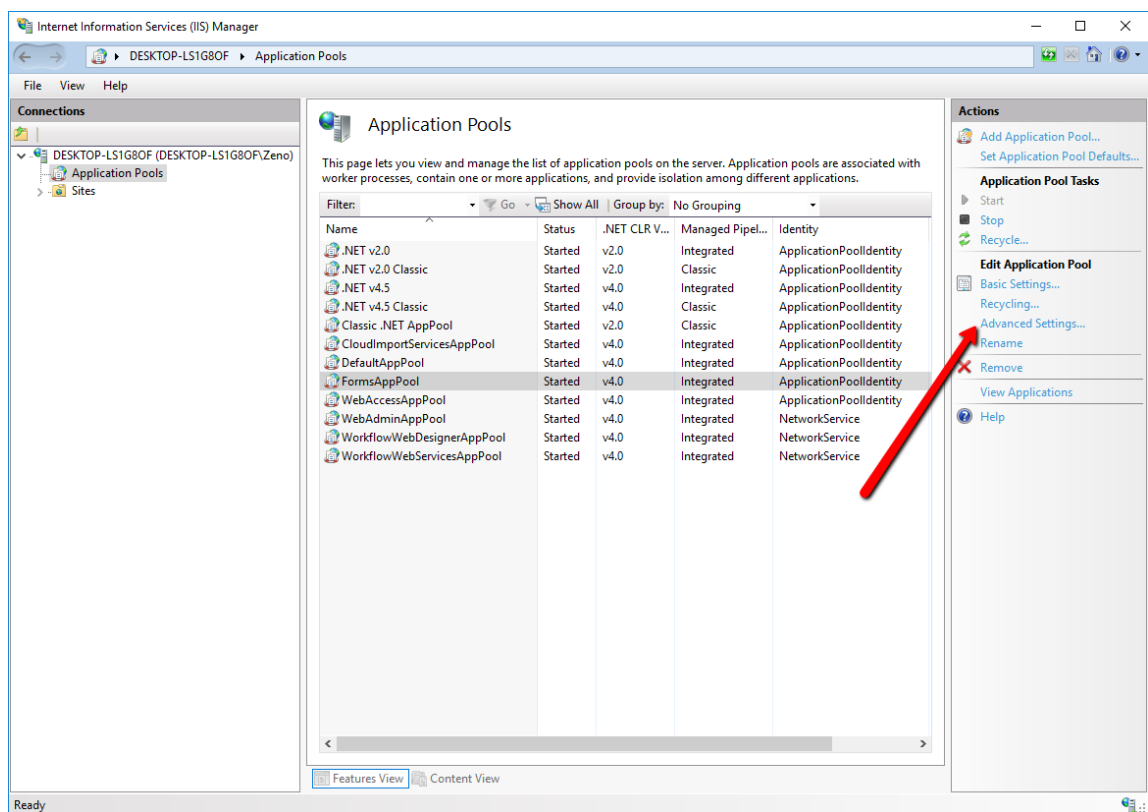


Figure 2.9: Open the FormsAppPool Advanced Settings.

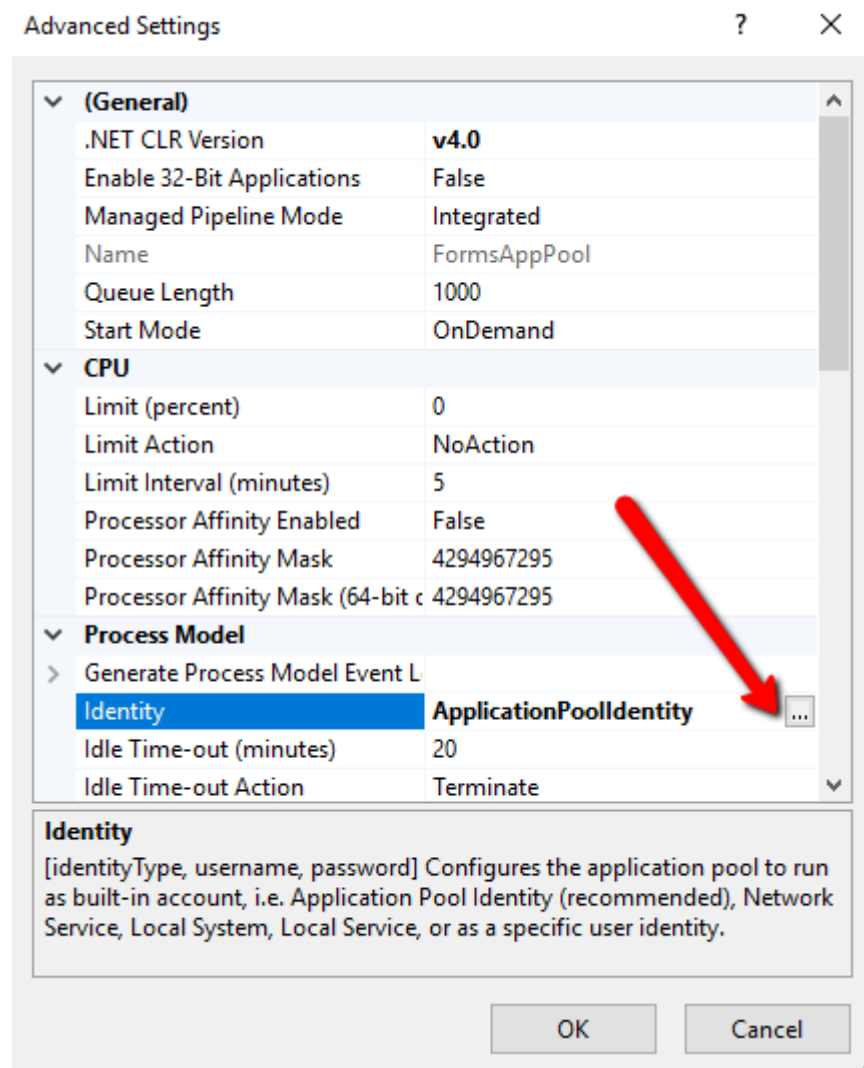


Figure 2.10: Edit the application pool identity.

In the Application Pool Identity dialog, we will change the Built-in account from “ApplicationPoolIdentity” (which is the default for all application pools) to “Local System”.

Now let’s go back to the Forms Configuration page and reload it, then re-enter the database information. This time, when we click Save, the database should get created. Sweet!

Forms Server

In this section, we can specify the Forms server's various settings. The most important setting is the Primary Forms Server URL. We won't actually use this in this tutorial, but it's worth mentioning. When multiple Forms servers are connected to the same database (such as for load-balancing), the Forms server specified here performs the routing of submitted forms, while the Routing Service associated with the other Forms server takes a back seat. If that is confusing, don't worry about it too much. Just know that, if you have multiple Laserfiche Forms installations sharing a single Forms database, you will need to designate one of them as the primary. You can read more about creating Forms Server Clusters [here](#).

We can also specify the time zone, which is used when calculating task deadlines and priorities. In most cases you can leave it as is, but if your Forms server is located in a different timezone than your organization (for example, it's cloud-based) then you can change it here.

Lastly, we can edit the default error messages that are seen by public users. This can be useful if we want to provide a contact email or alternative instructions, such as to use another website, or a phone number. Personally, I've never had to edit these error messages, but it's still good to know they can be edited.

User Authentication

Next, let's take a look at the User Authentication section of the Forms Configuration page. We have two options:

1. **Laserfiche Server:** The list of users who can log into Laserfiche Forms is pulled from a Laserfiche Server's Named Users list. This most often used in Laserfiche Avante systems.
2. **Laserfiche Directory Server:** The list of users who can log into Laserfiche Forms is pulled from Laserfiche Directory Service. This offers Single Sign-On capability (meaning that logging into one Laserfiche module automatically logs the user into all of them), but is only available in Laserfiche Rio.

Since we want our training environment to be self-contained, we are going to use Laserfiche Server authentication and specify the location of our Laserfiche Server, which is “localhost”. We will then pick the LFTraining repository and enter the credentials of the “Forms” user we created in the previous section.

For environments where users will be logging in using Windows authentication, the location of the Active Directory domain controller also needs to be specified, but in our case it’s not necessary.

Lastly, we have user synchronization, which is the mechanism via which Laserfiche Forms synchronizes its list of user accounts with the authentication service (which, in our case, is Laserfiche Server). We can leave the settings at their defaults, but user synchronization is an important concept. What it means is that any changes we make to the user accounts at their source (i.e. the Laserfiche Server) will not propagate to Laserfiche Forms right away. For example, if we add a password to our Admin user, it will take some time (up to 48 hours after the initial synchronization) before we can log into Laserfiche Forms with that new password. To get around this, synchronization can be performed manually, and we will look at that in the next chapter. But I wanted to explain this because the mechanism can be a bit confusing for newcomers.

Box 2.7. Access denied error when saving repository settings

A few people have reported getting an “Access denied” error when using the newly created Forms user to add the repository to Forms Configuration. If that happens, you can get around it by temporarily granting all privileges to the Forms user.

Email Settings

One of the most useful features in Laserfiche Forms is email notifications. These allow users to get notified when there are tasks they need to perform. In addition, a properly configured email server can let users perform approval

tasks straight from their email client, without even having to go to the Laserfiche Forms site.

In this section, we are going to download and install a free and open-source email server called hMailServer. If your organization already has an SMTP server, you can use that. But for self-contained training environments, using your own email server can provide additional flexibility.

Let's go ahead and [download hMailServer](#). As of this writing, the latest stable version is 5.6.6, and version 5.6.7 is in beta. By the time you read this, new versions may have been released. For best results though, I would recommend sticking with 5.6.6, which can be found on the [download archive](#).

The installation has just a few steps. Make sure to install both the server and the administrative tools, and use the built-in database engine (Microsoft SQL Compact), which we can do because our use will be non-commercial. Also, you will be asked to provide a 5-character password for the default hMailServer administrative user, so make sure to use something easy to remember (such as "hmail").

At the end of the installation, we will launch the hMailServer Administrator. Then we will add our hMailServer instance on localhost, and double-click it to edit it.

The first thing we need is a domain. In self-contained environments, this can be anything. We will name ours "lfttraining.com". Once we click Save, we will need to add an account. This can be done from the Accounts section under the domain we just created.

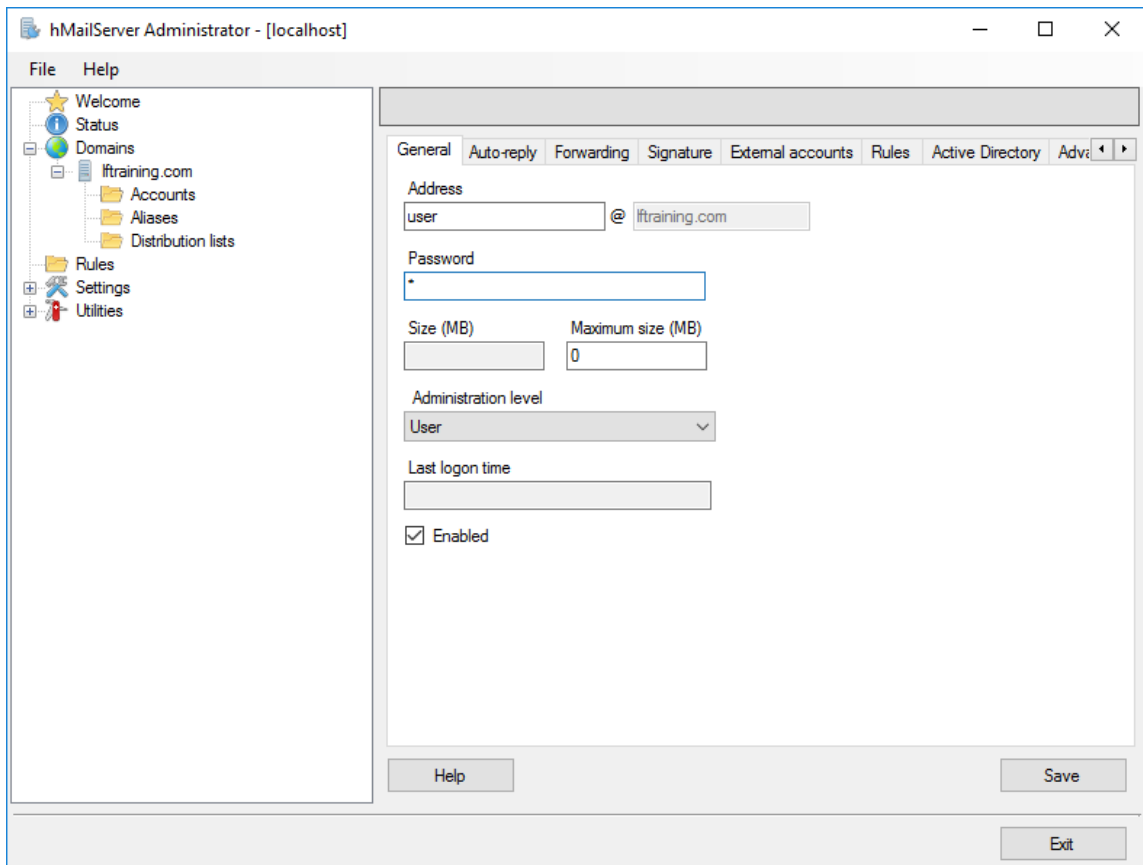


Figure 2.11: Creating the user@lftraining.com account.

Once everything is configured, we will go back to the Laserfiche Forms Configuration Page and reload the page. Then we will enter the Email Server settings:

```
SMTP Server: localhost
Port: 25
Username: user@lftraining.com
Password: <<The password you specified when creating the account>>
Sender's email account: no-reply@lftraining.com
```

Box 2.8. Sender's email account

You may have noticed that we entered “no-reply@lfttraining.com” as the sender's email account, even though we never configured that account in the hMailServer Administration tool. This is because the sender's email account is not actually used. It simply specifies the “From” field in outgoing emails. Sending emails from a no-reply address is common practice, since we don't want users to reply to emails sent by Laserfiche Forms, at least for now.

To validate our email settings, we will send an email to “user@lfttraining.com”. If everything has been properly configured, validation should succeed.

Note that you won't be able to send emails to “real” email addresses from this hMailServer. The reason is that the settings are essentially made-up. They will work inside our training environment, but any emails we try to send to outside email addresses will not be recognized as valid.

At this point, it would be a good idea to install an email client in our training environment as well. This will allow us to actually see the emails that are sent by Laserfiche Forms. One of my favorite email clients is Mozilla Thunderbird. Like hMailServer, it is free and open-source, and can be downloaded from [here](#). Once you install and launch it, you can add the “user@lfttraining.com” account to it and start receiving emails.

Box 2.9. Email client configuration

Whatever email client you decide to use, the configuration is going to be a bit tricky. I'm going to include step-by-step configuration instructions here for Mozilla Thunderbird. If you use another email client, the settings should be similar, although it is up to you to locate and configure them.

The first time we launch Mozilla Thunderbird, we will see a welcome message that asks if we want a new email address. We will click the “Skip this and use my existing email” button:

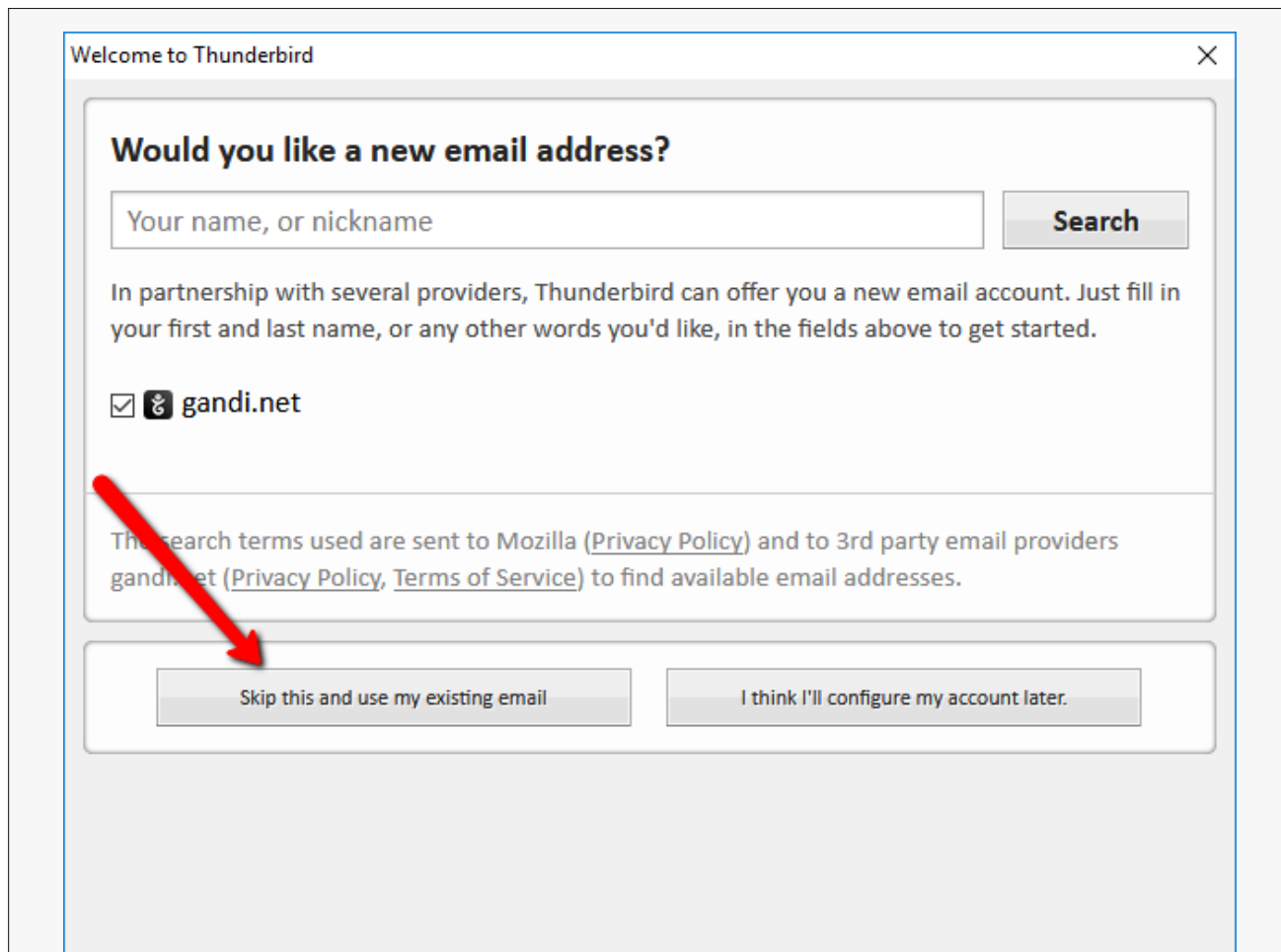


Figure 2.12: We want to use the account we just created.

Then, we will enter a name (it doesn't matter what it is) as well as "user@lfttraining.com" for the email, and the password we used when creating the account.

When we click Continue, Mozilla will look for the configuration in its database, and actually find it. **However, the configuration Mozilla finds is not valid.** We need to override it by clicking the "Manual config" button.

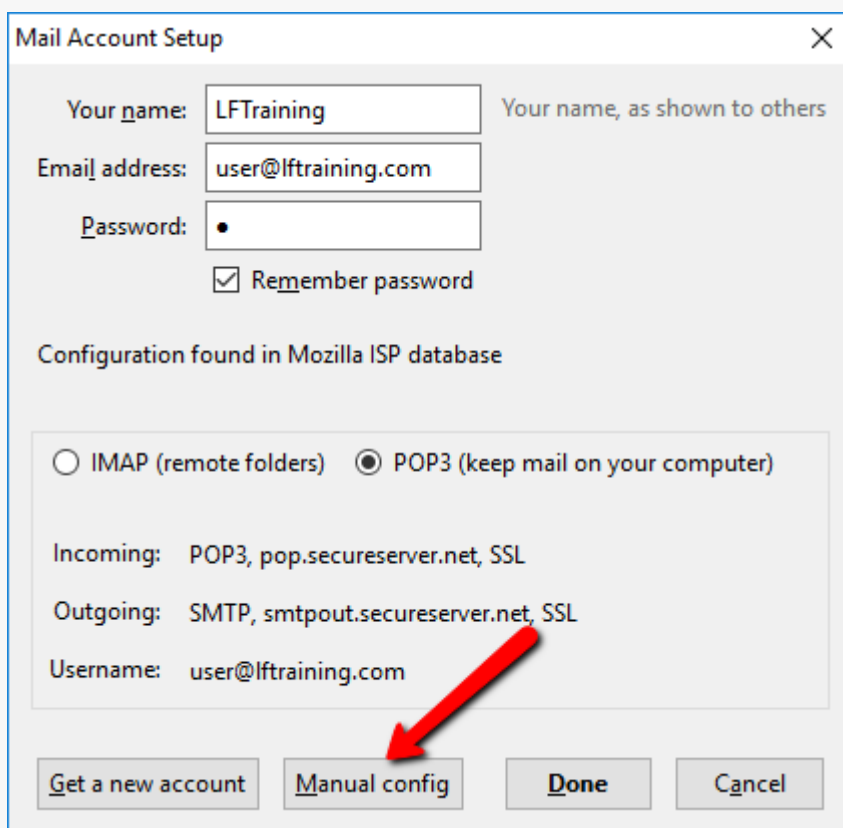


Figure 2.13: Manual override of email account settings.

In the manual configuration section, specify “localhost” as the server host-name for both Incoming and Outgoing servers (the former should be POP3 and the latter should be SMTP), then port 110 and 25, respectively. A screenshot with these settings is shown below:

Mail Account Setup

Your name: LFTraining Your name, as shown to others

Email address: user@lftraining.com

Password: •

Remember password

The following settings were found by probing the given server

| | Server hostname | Port | SSL | Authentication |
|----------------|-----------------|------|------|-----------------|
| Incoming: POP3 | localhost | 110 | None | Normal password |
| Outgoing: SMTP | localhost | 25 | None | Normal password |

Username: Incoming: user@lftraining.com Outgoing: user@lftraining.com

Get a new account Advanced config Re-test Done Cancel

Figure 2.14: Hostnames and ports for incoming and outgoing emails.

Click re-test to make sure everything is valid, then click OK. You will get a warning saying localhost does not use encryption, which is fine in this case since this is a training environment. Check the “I understand the risks” checkbox and click Done. Then you should be able to go to your Inbox, click “Get messages” and download any emails that may be on the server (such as the verification message from the Laserfiche Forms Configuration page)

Wow, that was quite a bit of work! The good news is that we are now done with the Email Settings configuration, and are ready to move onto the next section.

Email Templates

This section contains templates for various types of emails that Laserfiche Forms sends. We don't really care about these right now, but the Aggregated Notification email deserves special mention because it needs to be explicitly enabled.

Aggregated Notifications allow Laserfiche Forms to send daily emails to users with a list of Tasks that are assigned or available to them. This type of email digest can be particularly useful for users who get assigned a lot of Tasks in Laserfiche Forms. Instead of sending a notification for each and every single Task, which can get overwhelming, you can opt to send one email with a list. Or you can do both. It is really up to you.

Box 2.10. Laserfiche Forms Tasks

You may have noticed I capitalized the word "Task". This is because a Task refers to a step in a Laserfiche Forms Business Process that needs to be worked on by a user. We will take a look at this feature soon.

For now, let's go ahead and enable aggregated notifications, and check Saturday and Sunday if they are not already checked.

Notification Service

Laserfiche Forms Notification Service allows Laserfiche Forms to make real-time updates to a user's task list. Essentially, it allows users to see the most up-to-date view of their Task Inbox without having to refresh the page. I really like this feature because it makes Laserfiche Forms even more user-friendly. Let's go ahead and enable it by typing `//localhost:8181` as the Laserfiche Notification Hub URL. 8181 is the default port for this service, so we can use it safely.

Laserfiche

In this section we can specify the location of Laserfiche Workflow Server and Laserfiche Discussions. The former allows Laserfiche Forms to initiate workflows, whereas the latter enables adding FAQs to forms. This tutorial covers a stand-alone Laserfiche Forms system, so we are going to skip both of these steps, and move on to the next section.

Payment Gateway

One of the most exciting features in Laserfiche Forms 10.2 is the Payment Gateway feature. This allows your users to make payments to your Braintree merchant account directly from the forms you create in Laserfiche Forms. We are going to use this feature in Sandbox Mode in this tutorial (meaning no real money will be transferred). So let's head over to Braintree and [sign up for the sandbox](#).

Box 2.11. Braintree Signup

When signing up, we need to use our real email address, instead of the one we created on our fake lftraining.com domain.

After filling the signup form, we will receive an email, which will have a link to a user information form. We will use this to create a password. Once we sign up, we will be taken to the Braintree Dashboard, which contains the three pieces of information we need to integrate Braintree with Laserfiche Forms: Merchant ID, Public Key and Private Key. Go ahead and enter these into the Laserfiche Forms Configuration page. Remember to check the "Sandbox mode" checkbox at the top.

Diagnostics

The last step is Diagnostics configuration. We will actually leave this the way it is, but it is worth mentioning. The Log Level setting becomes very helpful during troubleshooting scenarios. For example, if we are getting an error message in Laserfiche Forms, we can come to this page and change the Log Level from **Warning** to **Debug**, which will make Laserfiche Forms generate more detailed logs that can help us get to the bottom of the issue.

We are now done with Laserfiche Forms Configuration! We are ready to create a very basic Business Process to make sure everything is working.

2.1.5 Testing Our Configuration

Let's now go to "localhost/forms" and click on the Design link on the navigation bar. We will see a dialog box that will ask us to choose an existing process or create a new process. We don't have any existing processes yet, so we will create a new one. Then, we will specify that we want to use the "Form Submission" template.

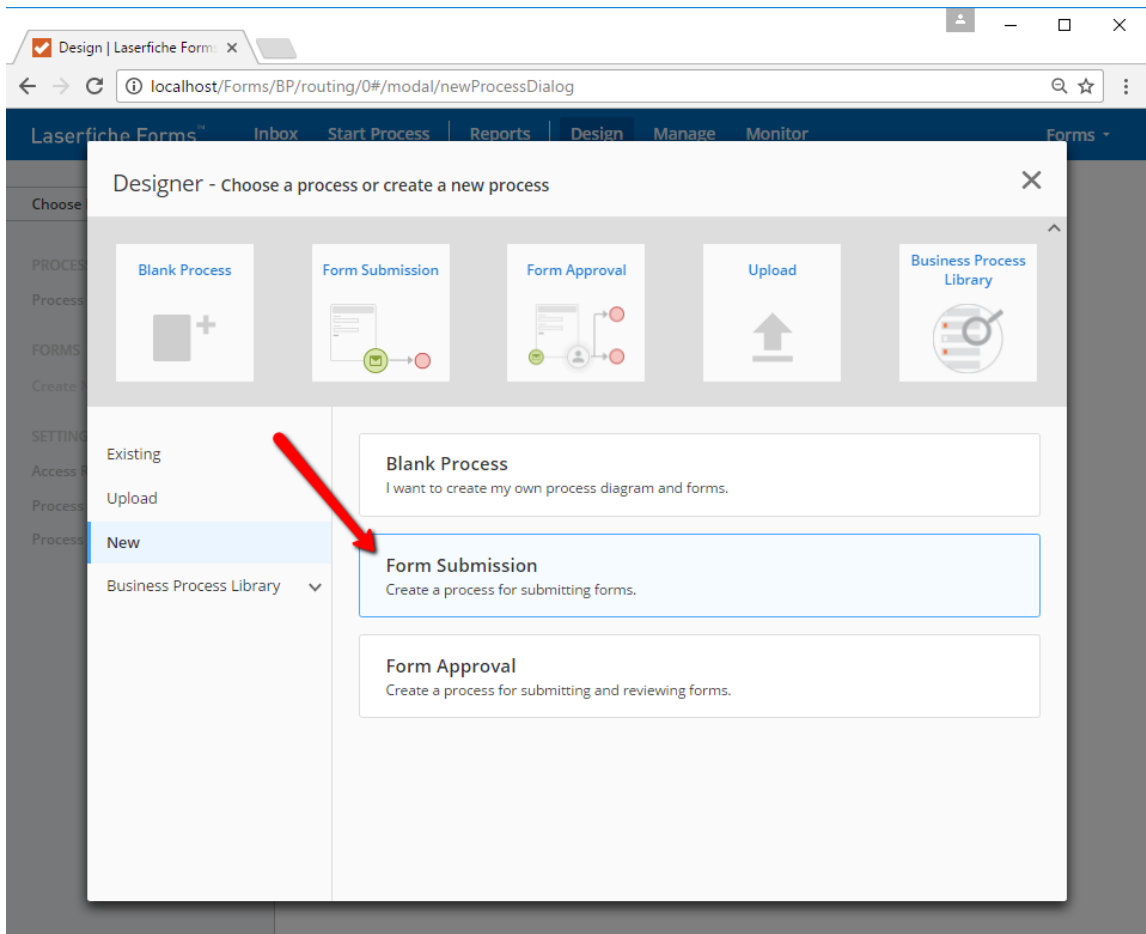


Figure 2.15: Creating a new Form Submission process.

Let's name this process "My First Business Process". It's a bit corny, but don't worry, we won't be publishing it to anyone in the organization. :)

Since the goal of this Business Process is to make sure our test environment configuration is valid, we will test several things:

1. Sending emails
2. Storing submitted forms in the LFTraining repository
3. Making (fake) payments via our Braintree Sandbox account

When you create the Business Process, it will take you to the Process Diagram. You should see a Start activity, an End activity, and a bunch of links and settings on the left-hand side. Let's click on Starting Form and add a couple of fields to it.

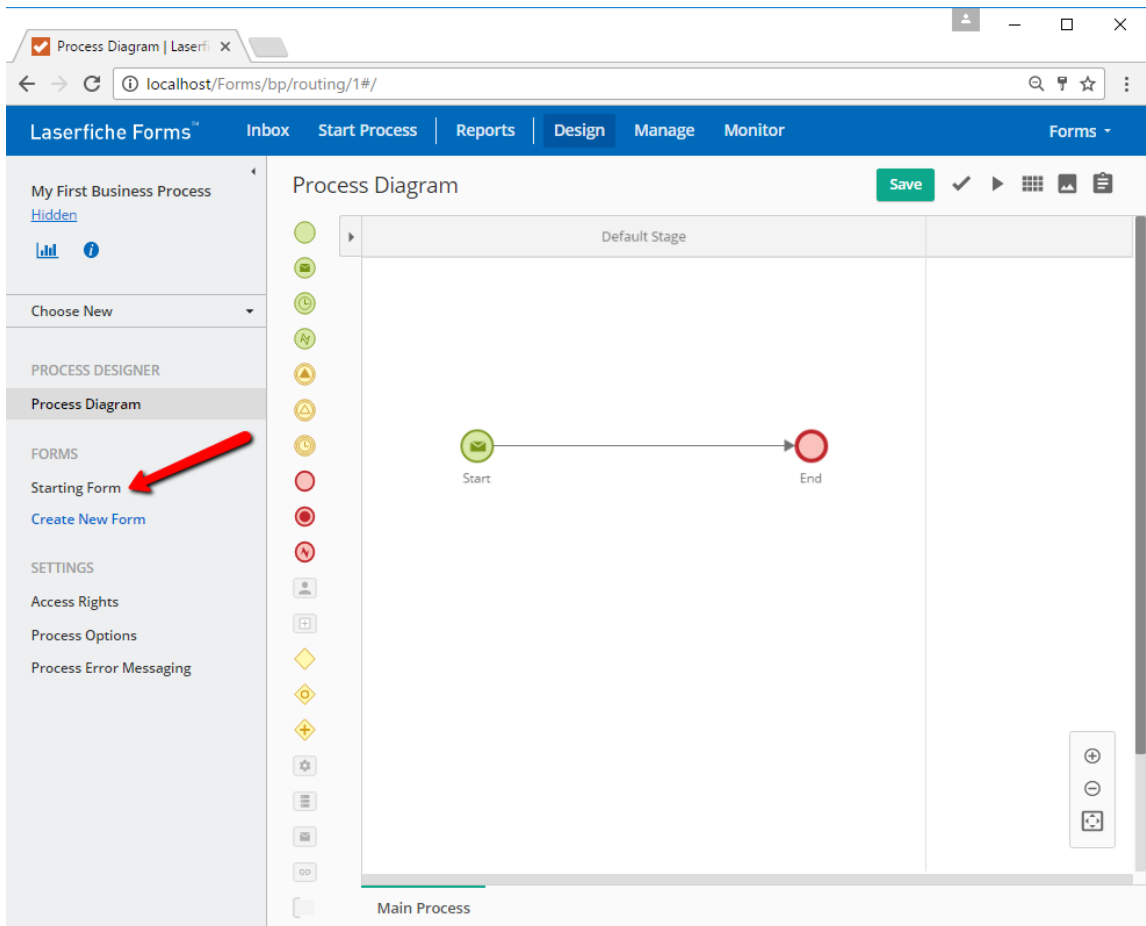


Figure 2.16: Navigating to the Starting Form from the Process Diagram.

In the Form Designer, let's add two Single Line fields by dragging and dropping them into the canvas. We will name them "First Name" and "Last Name". There are a bunch of other options that can be configured for each field, and we will visit them later. For now, naming the fields is sufficient.

We will also edit the Form Title by clicking on it and selecting Edit. We

will call our form “My First Form”.

Once done, we will go back to the Process Diagram by clicking its link on the left pane.

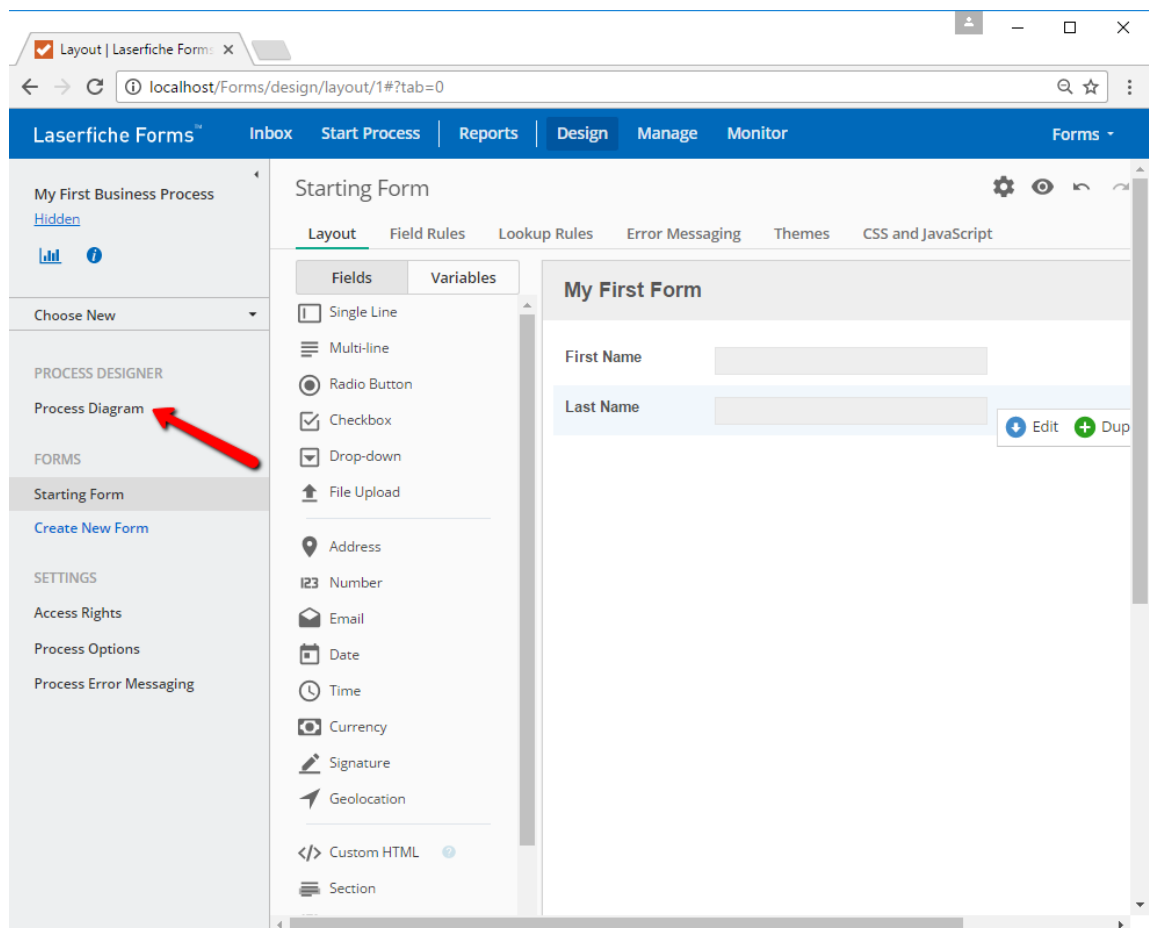


Figure 2.17: Navigating back to the Process Diagram.

In the Process Diagram, we are going to add two Service Tasks to the process. The first one is going to be a Save to Repository Service Task and the second one is going to be an Email Service Task. These are shown below:

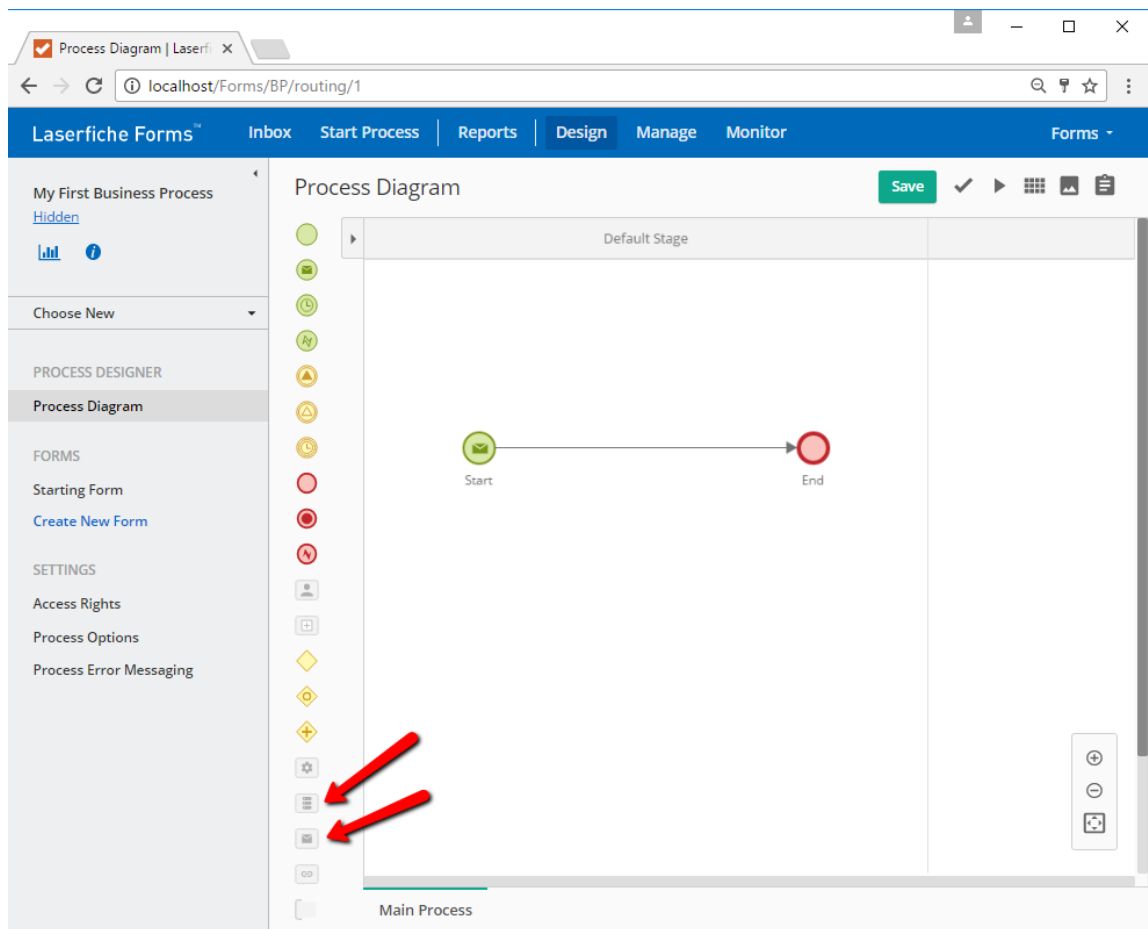


Figure 2.18: Save to Repository and Email Service Tasks.

Drag and drop these into the process, and connect them so that the process flows from the Message Start Event to the End Event, going through the two service tasks along the way. You can distinguish them by their icons.

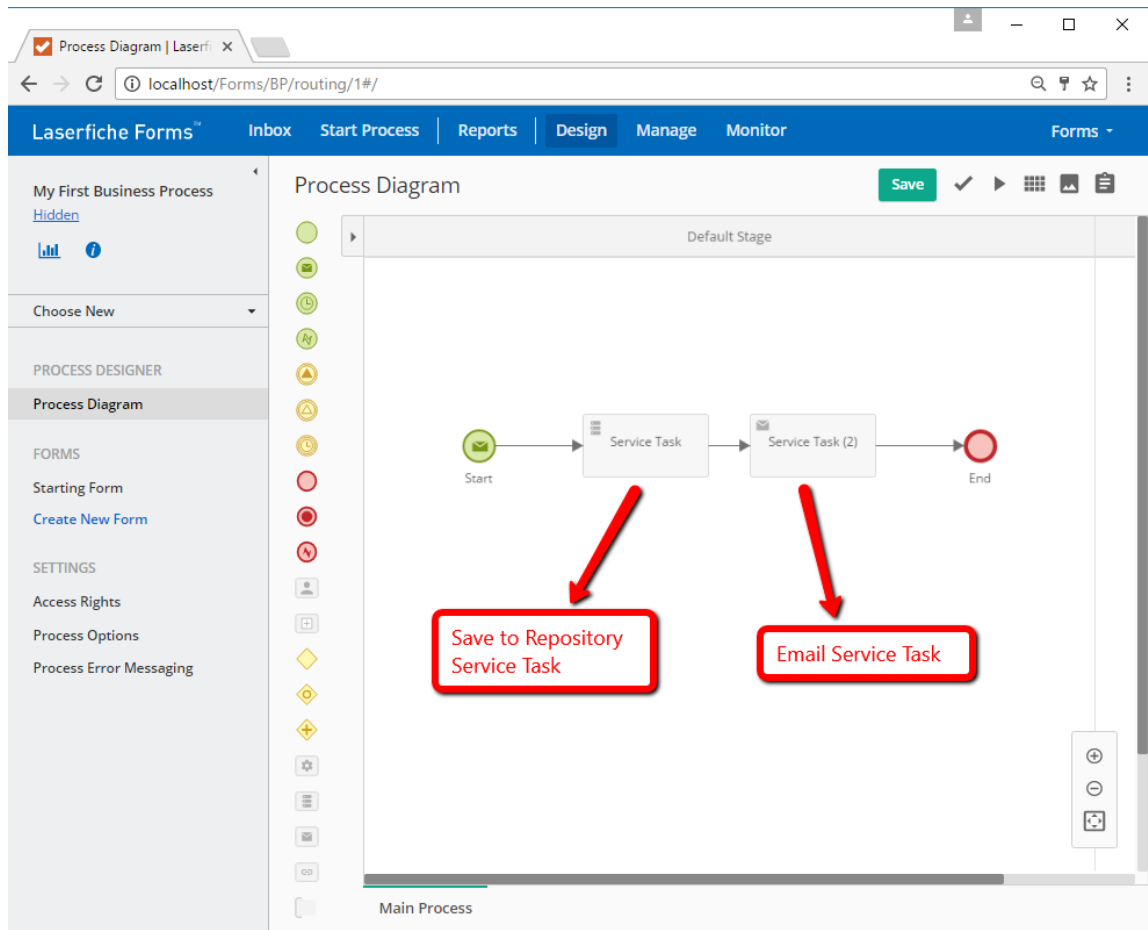


Figure 2.19: Connecting the service tasks to the Message Start and End Events.

Now let's configure each one.

Save to Repository Service Task

As its name implies, the Save to Repository Service Task allows us to save images of submitted forms to the Laserfiche Repository. In this section, we will configure where in the repository the forms should get saved.

It's a good habit to provide a descriptive name to items in Process Diagrams, so we will start by naming this service task "Save submitted form to LFTrain". Then, we will create a new repository profile.

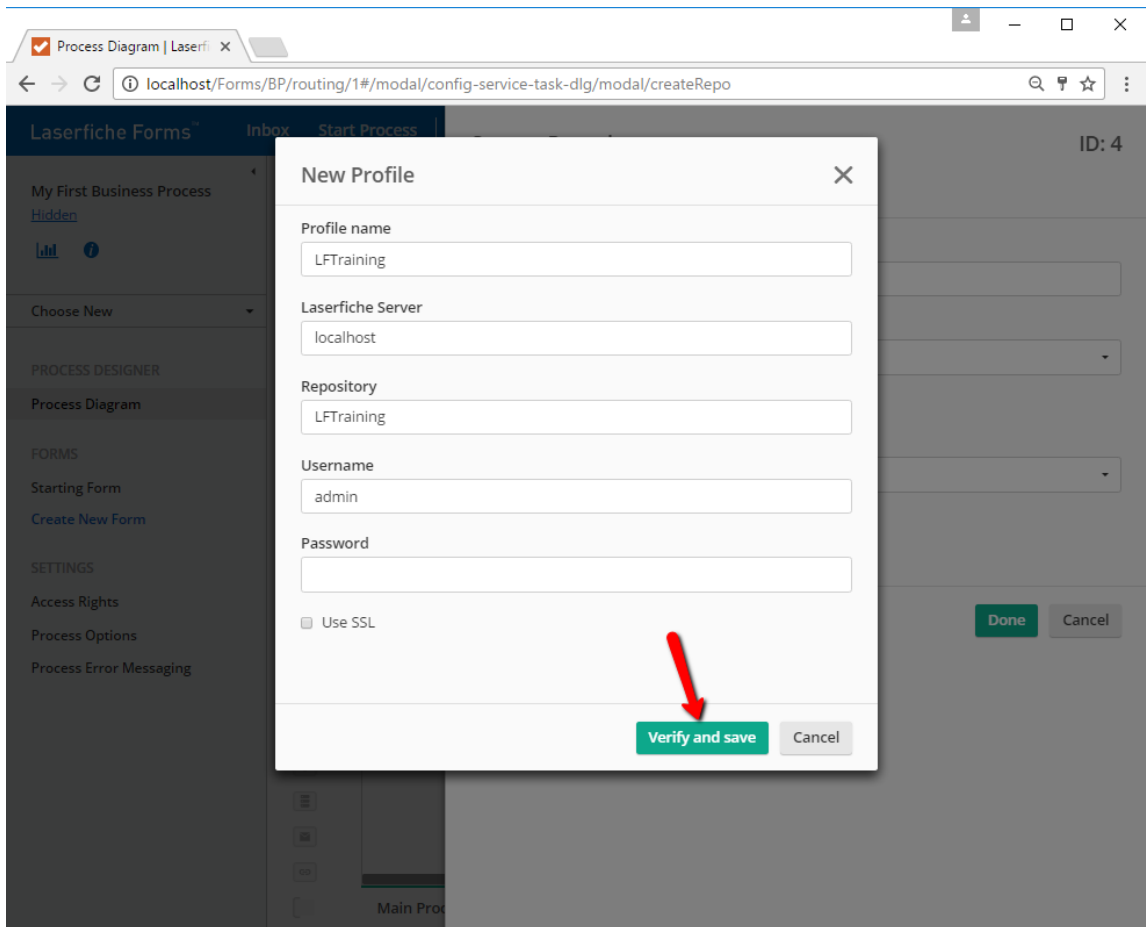


Figure 2.20: Creating a new profile for Save to Repository Service Task.

Once we verify and save the new profile, we can specify what we want to save and where.

We are given two options. The first option allows us to save the submitted form from a specific process step. The second option allows us to save a form (usually a different form than the one submitted) with current process data. We will cover each of these options later. For now, we will pick the first option and select “Start (Starting Form)”.

We can also specify the document name for the stored form, and change its path. We are going to leave these the way they are. However, we will change the file format from PDF to TIF. TIF is a better default format, because it allows

repository users to use Laserfiche annotations on the document. It can still be exported or emailed as a PDF if needed.

Lastly, we can configure Fields (metadata) for the stored form or its folder. We will use this setting later. For now, let's just leave those the way they are and move on to configuring the next Service Task.

Email Service Task

The Email Service Task allows sending discrete email notifications during a Forms Business Process. This task is not the only way to send email notifications, but in this case we will use it to quickly test our Forms configuration.

Let's name it "Notify submitter". You will notice the From field has been automatically filled out for us with the **no-reply** email address. For the Send To field, we will use "user@lftraining.com", and enter "Test Successful!" as the subject line. We will leave the rest of the options blank and come back to them in a later chapter.

Enabling Payments

The last thing we will do before testing the Business Process is enabling payments on the form we created. To do this, we will double-click the Message Start Event, and check the Collect Payment option. We can leave the amount at \$10 - it will not be collected anyway, since we are in Sandbox mode.

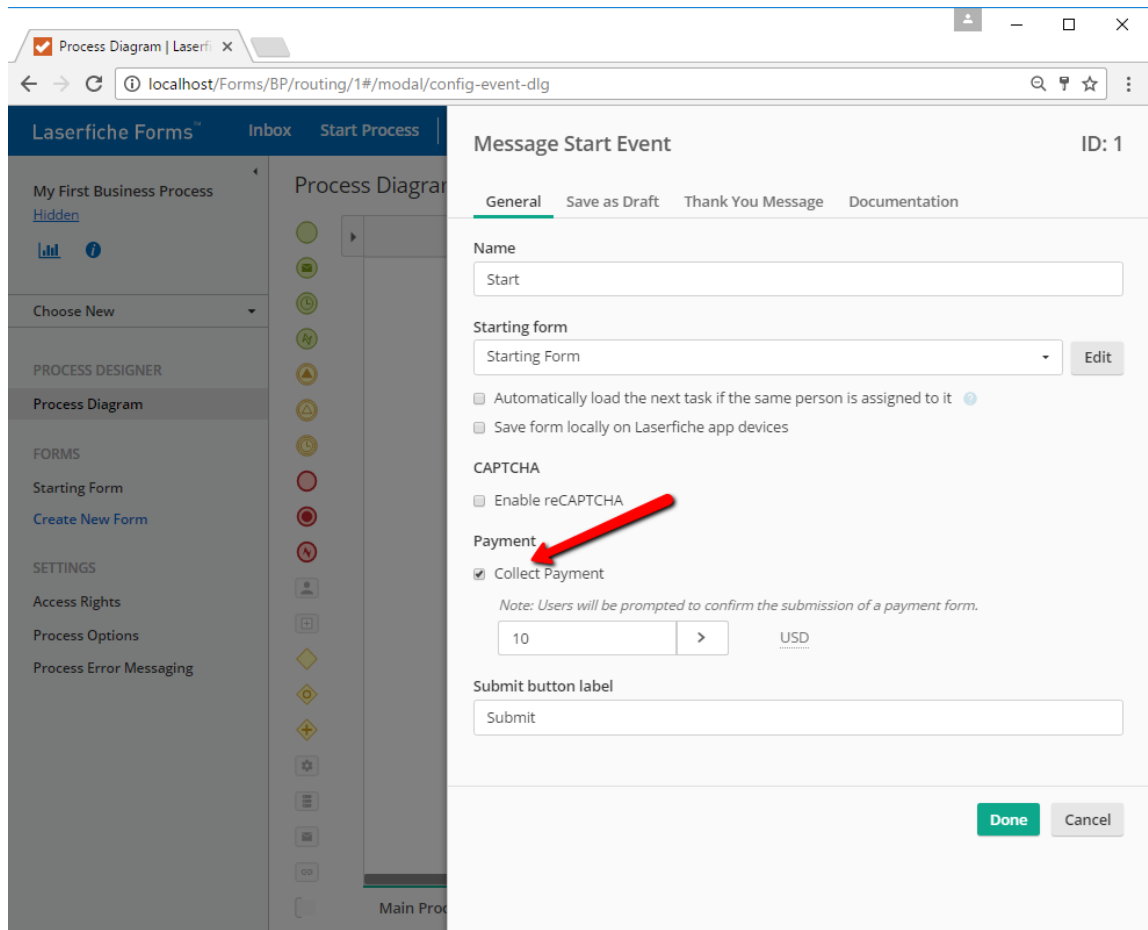


Figure 2.21: Enabling payments in the Message Start Event.

Testing

We are ready to submit a test form now. We first need to save the changes we made to the Process Diagram. Then we will click the Save button, and then the Start Process button, as shown below:

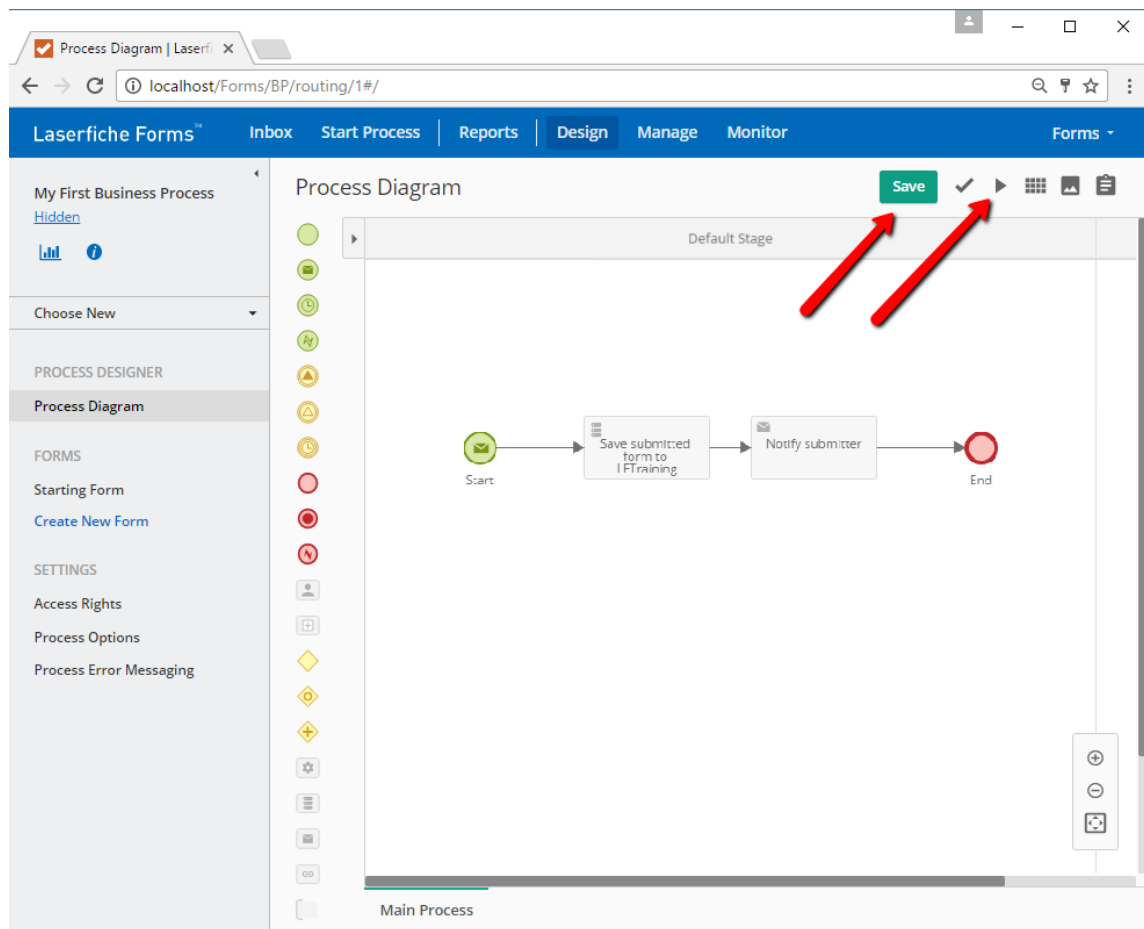
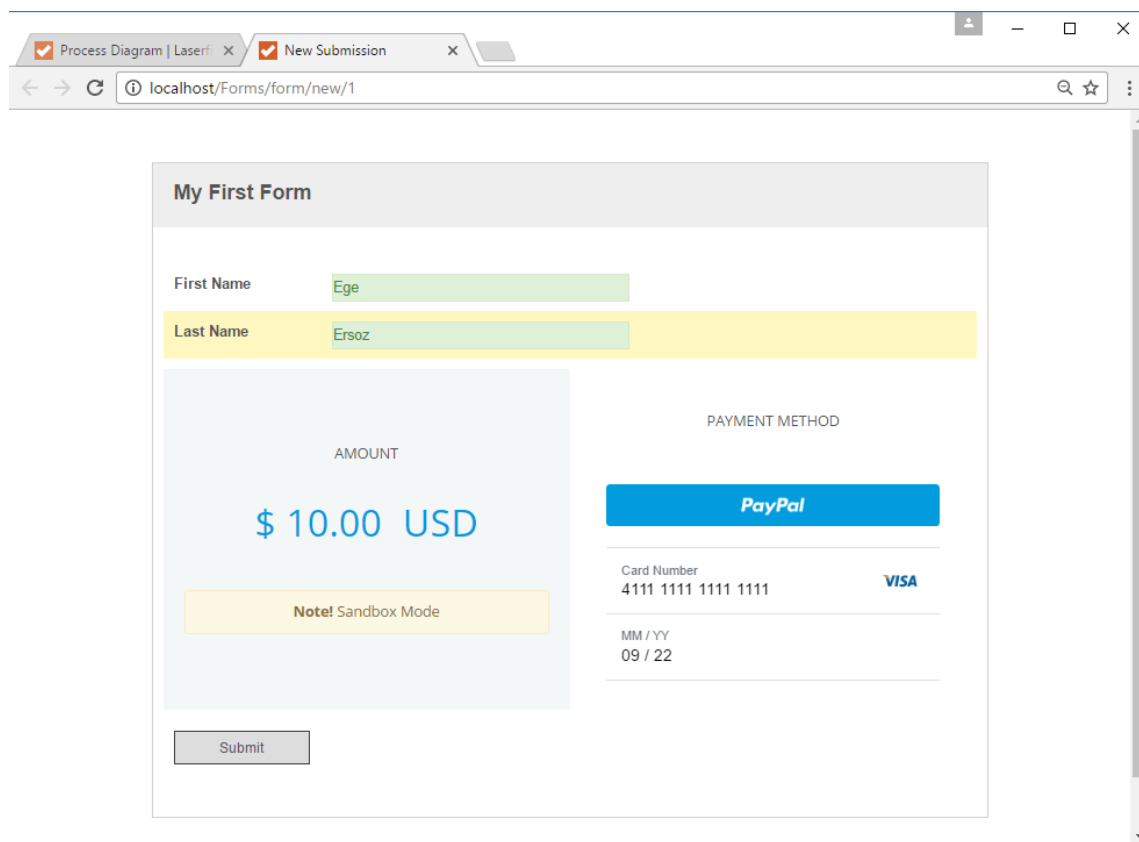


Figure 2.22: Saving a Process Diagram and starting the process.

Our Starting Form should launch, and in addition to the First Name and Last Name fields, it should also have payment fields for a Credit Card and Expiration date. Let's fill these real quick:



The screenshot shows a web browser window with two tabs: 'Process Diagram | Laserf...' and 'New Submission'. The address bar shows 'localhost/Forms/form/new/1'. The main content area displays a form titled 'My First Form'. The form has two input fields: 'First Name' with the value 'Ege' and 'Last Name' with the value 'Ersoz'. Below these fields, there is a section for 'AMOUNT' showing '\$ 10.00 USD' and a note 'Note! Sandbox Mode'. To the right, there is a 'PAYMENT METHOD' section with a 'PayPal' button, a 'Card Number' field with the value '4111 1111 1111 1111' and a 'VISA' logo, and an 'MM / YY' field with the value '09 / 22'. A 'Submit' button is located at the bottom left of the form.

Figure 2.23: Submitting a test form.

Box 2.12. Test Credit Card Numbers

You may have noticed that I used **4111111111111111** as the credit card number and **09/22** for the expiration date. These are example test values that can be used in Braintree's Sandbox mode. You can find more of them [here](#).

Once we submit, Braintree will first verify the credit card info we entered, then present us with a Confirmation dialog. Clicking Confirm will submit the form.

Great! Let's now check to see if everything worked.

First, we should log into the LFTraining repository (<localhost/laserfiche>) and make sure that an image of our submitted form got stored under the root folder.

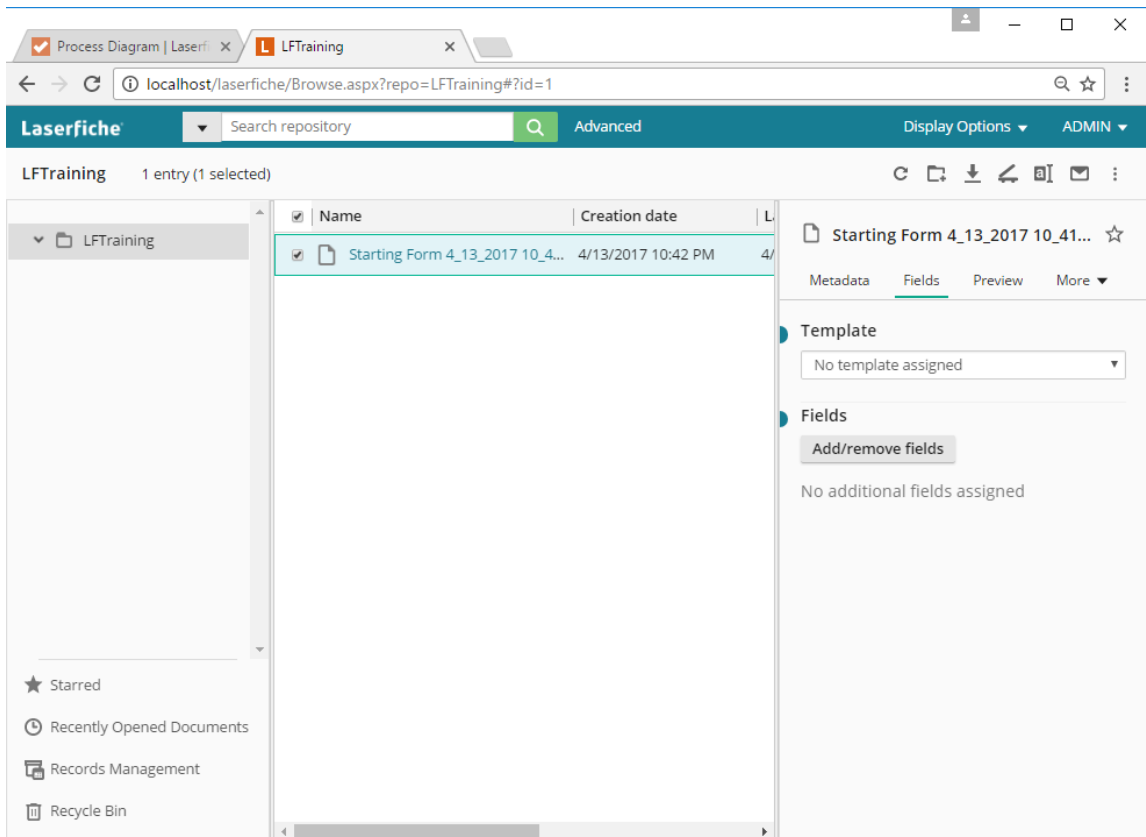


Figure 2.24: Submitting a test form.

Second, we should check our email and verify that we received a new email with the subject line “Test Successful!”.

Lastly, we should go to our Braintree Dashboard and check if a new transaction for \$10 has been recorded.

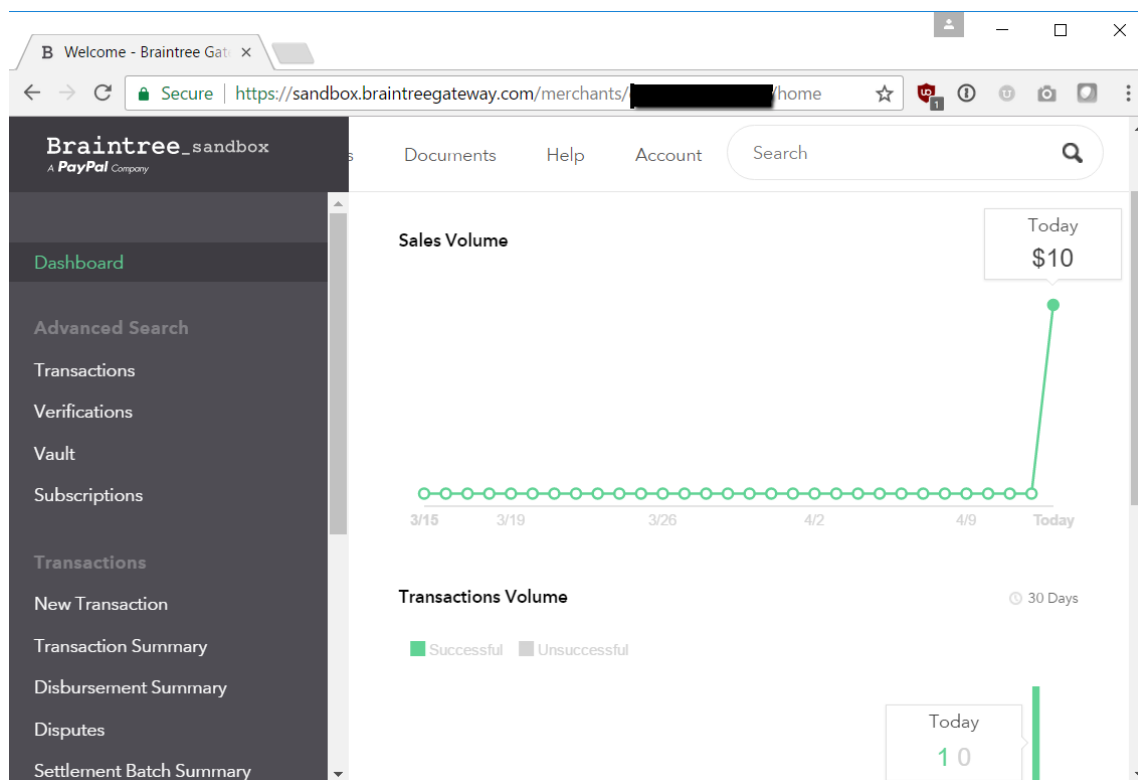


Figure 2.25: We should see a new (fake) transaction for \$10.

2.2 Chapter Summary

In this chapter, we got up and running by installing and configuring the necessary Laserfiche components and ancillary tools we will use in the rest of the tutorial. If you got to this stage without issues, great. If you ran into error messages or ran into unexpected results, revisit the relevant sections and make sure you followed the instructions correctly. You will need a fully working training environment in order to follow along with the rest of the tutorial.

Chapter 3

Business Process: Customer Satisfaction Surveys

In this chapter, we are going to build a Business Process in Laserfiche Forms that will allow users to submit customer satisfaction survey forms.

The purpose of this chapter is to provide an introduction to Laserfiche Forms and several of what I consider to be its core features.

Let's head to the **Design** link in Laserfiche Forms and create a new Business Process using the Form Submission template. We will name it "Business Autom While it is kind of a long name, we want to namespace everything we create in this book so that it can act as reference material later down the line.

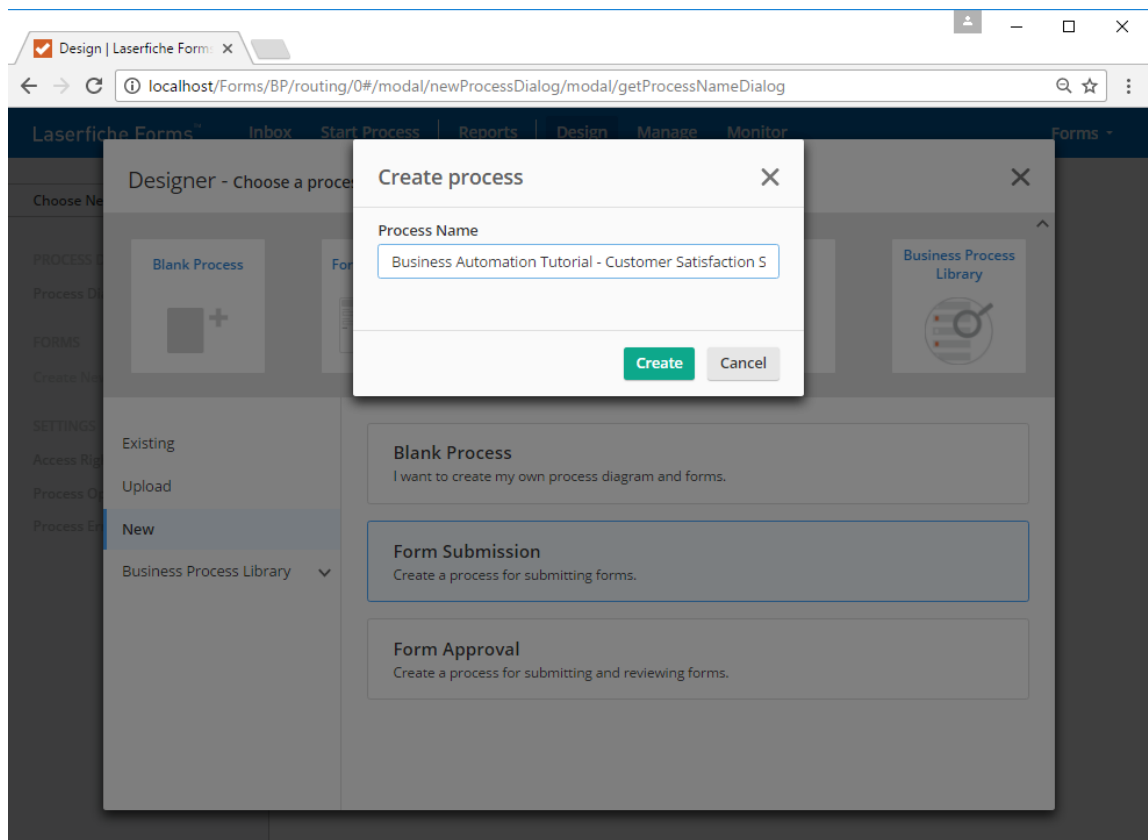


Figure 3.1: Creating the Customer Satisfaction Surveys business process.

We start in the Process Diagram. First, let's open our Starting Form (left side) and add some fields.

Box 3.1. Maximizing screen real-estate

The left-most pane, which provides links to various sections of the Business Process, can actually be minimized using the "Collapse" button shown below. This can be useful when we will be working inside a specific section for an extended period of time.

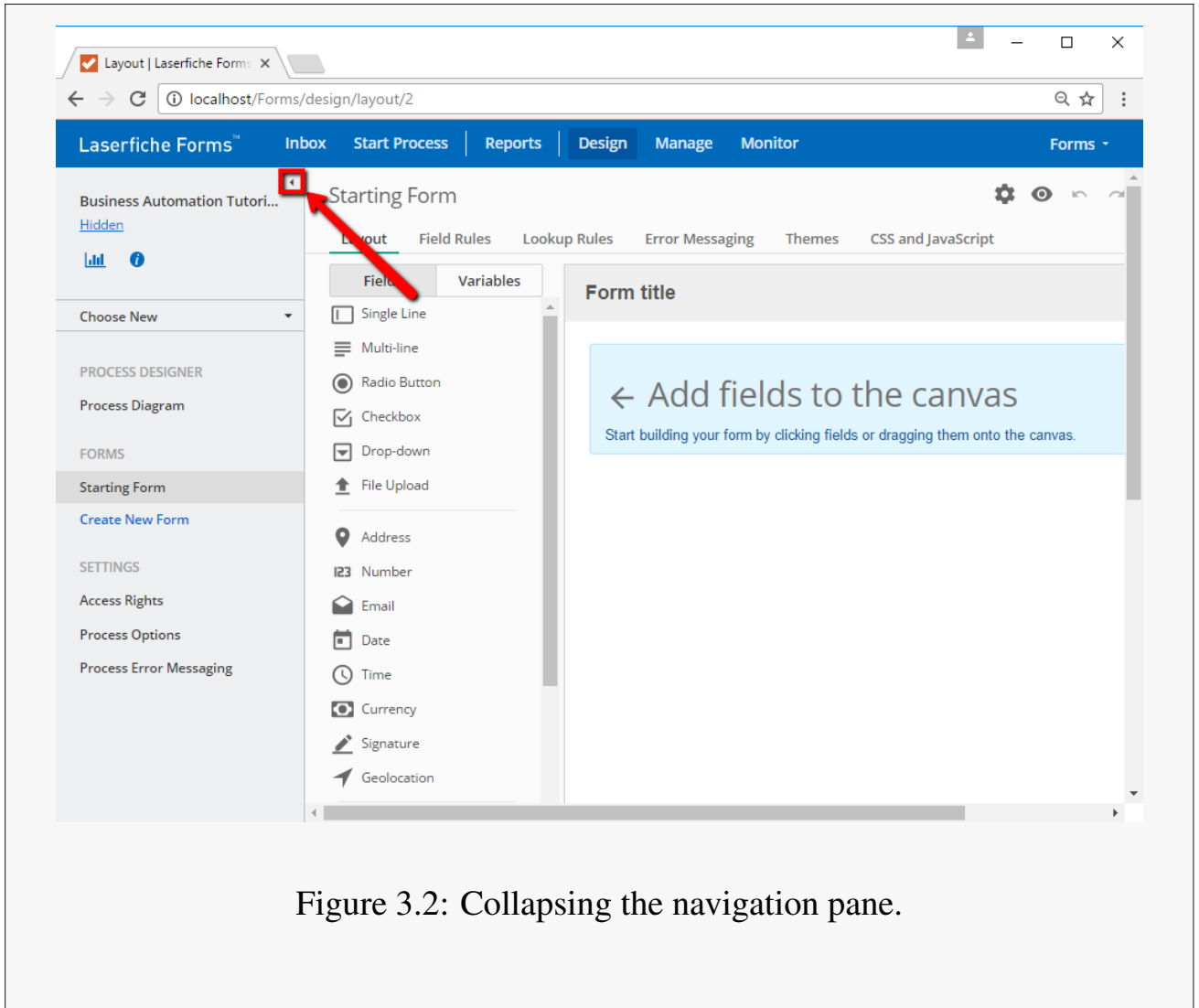


Figure 3.2: Collapsing the navigation pane.

3.1 Creating a customer satisfaction survey form

We will start by adding **First Name** and **Last Name** fields to our form. Similar to the form we created in [Chapter 2](#), these will be Single Line fields.

One of the many options Laserfiche Forms provides is marking fields as **Required**. This option has its pros and cons. On the positive side, it can enforce data consistency by ensuring that users provide a value. This can be

important if the information provided will be used by others in later stages. On the negative side, **Required** fields slow down (and possibly annoy) users, especially if it's not clear why they are required. Especially for public-facing forms, such as a customer survey form, it may be a good idea to mark as few fields **Required** as possible, so as to minimize the impact on response rates. We don't particularly care about the first name and last name fields in a customer survey, so we will leave them as optional.

Next, let's have a **Customer Service Representative** field that contains a list of salespeople in our organization. This will be a Dropdown field. It is a good idea to provide some descriptive text for this one. For the **Text above field** option, we will enter the following:

Please let us know which customer service representative assisted you.

Then for the choices we will enter three names:

- Bobby Drake
- Ashley Brown
- Jonathan Smith

Next, we are going to ask the user to rate the customer service representative. We will use three criteria: Responsiveness, Helpfulness and Overall Satisfaction. When it comes to single-selection multiple choice fields, we have two options: we can use a Radio Button field or a Dropdown field. There are pros and cons to each, but generally speaking, Radio Button fields are more appropriate for survey-type questions because answering them requires only one mouse click as opposed to two. This may sound like a minor thing, and it is, but when it comes to optional forms like a customer survey form, every minor thing counts.

Let's go ahead and add a Radio Button field and label it "Responsiveness". For the **Text above field** we will type:

How would you rate the responsiveness of the customer service representative?

And for the choices, we will enter the following:

- 5 - Very high
- 4 - High
- 3 - Average
- 2 - Low
- 1 - Very low

This is a question we really care about, so we will also mark the field as **Required**.

We need two more Radio Button fields, one for helpfulness and another for overall satisfaction. The **Helpfulness** field will be nearly identical to the **Responsiveness** field we have already added, and the only thing that will be different is its label and the text above the field. So instead of adding a brand new Radio Field from the Fields menu, we will select the **Responsiveness** field we created and then click **Duplicate**. This will create a copy of the field. We will then edit its title and help text.

52CHAPTER 3. BUSINESS PROCESS: CUSTOMER SATISFACTION SURVEYS

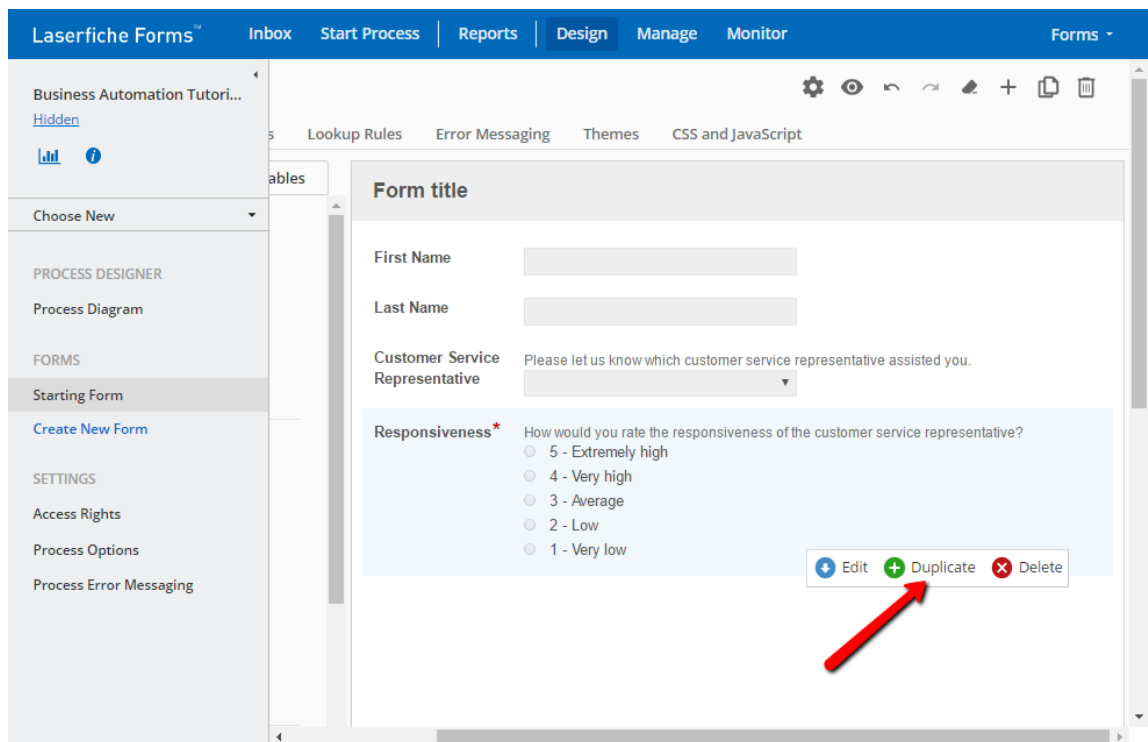


Figure 3.3: Duplicating an existing field.

Lastly, we will add the **Overall Satisfaction** field. This one will have slightly different help text and options.

How satisfied are you with our customer service?

And the options will be:

- 5 - Very satisfied
- 4 - Satisfied
- 3 - Average
- 2 - Unsatisfied

- 1 - Very unsatisfied

Box 3.2. Survey best practices

When designing surveys, you should pay attention to two things. First, there should be an odd number of choices. This allows users to pick the middle choice (“Average”) if they don’t have an opinion, and prevents such users (which can be a large percentage) from skewing your survey results in either direction. Second, your survey answers on the positive side should mirror those on the negative side. You’ll notice the highest option for the **Overall Satisfaction** field is “Very satisfied” whereas the lowest option is “Very unsatisfied”. This symmetry also helps you get accurate results from your survey.

The last thing we will do before giving things a whirl is edit the Form Title. We will title our form “Acme Customer Satisfaction Survey”. We will also enter the following description underneath:

Thank you for doing business with Acme! Please complete this short survey to let us know how we did.

We have been designing our form in the **Layout** section, which gives us a rough idea of the form’s look-and-feel, but we don’t really know what the form will *really* look like to a user once it is live. To this end, Laserfiche Forms has preview feature that can be utilized by clicking the **Preview** button on the toolbar:

54 CHAPTER 3. BUSINESS PROCESS: CUSTOMER SATISFACTION SURVEYS

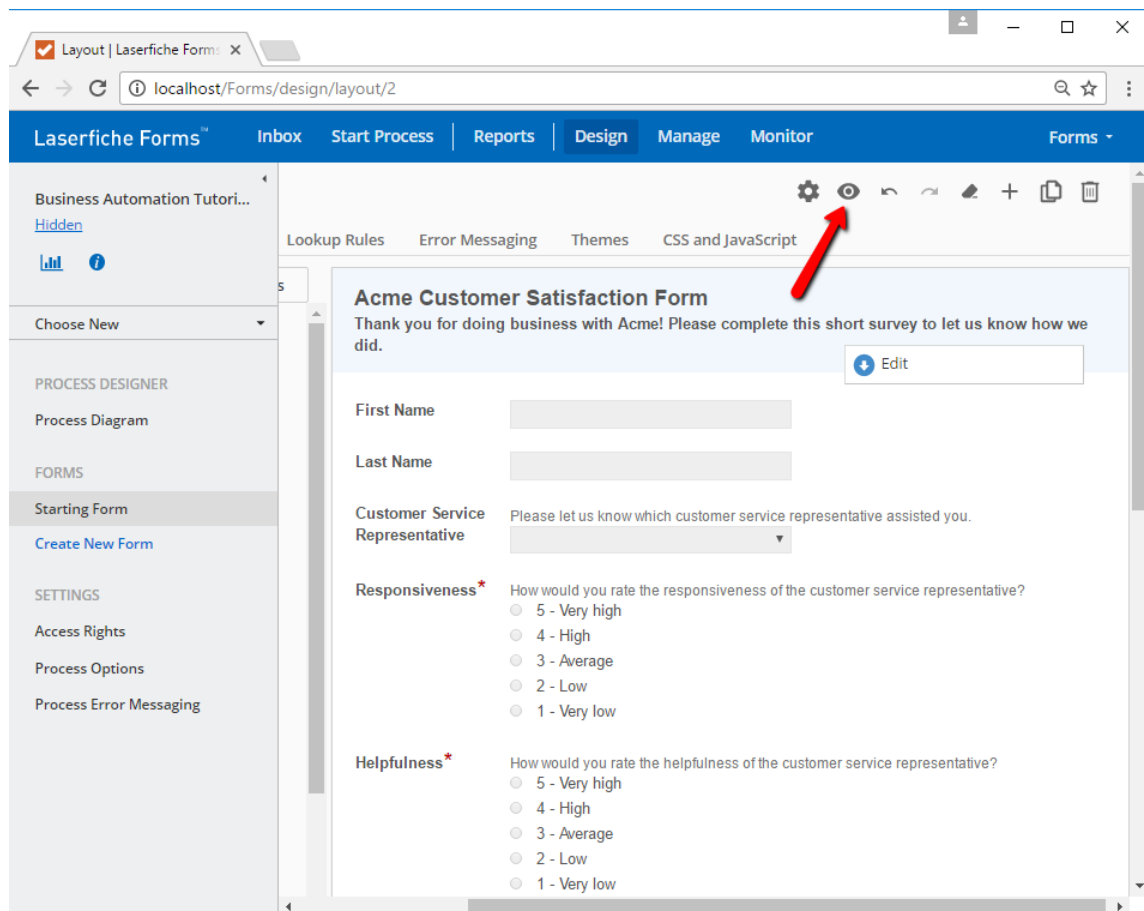


Figure 3.4: Previewing a form.

This will open a new tab in our browser and show us the user-facing view of the form that we created.

Preview Mode: Data entered will not be submitted.

Acme Customer Satisfaction Form

Thank you for doing business with Acme! Please complete this short survey to let us know how we did.

First Name

Last Name

Customer Service Representative Please let us know which customer service representative assisted you.

Responsiveness* How would you rate the responsiveness of the customer service representative?
 5 - Very high
 4 - High
 3 - Average
 2 - Low
 1 - Very low

Helpfulness* How would you rate the helpfulness of the customer service representative?
 5 - Very high
 4 - High
 3 - Average
 2 - Low
 1 - Very low

Overall Satisfaction* How satisfied are you with our customer service?
 5 - Very satisfied
 4 - Satisfied
 3 - Average
 2 - Unsatisfied
 1 - Very unsatisfied

Figure 3.5: Our customer survey form so far.

Note that, even though there is a **Submit** button, forms opened in Preview Mode don't get submitted. Therefore, clicking the button won't do anything.

The Preview Mode shows us the fields we have added to the form. You will notice that the three Radio Button fields have red asterisks next to their labels, which denote **Required** fields.

The form looks a little... basic. This is because we haven't added any styling to it.

3.1.1 Editing form theme

Laserfiche Forms allows customizing the “theme” of forms, which is what we will do now by going to the **Themes** tab of the form.

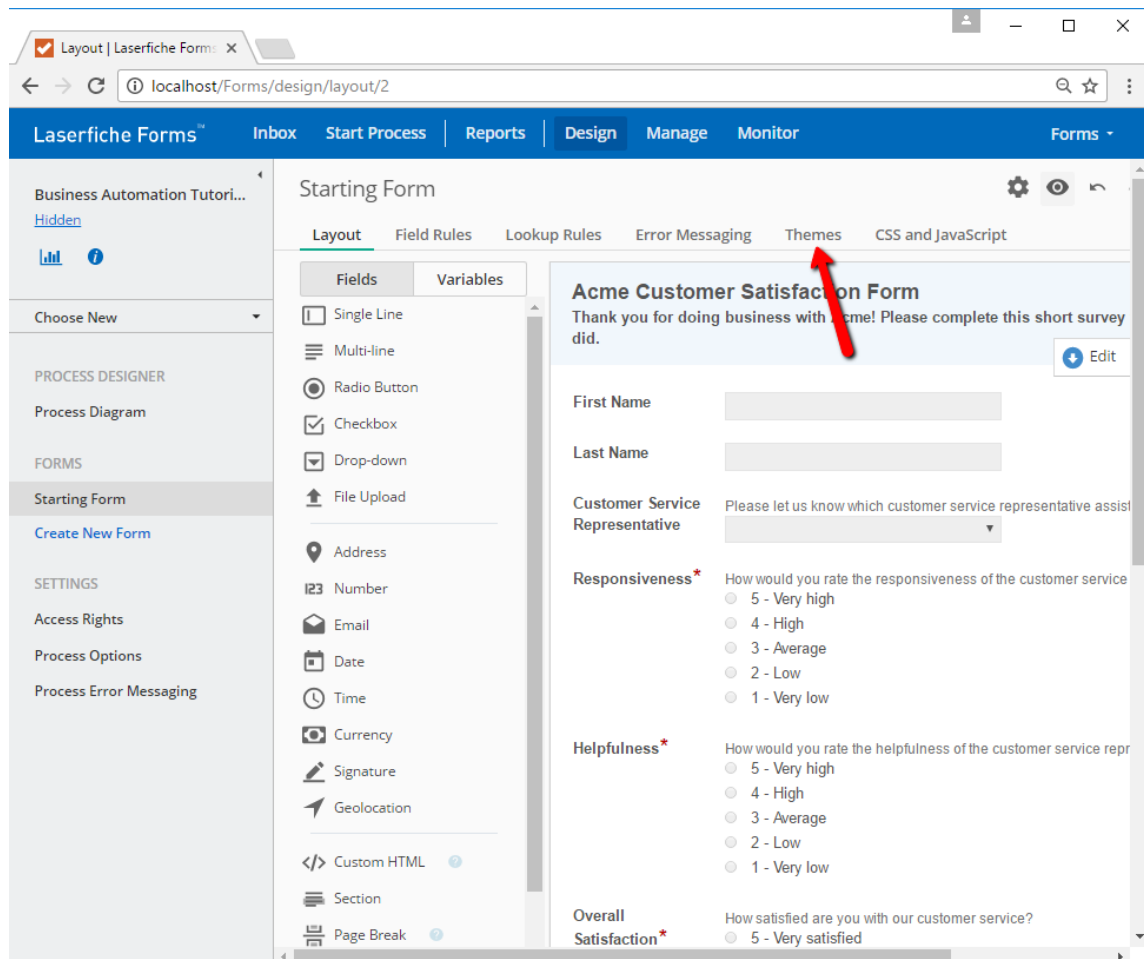


Figure 3.6: Themes link for the active form.

The theme editor offers a variety of themes out of the box. They are designed more as a starting point for theme customization, rather than as production-grade themes. The currently selected theme is called “Basic”, and as the name implies, it’s quite basic. Let’s scroll down and select the “Blueprints” theme, which features a friendly color scheme as well as a fancy background.

After selecting the theme, a preview is shown on the right pane. That said, I always use the **Preview** button to view a full-page preview. The reason is that the version of the form shown in the **Themes** page shows only the selected theme's styling, and does not include any custom scripts or styling we may have on the form. So it's a good habit to rely on the full-page previews to get an idea of what the form will *actually* look like.

Even though the Blueprints theme looks pretty nice, it doesn't have a logo. Let's add one now by going to the **Customize** tab of the theme editor. This tab allows us to make ad-hoc changes to the styling of various parts of the form. We will select the first option, which is **Logo**.

58 CHAPTER 3. BUSINESS PROCESS: CUSTOMER SATISFACTION SURVEYS

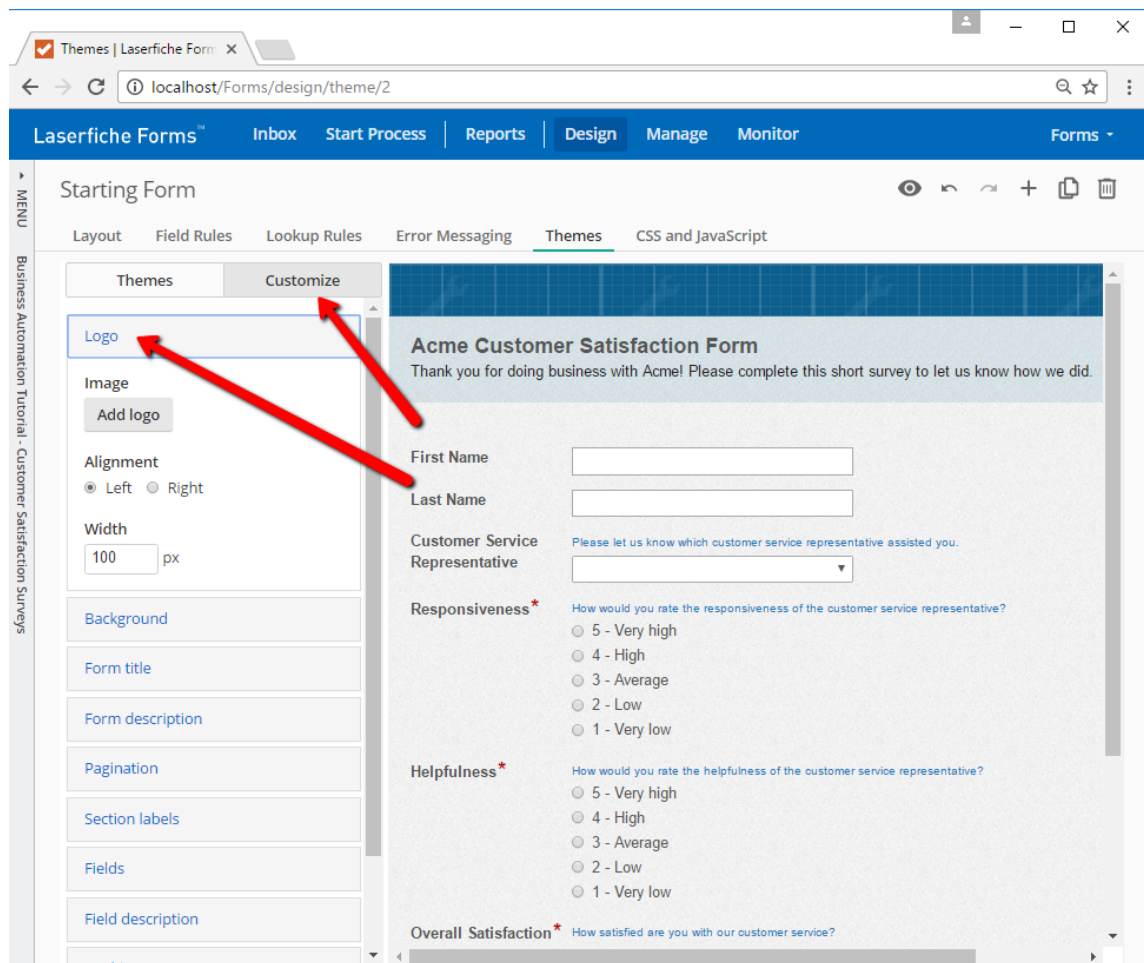


Figure 3.7: Adding a logo by customizing the theme.

In the dialog box that opens, we see our **Image Gallery**, which is currently empty. We have two options. We can either upload an image from our computer, or provide an image URL. The latter option can be useful in scenarios where your company already have a website and you want to use an existing logo asset. In this case, I will use a logo located at “<http://i.imgur.com/3vwC7v>”

¹Image credit: [Bouncy-Bunny](#) on DeviantArt. ©2014-2017 Bouncy-Bunny.

Box 3.3. Good-looking logos

You can use just about any image for a form logo, as long as it has in PNG, JPG or GIF format. However, it's a good idea to use PNG files with transparent backgrounds whenever possible. The reason is two-fold. First, the PNG format is quite light-weight and it is optimized for web use. Second, PNGs that have transparent backgrounds give you more flexibility when designing your forms, because they inherit whatever element you place them on for their background. You can see it in the screenshot below: the logo is currently too large for the form header, but its background seamlessly changes when it spills over to the form itself. A non-transparent background would look quite ugly.



Figure 3.8: PNGs with transparent backgrounds maximize styling flexibility.

There are various tools you can use to make transparent PNGs. Any professional image-editing tool has the capability. I personally use [Clipping Magic](#). It requires a subscription, but it's quite cheap, and I find the price to be worth it because I do a lot of image editing work. Alternatively, if your organization has a marketing department, you can also ask them to provide you with a transparent logo.

The logo we used has dimensions of 200x200, which makes it a little too large. Let's reduce its width to 140 pixels. We can also change its alignment to be right-justified, but I think it looks better on the top left corner.

Once we are done, we should save our new theme by clicking the green **Add to custom themes** button at the top of the form image. That way, we can use it on other forms without having to go through the same song-and-dance with the logo importing and such. We will save this theme as "Acme Industries Theme".

Alright, we are done with the theme editor for now. If you'd like, you can play with some of the other theme customization settings. The theme editor can be very useful for making styling changes to the entire form. For more fine-grained changes, such as changing the font or color of a single form field, we can use CSS.

3.1.2 Field rules

Our Customer Satisfaction Survey form is coming along nicely. One thing we may want to add is the ability for users (i.e. our customers) to provide more detailed descriptions of their experience. Specifically, if someone picks "Very High" or "Very Low/Unsatisfied" on our fields, we want to give them a chance to elaborate.

To do this, we will go back to the **Layout** section and add two Multi-line fields at the bottom. One of them will have a positive tone whereas the other will have an apologetic tone.

We will label our first Multi-line field "Details" and use the following text above field:

We are glad to hear you had a great experience! We would love to hear the details so we can give hearty praise to our representatives!

Let's also increase **Field height** from the default 3 lines to 5 lines. While the number of lines won't actually prevent users from typing essays (we can use the **Character limit** option for that), it tends to be a good visual indicator for the expected length of input.

Now we are going to duplicate this field and change the new one's label to "Suggested Improvements". This field will have the following text above field:

We are very sorry to hear you had a poor experience. Please let us know what we can do better.

When we preview the form, we can see both Multi-line fields we added. However, that's not really what we want. Ideally, the **Details** field should be displayed if the customer had a great experience, whereas the **Suggested Improvements** field should display if they had a poor experience. Fortunately, this type of logic is very easy to implement.

We will switch over to the **Field Rules** tab, the link for which is at the top. Field Rules allow adding conditional logic to forms so that fields are hidden or shown based on values of other fields on the form. This is a very powerful feature and can greatly improve user experience.

By default, forms don't have any field rules. We will now add a new one by clicking the **New field rule** link in the middle.

Let's configure this rule so that the **Details** field is shown when the **Overall Satisfaction** field is either "5 - Very satisfied" or "4 - Satisfied". To add the second conditional, we can click the plus button next to the first one.

Then let's add a second rule for the **Suggested Improvements** field. We will configure this rule to display that field when the **Overall Satisfaction value** is "1 - Very unsatisfied".

Here is what our Field Rules should look like when we are done:

Starting Form Save

Layout Field Rules Lookup Rules Error Messaging Themes CSS and JavaScript

Field rules show or hide fields based on other field values. [Learn more](#)

1 ↑ Show Details and Ignore the data when t + ×

When any of the following is true:

↓ Overall Satisfaction is 5 - Very satisfied + ×

Overall Satisfaction is 4 - Satisfied + ×

2 ↑ Show Suggested Improve and Ignore the data when t + ×

When any of the following is true:

↓ Overall Satisfaction is 1 - Very unsatisfied +

+ Add rule

Figure 3.9: Our first Field Rules.

We need to make sure to click the **Save** button on the top right corner. After that, we will preview our form. If we configured everything correctly, the **Details** and **Suggested Improvements** fields will be hidden when the form opens, and the appropriate one is displayed based on our selection in the **Overall Satisfaction** field.

Box 3.4. More complex field rules

It is possible to add more complex field rules. For example, we can show or hide fields based on the inputs to several different fields, and make the rule even more specific using operators such as “contains” or “starts with”. Keep in mind however that field rules can eventually become too complex and cause conflicts and unexpected behavior. Laserfiche Forms tries to warn us if it detects such issues, but as a rule of thumb, if your form behaves unexpectedly with regards to fields not being hidden or shown correctly, field rules should be the first thing to check.

3.2 Storing forms and sending emails

We are done with the form for the moment. Now let's think about what should happen to submitted forms. If we switch over to the Process Diagram, we can see that the Business Process ends immediately after form submission. Laserfiche Forms will retain the submitted data for reporting purposes, which can be useful, but often times we want to store images and metadata of submitted forms in a Laserfiche repository so that they can be easily referenced by repository users.

Before we do that though, let's go back to the **Management** section in Web Access and create a metadata template.

3.2.1 Metadata templates and fields

We will log into the LFTraining repository as Admin (no password, unless you added one), then switch over to the Management page. On the left side, we will click the Metadata link.

Every new Laserfiche repository contains a set of metadata fields. Some of these are provided because they are commonly used (such as the Date field) whereas others are provided as a reference. We will switch to the Templates tab and click the plus icon to create a new template. We will name it "Customer Survey". We can also optionally provide a description, as well as a color. Description is more for administrative purposes, but template color changes the color of the folder or document that has the template assigned. Let's pick light green for this template.

Now, we need to select the newly created template and add some metadata fields to it. We will do this by clicking the plus icon on the right-hand pane. Let's add these fields:

| Field name | Type | Required? |
|----------------|--------|-----------|
| Customer Name | Text | No |
| Service Rep | Text | Yes |
| Responsiveness | Number | Yes |
| Helpfulness | Number | Yes |

64 CHAPTER 3. BUSINESS PROCESS: CUSTOMER SATISFACTION SURVEYS

| | | |
|----------------------|--------|-----|
| Overall Satisfaction | Number | Yes |
| Submission Date | Data | Yes |

After we add our fields to the template, we need to make sure we use the **Save** button underneath the right pane. Below is what our template should look like:

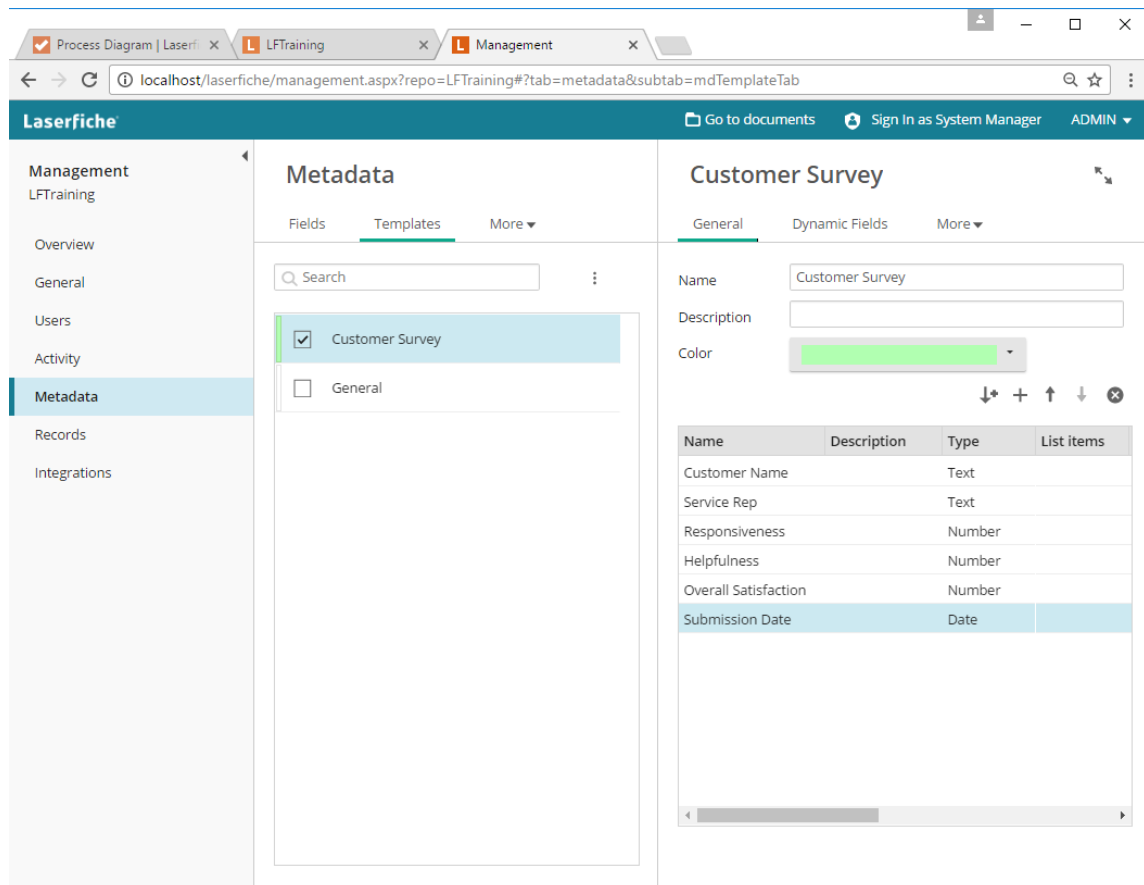


Figure 3.10: Customer Survey metadata template.

You may have noticed a few things when creating the fields above. First, we marked the Service Rep field as **Required**. However, if you recall from our form, it is not set as required there. This is an important thing to mention: if we leave things the way they are, then forms that are submitted with blank

Customer Service Representative fields will not be stored in the repository. The Business Process will be suspended by Laserfiche Forms and an error will be reported.

The second thing is that we specified the **Responsiveness**, **Helpfulness** and **Overall Satisfaction** fields to have the Number type, but their values on the form contain non-numeric characters, such as “5 – Very satisfied”. This, too, will prevent forms from being saved.

Lastly, our Customer Survey Form does not have a date field, but we added one to the metadata template and marked it as Required. So we will need to address that as well.

Let’s go back to Laserfiche Forms and switch to the **Layout** tab of our form to make the necessary edits.

We will start by making the **Customer Service Representative** field required. In reality, customers may not remember the name of the service rep, but for this exercise we will assume our reps make such wonderful impressions that everyone remembers them, for good or ill.

Then, let’s adjust the three Radio Button fields. We marked their corresponding metadata fields to be Number fields. How do we handle that here?

One option would be to change the choices to numbers, for example from “5 – Very satisfied” to simply “5”. This would work, but it’s not ideal. On surveys especially, it’s a good idea to attach descriptors to each survey option so that the user gets an accurate idea of what each rating stands for.

Fortunately, Laserfiche Forms has a neat feature that allows us to handle this scenario: we can keep the choices the way they are, but assign them specific values, which would be their numeric representations. We can do that by editing each Radio Button field and checking the **Assign values to choices** option, then adding the corresponding numeric values, as shown below:

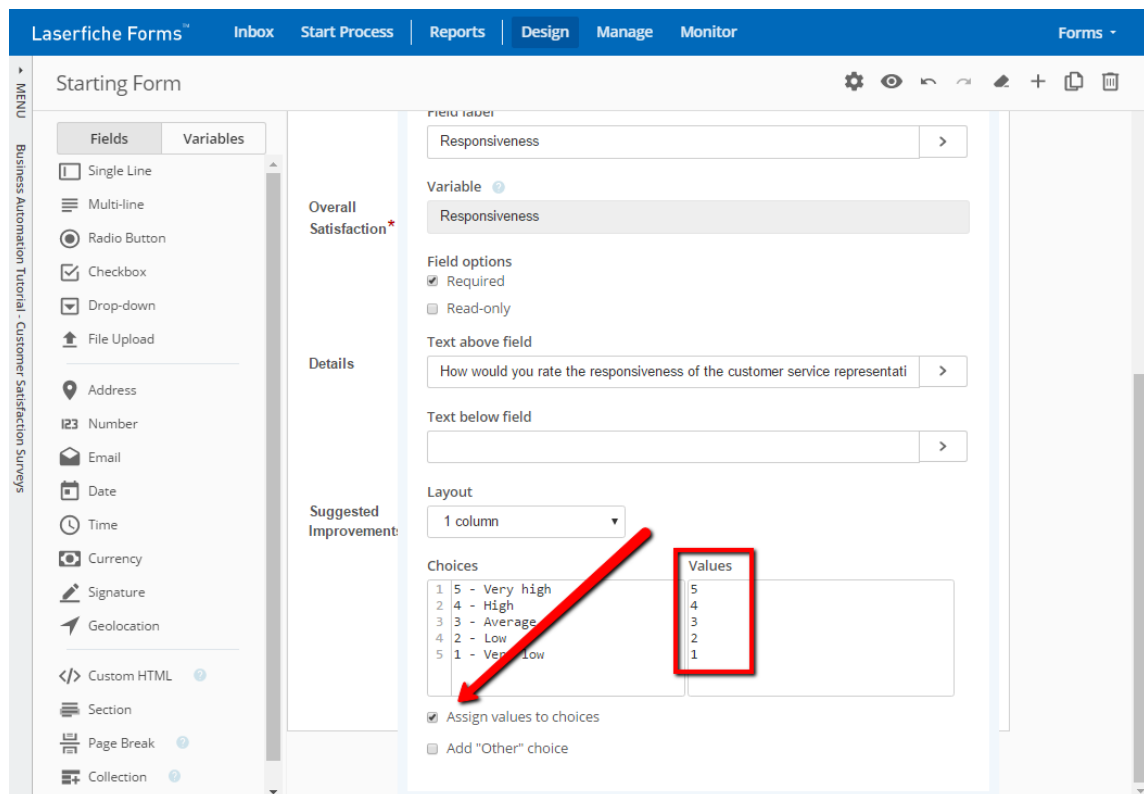


Figure 3.11: Assigning numeric values to choices.

Once we do that for all three Radio Button fields, customers will still see the full descriptions for each radio button, but behind the scenes the values will be saved into their metadata fields as numbers.

The last thing we need to handle is the submission date, which currently does not exist on our form as a discrete field but we added to our metadata template as a required field. There are two ways to handle this (actually there are more than two ways, but we are keeping things simple for now).

- **Option 1:** We can add a Date field to our form and set its default value as Today's Date. We can then hide the field using Field Rules or CSS.
- **Option 2:** We can use a built-in variable in the Save to Repository Service Task (which we are about to add).

Generally speaking, if we can avoid adding more fields to a form, we should. Doing so keeps forms light-weight and easy to manage. So we are going to go with Option 2. Let's switch to the Process Diagram and configure a Save to Repository Service Task.

3.2.2 Save to Repository service task

If you remember from [Section 2.1.5](#), we can add this service task by dragging and dropping it from the process toolbox on the left side. We will drop it right on top of the arrow that currently connects the Message Start Event and the End Event. Once the arrow is highlighted in yellow and we release the mouse button, it will lodge nicely into place. Alternatively, we can place it elsewhere and connect the steps manually.

We will then double-click the service task and create a repository profile. Repository profiles have to be configured for each Business Process, but once configured, they can be re-used across the different Save to Repository service tasks in that process. We will configure the LFTraining repository and use the Admin user. Once we create the profile, we will select the **Save the submitted form from this process step** option and pick the Starting form.

This time, let's be more specific with the way we store the forms in the repository. We will use these options:

```
Document name: Customer Survey Form
Path: \Customer Surveys\{/dataset/Customer_Service_Representative}
Save as: TIF
```

You may have noticed the “{/dataset/Customer_Service_Representative}” value we used as part of the Path. This is called a “variable”. You can think of it as a container that contains some piece of data. In this case, we are using the **Field Variable** for the **Customer Service Representative** field, so that the value the user picks in that field will determine the subfolder in which the form is stored. There are also **Process Variables**, which come from the overall process rather than a specific form field.

68 CHAPTER 3. BUSINESS PROCESS: CUSTOMER SATISFACTION SURVEYS

You don't actually have to type the variable name manually. You can use the **Insert Variables** button attached to the field and pick the field from the dropdown list:

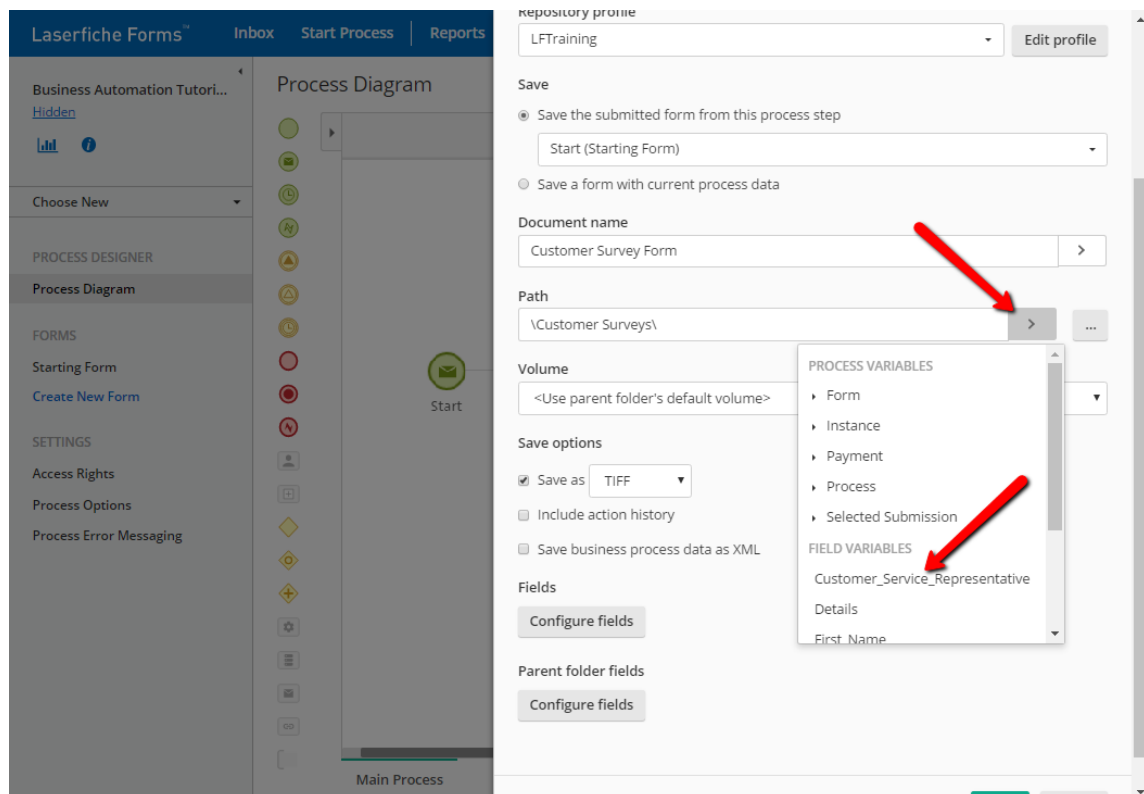


Figure 3.12: Inserting the Customer Service Representative variable into the Path.

Next, let's map the form fields to the metadata template we created in the repository. We can do this by clicking the **Configure Fields** button for Fields:

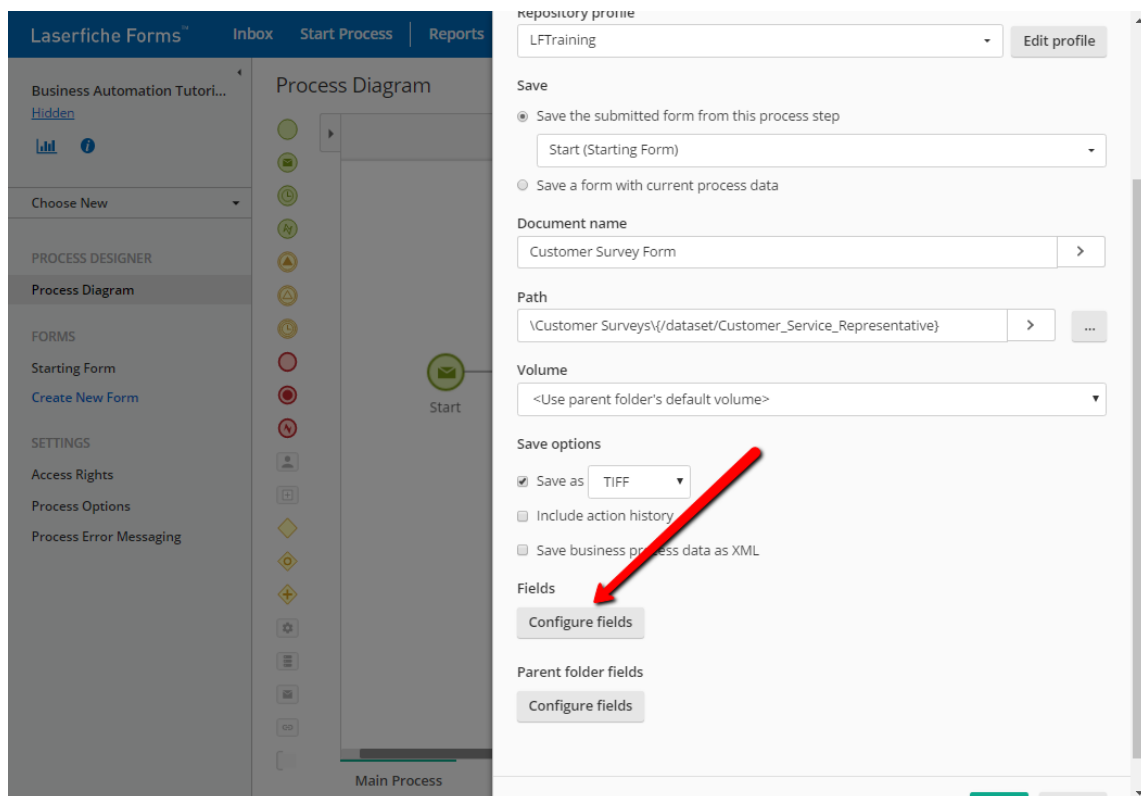


Figure 3.13: Configuring fields for the stored form.

In the Configure Fields dialog box that opens, we will first select the Customer Survey template from the dropdown. This will show us the list of fields that are in that template. Then, we will start mapping each form field to its corresponding metadata field. We can do this by using the Insert Variables button next to each field, just like we did above when we inserted the **Customer Service Rep** field's variable into the Document name.

Box 3.5. Customer name field

Our form has **First Name** and **Last Name** fields, but the metadata template has a single corresponding field called Customer Name. For these types of scenarios, you can simply enter both variables into the field, and separate them with a space.

70 CHAPTER 3. BUSINESS PROCESS: CUSTOMER SATISFACTION SURVEYS

We have Field Variables for each metadata field except for Submission Date. For that field, we will use a Process Variable, which will be the instance start date.

The image shows a 'Configure Fields' dialog box with a close button (X) in the top right corner. The dialog is divided into several sections:

- Template:** A dropdown menu showing 'Customer Survey'.
- Customer Name:** A text input field containing the variable `{/dataset/First_Name} {/dataset/Last_Name}` and a right-pointing arrow button.
- Service Rep:** A text input field containing the variable `{/dataset/Customer_Service_Rep}`.
- Responsiveness*:** A text input field containing the variable `{/dataset/Responsiveness}`.
- Helpfulness*:** A text input field containing the variable `{/dataset/Helpfulness}`.
- Overall Satisfaction*:** A text input field containing the variable `{/dataset/Overall_Satisfaction}`.
- Submission Date*:** An empty text input field with a right-pointing arrow button.

A context menu is open over the 'Submission Date' field, showing a tree structure of variables:

- PROCESS VARIABLES
 - Form
 - Instance
 - ID
 - Initiator
 - Start Date (highlighted with a red arrow)
 - Start Time
 - Payment
 - Start Date
 - Process
 - Selected Submission

At the bottom of the dialog, there is a 'Fields' section with a button labeled 'Add / remove fields'. At the very bottom, there are two buttons: 'Done' (in a green box) and 'Cancel' (in a grey box).

Figure 3.14: Using the Instance Start Date variable.

Box 3.6. Business Process instances

Throughout the book, we will occasionally use the term “instance”. An instance is simply that: a specific instance of the given Business Process. Each instance has its own variables as well as a unique identifier, called an **Instance ID**.

3.2.3 Exclusive Gateway

We are also going to use an Email Service Task to email the Customer Service Manager when a new survey form has been submitted.

However, we will be more specific here than we were back in [Chapter 2](#). Instead of emailing the manager for every single submission (which can get overwhelming quickly, since Acme Industries serves hundreds of customers every day), we will email them only if the **Overall Satisfaction** field says the customer was very satisfied. How do we do this?

Enter “Gateways”. Gateways are used to add routing logic to Process Diagrams. They can be Exclusive, Inclusive or Parallel. We will explore them in more detail in [Chapter 5](#). For this process, we will use an Exclusive Gateway, which is represented by the yellow diamond icon in the process toolbox on the left side. Let’s drag and drop this between the Save to Repository service task and the End Event.

Before we configure the gateway, let’s add a few more things to the Process Diagram. The first will be an Email Service Task, which we will place above the gateway. Then, we will add a second End Event, which we will place to the right side of the Email Service Task. Then we will select the gateway by clicking it once, and drag the little arrow on it and connect it to the Email Service Task. Then we will do the same to connect that to the End Event. Each item on the Process Diagram has connecting points, which appear only when you drag a connector and hover it over its boundaries, as shown below:

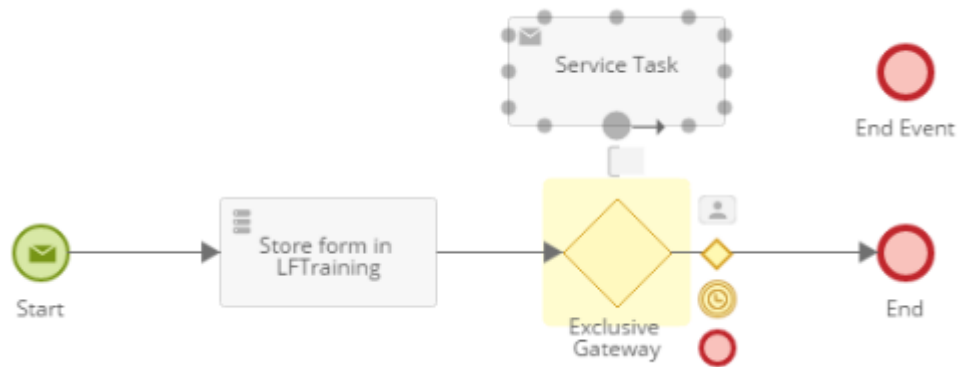


Figure 3.15: Connecting items in the Process Diagram.

Now we will configure the gateway by double-clicking it and switching to the Outflow tab. The Outflow tab is where we can define conditions for our gateway. Because it is an Exclusive Gateway, only one of the defined paths will be followed when the process runs.

In our Process Diagram, we defined two possible paths after the gateway. These paths are represented in the Outflow dialog. The first pathway has the original End Event as its next step, whereas the second one has the Email Service Task. We will mark the first one as the default by checking the **This is the default outflow path** checkbox. Then, we will enter a condition for the second one.

When you click the Conditional Expression field, an overlay appears. This overlay helps define a condition based on Variables. From the first dropdown field, we will select **Overall_Satisfaction**, which is the Field Variable for the **Overall Satisfaction** field. Then we will use the = operator, and enter “5” as the value. When we click the green **Insert** button, the condition will be entered into the field. Finally, we will name this path “Great performance” so that it becomes easy to distinguish when looking at the Process Diagram.

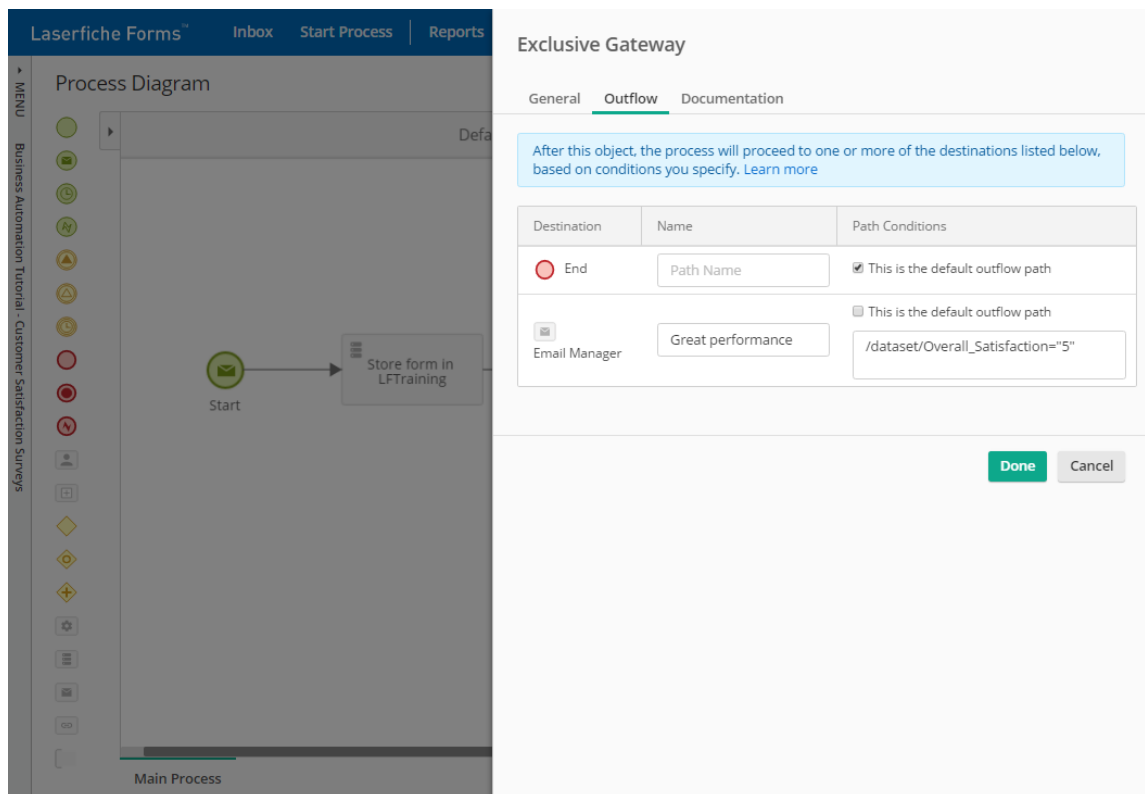


Figure 3.16: Configuring the Exclusive Gateway.

Box 3.7. Variable values in conditionals

You may be wondering why we used “5” instead of “5 – Very satisfied”. That’s because we assigned “5” as the value for that field. This is a potential *gotcha* when designing processes. You need to pay attention to what the actual values are, if you have assigned any.

3.2.4 Email service task

The last thing we have to do before testing the process is configuring the Email Service Task. Let's do that now.

If you remember from [Chapter 2](#), the **From** field will be auto-populated based on our Email Server configuration. We can override that if needed, but in most cases we should use the default value. That way, if users have any inbox rules configured for emails coming from Laserfiche, they don't accidentally miss the notifications.

For the **Send to** field we will use the user account we configured in the previous chapter, which is "user@lfttraining.com". Our subject line will say "New survey form submitted by a very satisfied customer!". We will also attach the form to the email as a file - that way, the manager can open it to see if the customer entered any details into the **Details** field on the form.

Lastly, we will write something in the email body. There are views we can use. The first view is the Visual view, which is a rich-text editor that lets us use various markups and formatting options, such as bolding and list numbering. Let's start in this view and type the following:

```
Greetings Manager!
```

```
A customer just submitted a survey form and indicated they were very satisfied with the service provided by {\dataset\Customer_Service_Representative}! You should go high-five them!
```

```
Cheers,  
Laserfiche Forms
```

We will also select the word "high-five" and make it bold.

You will notice we used the Field Variable for the **Customer Service Representative** field again. Don't worry, the email editor also has a variable picker dropdown menu. I will leave it up to you to find it though. (Hint: look at the toolbar ;))

Let's add one more thing for fun. I mentioned the Visual view is one way to create these emails. The other view is the HTML view, which shows the

76 CHAPTER 3. BUSINESS PROCESS: CUSTOMER SATISFACTION SURVEYS

HTML markup underneath the rich text. Go ahead and switch to that using the **Html** tab on the top right corner of the editor.

What you are looking at is HTML. If you have done any web design or programming before, it will look familiar. If not, don't worry. In the next chapter we will give a quick rundown of different HTML tags and what they stand for. For now, find the part that says:

```
<b>high-five</b>
```

And change it to the following:

```
<b style="color:red">high-five</b>
```

As you may have guessed, we just made the word “high-five” red. Now when you switch back to the Visual editor, you should see it stand out:

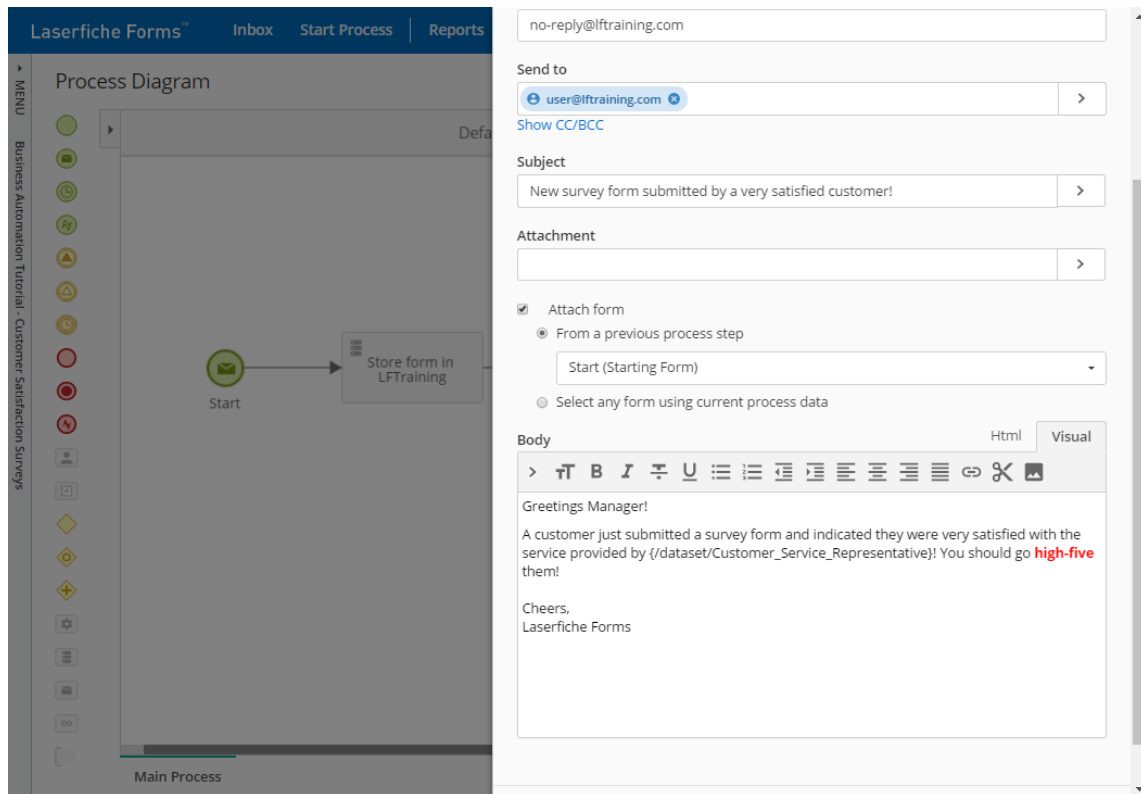


Figure 3.17: High-fiving with high energy!

Perfect! We are now done with the Process Diagram. To make sure we have configured everything correctly, we can use the Validate button. This will tell us if there are any glaring issues with our configuration. I always validate my diagrams before saving them. It has saved my neck quite a few times!

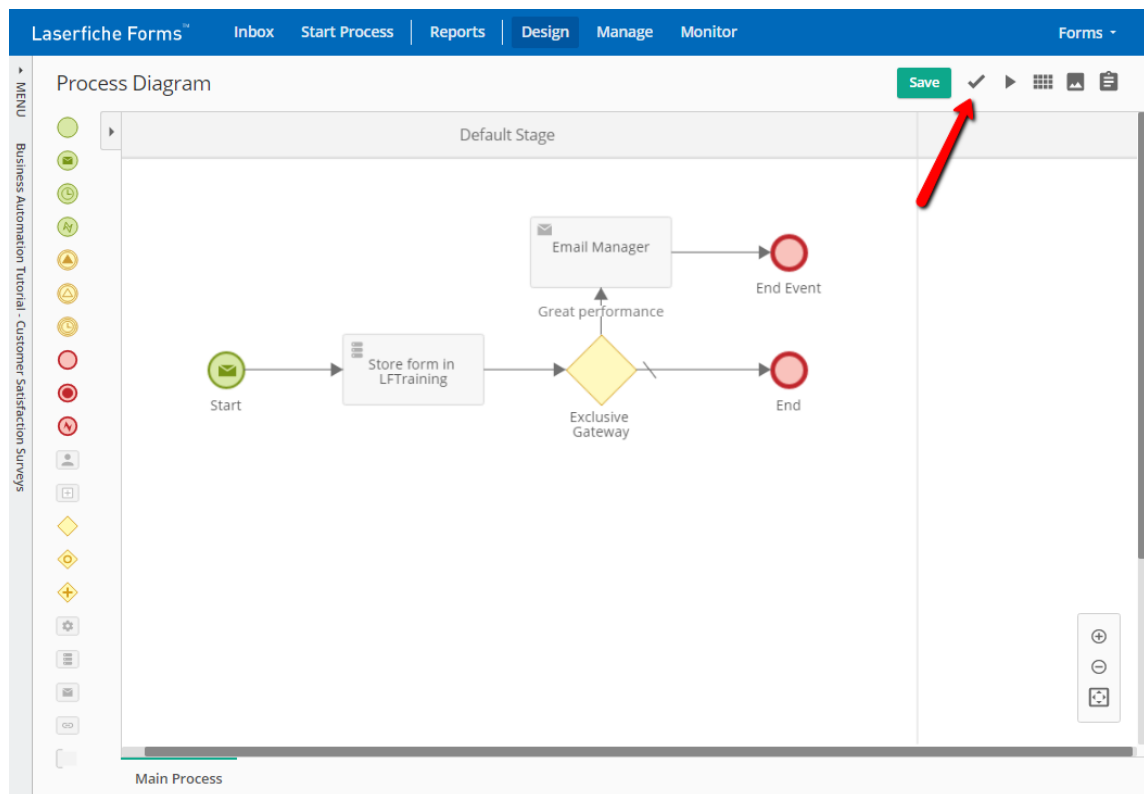


Figure 3.18: Validating the completed Process Diagram

Let's test our process! We will run two tests: first we will submit a form where we will pick "Bobby Drake" as the customer service representative and "5 - Very satisfied" for the **Overall Satisfaction** field. Then, we will submit another form, and this time pick "Jonathan Smith" as the rep and "1 - Very unsatisfied" for **Overall Satisfaction**. If we configured everything correctly, the following should happen:

1. The forms should display the **Details** and **Suggested Improvements** fields according to the Field Rules we have defined.
2. Both forms should be stored in the repository in the appropriate subfolders (based on the Customer Service Representative we pick)
3. We should receive only *one* email. This email should have a copy of the

submitted form attached to it, and the word “high-five” should be bold red.

If you configured everything correctly and your tests work as expected, you are ready to finish this chapter and proceed to the next one. If not, make sure to go over the corresponding steps again. Designing forms and processes correctly can take several iterations - don't give up!

3.3 Chapter summary

In this chapter, we designed a brand new Business Process for our company's customers to be able to submit customer survey forms. We looked at the following topics:

- Creating forms with different field types and options
- Customizing the look-and-feel of forms
- Adding Field Rules to make forms more user-friendly
- Mapping form fields to repository metadata fields
- Controlling process flow using Exclusive Gateways
- Styling emails using HTML

I would encourage you to continue playing with this Business Process, especially if you are curious about different tools and options we haven't yet covered. In the next chapter, we are going to create a more detailed Business Process, which will have database lookups, custom CSS and JavaScript as well as User Tasks with deadlines.

Chapter 4

Business Process: Purchase Orders

In [Chapter 3](#), we created a Customer Satisfaction Survey form and stored it in the LFTraining repository, with a conditional process rule that emailed the Customer Service Manager if the customer was very satisfied with the service.

In this chapter, we will develop a new Business Process, this time for the submission and approval of purchase orders. After a purchase order form is filled out and submitted, it will go to the submitter's department manager for approval. We will learn about User Tasks, database lookups and error message configuration. In addition to several important features in Laserfiche Forms, we will also learn about ancillary concepts related to database administration and web programming.

Box 4.1. Database server access

You will need access to a Microsoft SQL Server 2012 or later for this chapter and need to be able to create and manage databases on it. If you followed [Chapter 2](#) to create an isolated training environment, you already have a Microsoft SQL 2012 Express instance installed locally, as well as SQL Management Studio. However, if you are utilizing your company's SQL Server (which will be located on another

server on the network), make sure to speak to your IT administrator so that they can provide you with an account that has sufficient permissions to create databases.

The purpose of this Business Process is to allow our company's employees to make requests to purchase materials and equipment. For example, if a construction worker notices that their crew will need more concrete soon, they should be able to make a purchase request. If the request is approved by the Construction Manager, the worker should be sent a unique purchase order number and use it to make the purchase. It would also be nice if the Chief Operations Officer of our company could run reports on this process to see where the bottlenecks are.

Let's go ahead and dive right in! We will begin by clicking the Design link on the navigation bar at the top and selecting the New option on the left side. This time, we will start with a Form Approval template. The template we pick here does not matter too much - it just populates our Business Process with a few items so that we are up to a quick start. We will name this one "Business Automation Tutorial - Purchase Orders". Once it is created, we will start editing the Starting Form.

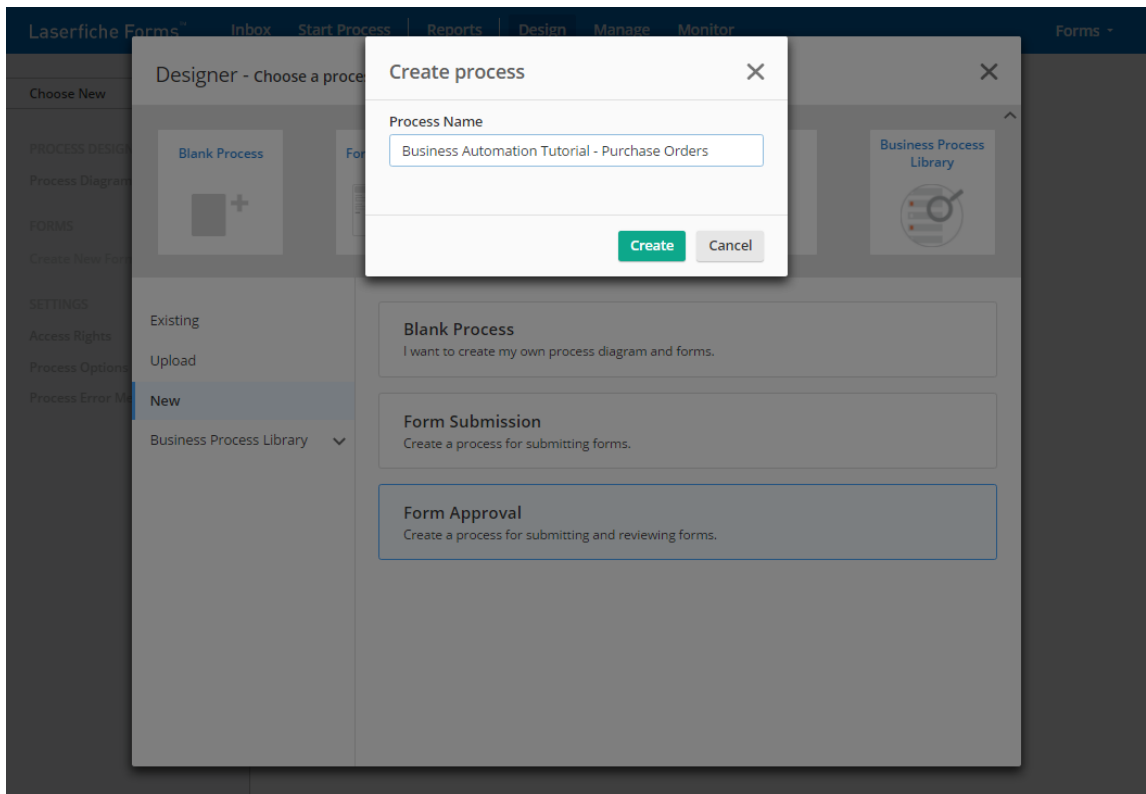


Figure 4.1: Creating our process using the Form Approval template.

Box 4.2. Starting with the form layout

You may have noticed that we are always starting our process development by creating and editing a form. This is because Forms constitute the lower layer of a Business Process, and it is easier to go bottom-up rather than top-down. Creating a form first allows us to utilize the Field Variables on that form when designing the Process Diagram later. This is not the only way to create Business Processes - you can design a rough mock-up of the process first and then create the forms and their fields. It's a matter of preference.

4.1 Creating the Purchase Request form

We will start by adding a title and description.

```
Form title: Purchase Request Form
Form description: Use this form to specify the materials you are requesting
and which projects they are for.
```

On the top left corner, you may notice that the Form name is **Starting form**. You may be wondering what that is and how it is different from the Form title we just entered. You can think of the Form title as the user-facing title, and the Form name as the administrator-facing name that distinguishes the form from the others in the Business Process. And since this Business Process will have a second form, it's a good idea to give each one more specific names. We can do this by clicking the Form Settings button, represented by the gear icon on the top right corner:

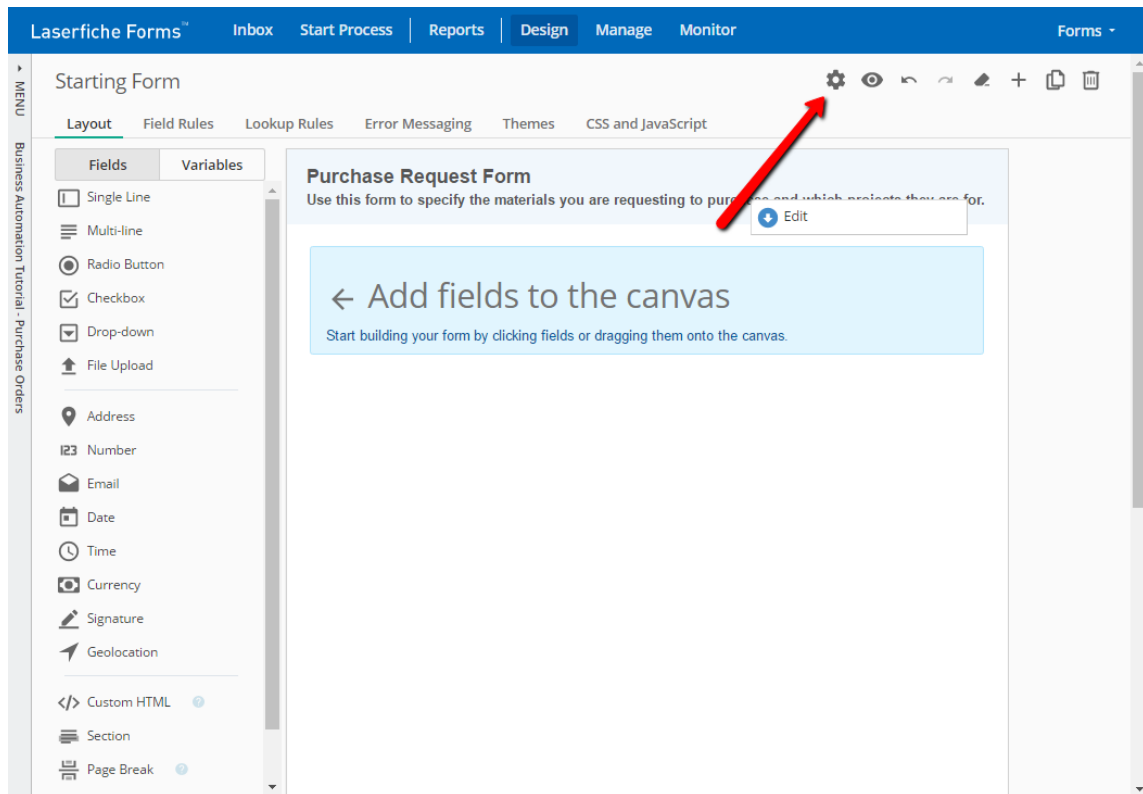


Figure 4.2: Accessing the Form Settings dialog.

It's a good idea to keep form names and form titles identical, or at least similar. So we will name this form "Purchase Request".

There are several other options here that are worth mentioning. The first set of options deal with form layout. By default, forms use what is called a "responsive" layout. Responsive in this context means the layout of the form changes automatically based on the screen size of the device the form is viewed on. This is ideal if we expect our users to fill out the form on a variety of device types, including desktop computers, laptops, tablets and smartphones.

We can also adjust the maximum width of the form, which can be useful if the default maximum width is not enough to properly fit all the elements. For example, if we have a table with lots of columns, then we may want to increase the maximum width to 1000 or 1200 pixels to make sure the table columns are wide enough for their fields.

Box 4.3. Ideal form width

I try to keep forms at the default 800 pixel width whenever possible. The reason is that wider forms, when stored in the repository, can lose their crispness and appear blurry, making their text difficult to read. There are ways to get around this that we will mention later, but it is worth keeping in mind regardless.

We can also change the alignment and width of field labels. The default alignment is for the label to be on the left side, but Top is also a common option. Generally speaking, these settings are worth playing with if we have fields that have long labels in them. We will leave them on their default settings for the time being.

Lastly, there is a backend validation setting. This is a new setting added in version 10.2. Prior versions relied on front-end validation only, meaning it was up to the user's browser to make sure things like required fields were filled out and their format was correct. With backend validation, the Forms Server adds a second layer of control to ensure all data is valid. We will leave that at its default setting as well, which is "validate user input".

Before we add any fields, let's change the Theme of our form to the one we created in [Chapter 3](#). We will switch over to the Themes section, and under Custom themes we will select the **Acme Industries** theme.

Now let's add the following fields to our form:

| Field label | Type | Required | Read-only | Default value | |
|--------------------|-------------|----------|-----------|-------------------------|--|
| Date | Date | No | Yes | current_date | |
| Employee Name | Single-line | Yes | No | {/_currentuser_display} | |
| Crew Leader | Dropdown | Yes | No | | |
| Summary of request | Multi-line | Yes | No | | |

Previewing the form now should show us the following:

Preview Mode: Data entered will not be submitted.

ACME INDUSTRIES
It Might Just Work!

Purchase Request Form
Use this form to specify the materials you are requesting to purchase and which projects they are for.

Date Date captured on form submission

Employee Name*

Crew Leader*

Summary of request*

Figure 4.3: Our half-built Purchase Request Form.

The first thing you will notice is that the **Employee Name** field says “Forms”. That’s because the field has a default value of “`{/_currentuser_display}`”, which is the display name of the current user. If the current user does not have a name configured in their account, the system falls back on the user name, which in this case is Forms.

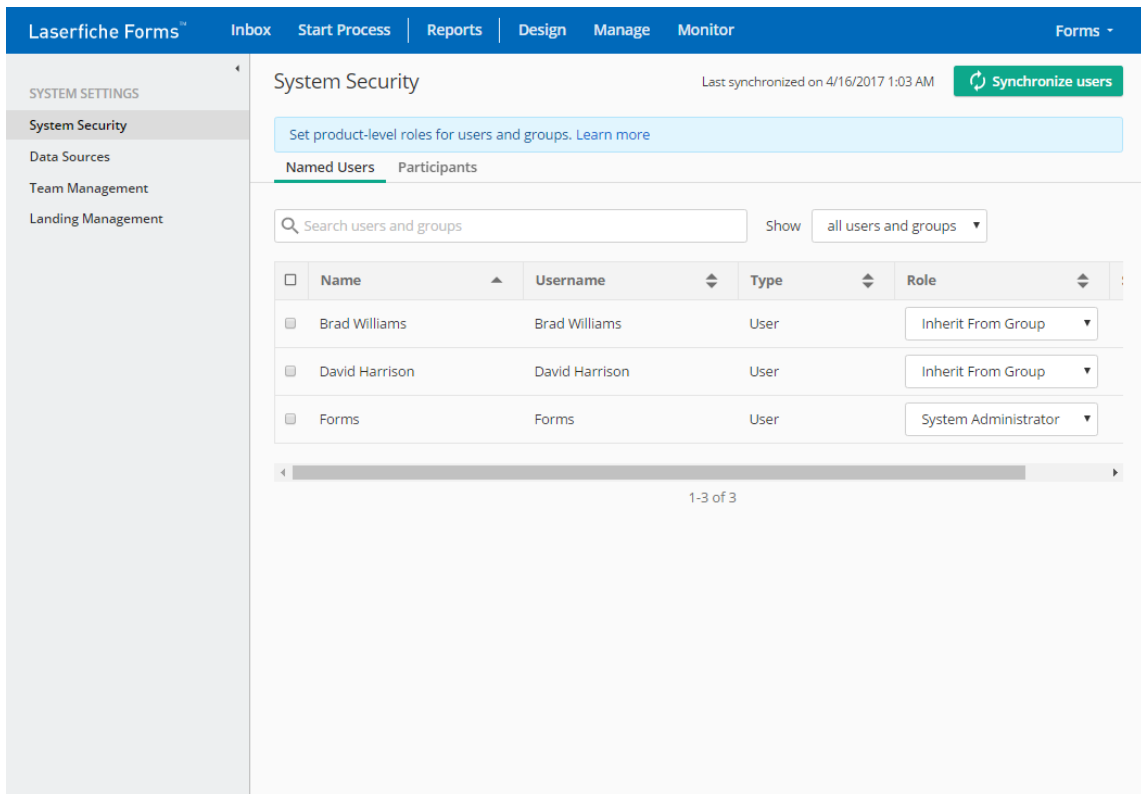
We want to use “real” users for this Business Process though. So let’s go to the repository management page and create a couple of new users. As a reminder, you can get to repository management by logging into the repository via Web Access, then using the Management link at the top right corner menu. You will also need to log in as a System Manager in order to be able to allocate named licenses to these new users.

We will name our new users “David Harrison” and “Brad Williams”. We will make both of them Full Named Users, and give them easy to remem-

ber passwords (we will be logging in as each one when testing our Business Process later).

So now let's go back to Laserfiche Forms and try to log in as these users... but wait. Recall from [Section 2.1.4](#) that we first need to manually synchronize the account list in Laserfiche Forms with that of the Laserfiche Server. We *could* wait for up to 24 hours for it to happen automatically, but we would rather not since we are in the middle of a tutorial and we don't want to lose momentum!

While still logged into Laserfiche Forms as the Forms user, go to the menu on the top right corner and click the **Administration** link. You will find yourself in the System Security page. On the top right you will see a green button with the label **Synchronize users**. Click that, then click Yes. This will refresh the user list, and Brad and David will now be in it.



The screenshot shows the Laserfiche Forms interface. The top navigation bar includes 'Laserfiche Forms™', 'Inbox', 'Start Process', 'Reports', 'Design', 'Manage', 'Monitor', and 'Forms'. The left sidebar lists 'SYSTEM SETTINGS' with sub-items: 'System Security', 'Data Sources', 'Team Management', and 'Landing Management'. The main content area is titled 'System Security' and shows 'Last synchronized on 4/16/2017 1:03 AM' and a green 'Synchronize users' button. Below this is a blue box with the text 'Set product-level roles for users and groups. Learn more'. There are two tabs: 'Named Users' (selected) and 'Participants'. A search bar is present with the text 'Search users and groups' and a 'Show' dropdown menu set to 'all users and groups'. A table lists the users:

| <input type="checkbox"/> | Name | Username | Type | Role |
|--------------------------|----------------|----------------|------|----------------------|
| <input type="checkbox"/> | Brad Williams | Brad Williams | User | Inherit From Group |
| <input type="checkbox"/> | David Harrison | David Harrison | User | Inherit From Group |
| <input type="checkbox"/> | Forms | Forms | User | System Administrator |

At the bottom of the table, there is a pagination indicator '1-3 of 3'.

Figure 4.4: Our freshly synchronozed user list in Laserfiche Forms.

We will eventually test out form with these two users, but we are not quite ready to do that because the form is not finished. However, we are making great progress. Let's click the Manage link on the top navigation bar, then click Edit next to our Business Process. We will make a few more additions to the Purchase Request form.

Currently, the form contains four fields. One of these fields allows entering a summary of the request. However, construction crew members need to be a bit more specific. The form description says they also need to specify what materials we need and for which projects. How can they do that on this form?

Laserfiche Forms allows two different tools for this: Collections and Tables. They are actually the same data structure underneath, but fields in a Collection are laid out vertically whereas those in a Table are laid out horizontally on columns. They are also created a bit differently. Let's go ahead and use a Collection - I find them easier to work with. We will drag and drop one onto the canvas.

We will label our Collection "Materials". We will also modify its options such that users can enter at minimum 1 set and at most 3 sets of items per Purchase Request. We will also change the text for the add set link to "Add another item". Then we will click Done.

Our collection is currently empty. We need to add some fields to it. We will add the following:

| Field label | Type | Required | Read-only | Choices |
|----------------|-------------|----------|-----------|--------------------------------|
| Project Name | Single-line | Yes | No | |
| Project Number | Number | Yes | Yes | |
| Item | Single-line | Yes | No | |
| Category | Dropdown | Yes | No | Concrete, Scaffolding, Power T |
| Cost | Currency | Yes | No | |

The last field in our **Materials** collection is the **Cost** field, which represents the dollar cost of the specified item. Since there can be multiple items being requested, it would be good to also have a field that shows the sum of all values entered into the Cost fields. Let's go ahead and add a Currency field after **Materials** (outside of it) and label it "Total". Make it both **Required** and

Read-only.

4.1.1 Automatic calculations in Laserfiche Forms

You may be wondering how the **Total** field will be calculated. Are we going to ask our users to have a calculator handy so that they can sum up the cost of items they are requesting to purchase? Hopefully not, because that would slow users down quite a bit and probably annoy them as well!

Prior to Laserfiche Forms 10, field calculations had to be performed using custom JavaScript. While this gave form designers a lot of power, it also required expertise in web programming and slowed down Business Process development. Starting with version 10.0, Laserfiche Forms has a built-in feature that allows specifying formulas inside fields. These formulas follow the [OpenFormula](#) standard, so anyone who is familiar with formulas in popular spreadsheet software such as Microsoft Excel should feel right at home with this feature.

Let's go to the field options for the **Total** field and switch to the **Advanced** tab. The first option inside that tab is **Formula**. There is a plethora of formulas we can use, but for now, we will enter the following:

```
=SUM(Materials.Cost)
```

As you can probably deduce from looking at the formula, it says “calculate the sum of all Cost fields inside the **Materials** collection”.

Now when we preview the form and populate some **Cost** fields, the sum will be calculated automatically and the **Total** field will be populated with it. Pretty awesome!

4.1.2 Database lookups

Before we add more fields to our form, let's consider the user experience. The form auto-populates the **Employee Name** field when it is opened, which is great. But then the user has to pick their crew leader from a dropdown menu. Shouldn't we already have this information somewhere?

Similarly, the user has to populate both the **Project Name** and **Project Number** fields manually. These are required fields, so if they don't have either piece of information, they will either make something up (just to be able to submit the form) or have to ask their coworkers. Neither of these scenarios is desirable. It would be great, for example, if we could populate the **Project Number** based on the **Project Name** that is entered.

Laserfiche Forms has the ability to perform lookups and run stored procedures on ODBC-compliant databases. In this section, we are going to create two database tables, one containing employee and crew information, and another one containing project information. We will then tell Laserfiche Forms about these tables, and configure some lookup rules in our form to pull data from them.

As mentioned in [Box 4.1](#), you will need access to a Microsoft SQL Server 2012 or later. Specifically, you need to be able to use a database management tool, preferably Microsoft SQL Management Studio, to be able to log into the database server and create objects on it. The rest of this chapter will assume that you have everything installed locally or you know your way around database management in general.

Creating SQL tables

Let's go ahead and launch SQL Management Studio and log into our database server.

Box 4.4. SQL Server name

In the **Connect to Server** dialog box, you have to pay special attention to the Server Name field. If you installed SQL Express as covered in [Chapter 2](#), then this field should say "localhost\sqlexpress".

Inside the Object Explorer pane on the left side, we have a list of nodes. Let's expand the Databases node. We should see two databases: **Forms** and

LFTraining. The former is the database for Laserfiche Forms, whereas the latter is the database for our LFTraining repository. If you have other Laserfiche modules installed, such as Workflow, their databases will also be listed here.

We are going to create a new database by right-clicking the databases node and selecting the **New database...** option. The database creation dialog has a host of different options, but for us the only thing that is important at this stage is the database name, which we will enter as “AcmeTables”. Then we will click OK.

The AcmeTables database should be added to the Object Explorer pane. Let’s expand it and right-click its Tables node, and select the **Table...** option. This will launch a new tab on the main pane where we will be able to design our new table.

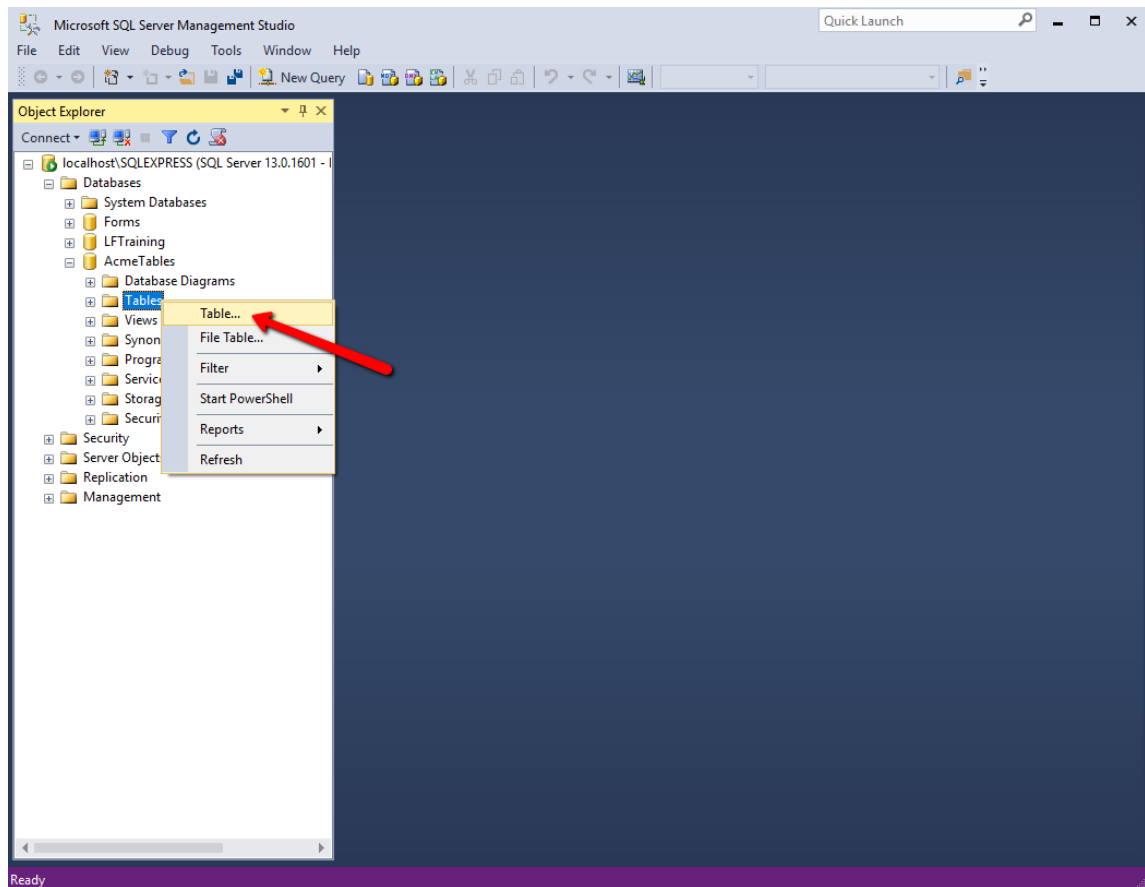


Figure 4.5: Opening the table designer.

We should have three options: Column Name, Data Type, and Allow Nulls. Column name should be self-explanatory: it is essentially the label for the data that will be in that column. Data Type contains a large variety of SQL data types, most of which are not relevant to us. Finally, Allow Nulls lets us specify whether a value is required for cells under that Column. Let's create a small table with the following specs:

| Column Name | Data Type | Allow Nulls |
|-------------|--------------|-------------|
| Worker Name | nvarchar(50) | No |
| Crew Leader | nvarchar(50) | Yes |

We are going to allow null values in the **Crew Leader** field, because some workers may not have a crew leader assigned yet.

Box 4.5. SQL data types

Microsoft SQL has many different data types, each serving a specific purpose. For example, the **int** type is for integers from -32,768 to 32,768, whereas **datetime** is for dates that are combined with a time of day. There is often more than one data type that can be used for a given piece of data. For example, it is possible to represent monetary or currency values using the **money** data type as well as the **decimal(19,2)** data type. We won't really get into the specifics of SQL data types in this tutorial. Just know that, for simple relational tables such as the one we created, **nvarchar** is good enough in most cases, and **nvarchar(50)** means the string value can be a variable number of characters up to 50, which is reasonable for data that will represent people's names.

Once we are done with designing our table, we should save it using Ctrl+S, or by clicking the Save button on the toolbar. We will name our new table "WorkerList".

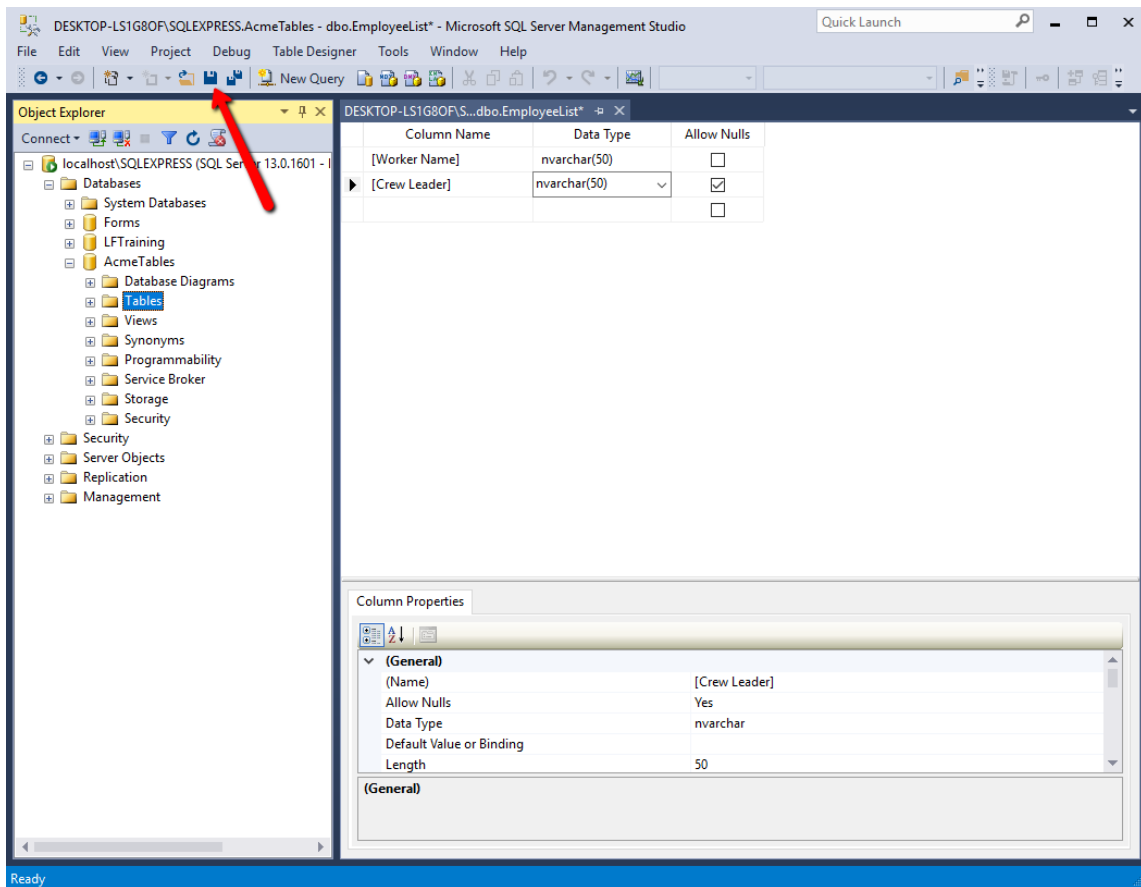


Figure 4.6: Saving our WorkerList table.

Now, when we select the Tables node of AcmeTables and press F5, we should see the new table we created.

While here, let's also create a table for our projects. Once again, open the table designer and create a table with the following specifications:

| Column Name | Data Type | Allow Nulls |
|----------------|---------------|-------------|
| Project Name | nvarchar(100) | Yes |
| Project Number | int | Yes |
| Project Status | nvarchar(10) | Yes |

We will name this table 'ProjectList'. Once we save it and refresh the

Tables node in our database, we should see both of our tables.

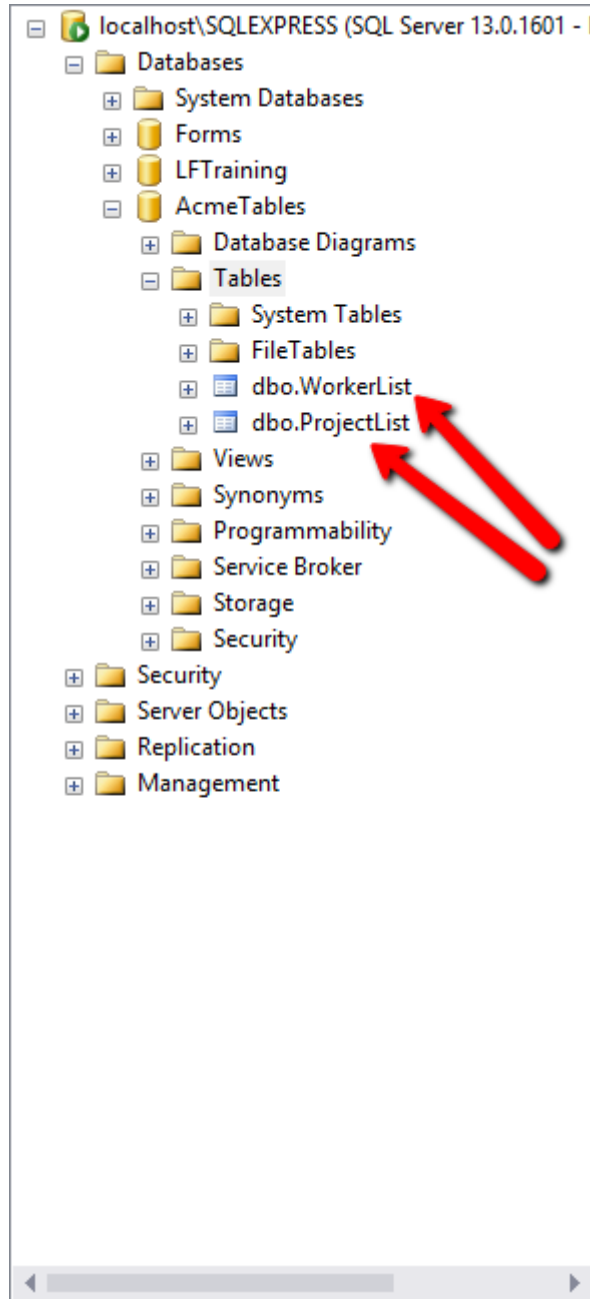


Figure 4.7: WorkerList and ProjectList tables.

Now that we created our tables, let's put some data in them. The way we do this is by right-clicking each one and selecting **Edit Top 200 Rows**. When the table editor opens inside the main pane, we will enter the following data into the WorkerList table:

| Worker Name | Crew Leader |
|----------------|----------------|
| David Harrison | Brad Williams |
| John Blake | Scott Copeland |
| Luke Edwards | Brad Williams |
| Neil Smith | David Fong |

It's important to note that there isn't a need to explicitly save our work like we did when designing the tables. Data is saved the moment we move out of the row we are editing.

Next, we will enter the following data into the **ProjectList** table:

| Project Name | Project Number | Status |
|-------------------|----------------|--------|
| Greenland Heights | 1023 | Closed |
| Lakeview Blvd | 1024 | Open |
| Borne Residence | 1025 | Open |
| University Gym | 1026 | Open |

You may have noticed that we have a **Status** field on our **ProjectList** table, but not in our form. We will remedy that towards the end of the chapter when we start adding some advanced features. For now, we are done with our database tables. Let's go back to our form.

Adding a new data source

Now that we have our lookup sources ready, let's utilize them in Laserfiche Forms.

The first thing we need to do is tell Laserfiche forms about the new database we created. This is done from the Administration section of Laserfiche Forms, which we can get to from the menu on the top right corner.

Once in the Administration page, we will switch to the Data Sources section

using its link on the left side. Then we will click on the **New Data Source** button on the top right corner.

We will use the following values:

```
Name: Acme Tables
DBMS Type: Microsoft SQL Server
Server: localhost\sqlexpress
Database: AcmeTables
```

For the account, you can use the same System Manager account you used in the Web Access Management page when adding new repository user accounts. Here is my setup:

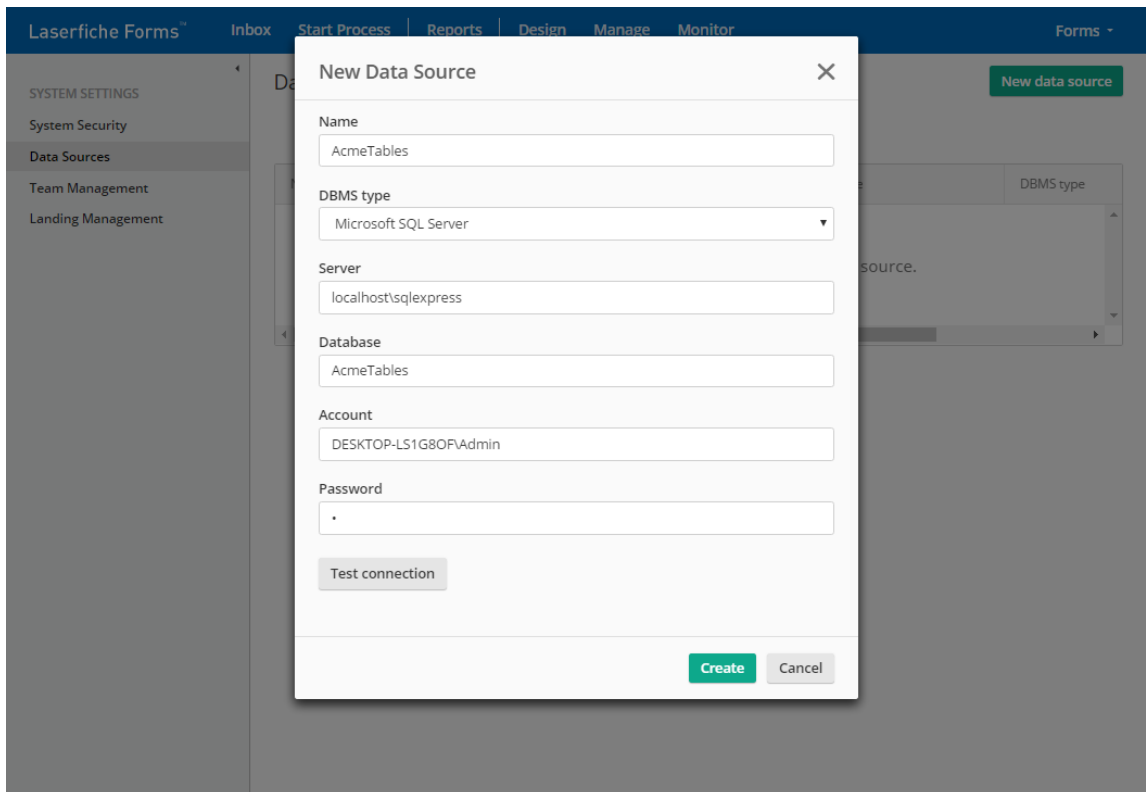


Figure 4.8: Adding a new data source to Laserfiche Forms.

When you click the **Test connection** button, you should get a success

message. However, here are a few error messages people commonly get at this step:

- **LFF2400 - DataSourceconnectionError:** This usually means you entered the data source details, such as the Server or the Database, incorrectly.
- **LFF3 - IncorrectPassword:** This means you entered the wrong account name or password.

Once the data source is successfully configured, we need to associate it with our Business Process. We can do that by clicking the newly added data source, then clicking Add/remove processes:

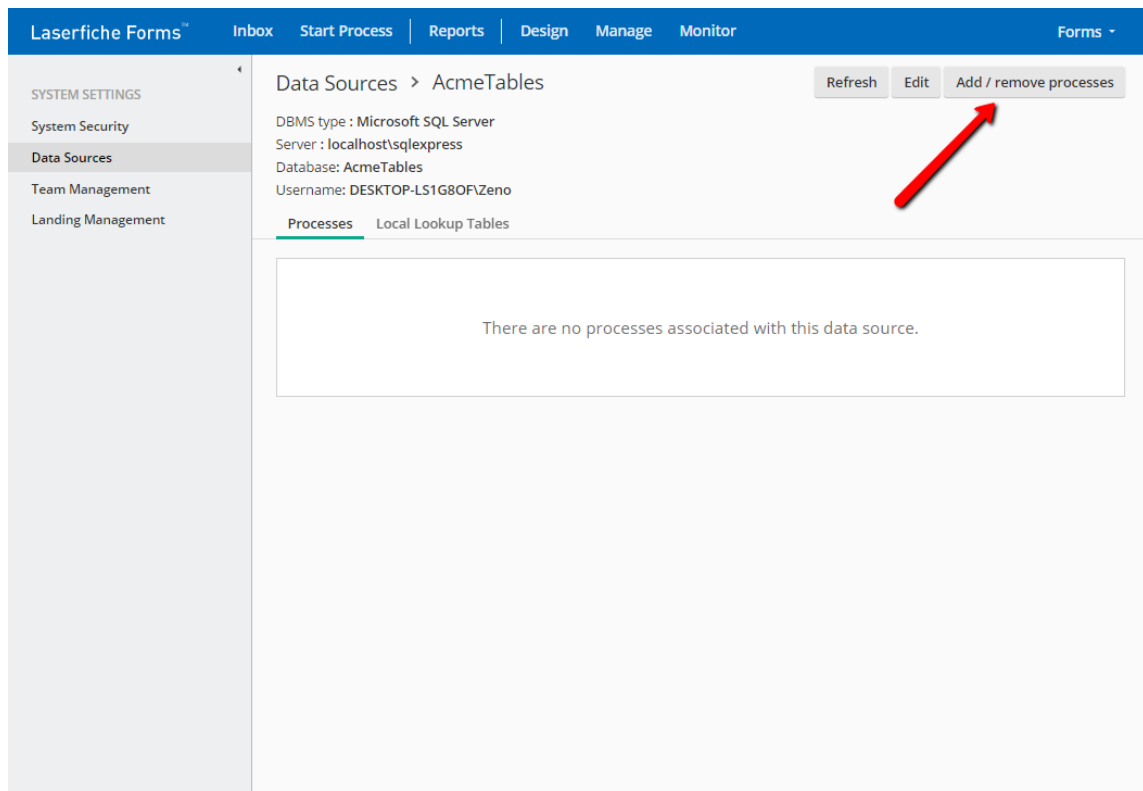


Figure 4.9: Associating our new data source with the Purchase Order Business Process.

The reason we have to do this is security. We are logged into Laserfiche Forms with the Forms account, which is a System Administrator, meaning it can modify settings in the Administration section of the system. Most users will not have this privilege, but they may still have permissions to create Business Processes and configure lookup rules in them. This step allows us to control who has access to which data source.

Configuring lookup rules

Once we associate our Business Process with the new data source, we can go back to our Purchase Request Form and switch to the **Lookup Rules** tab. Adding Lookup Rules is similar to adding Field Rules. The main difference is that we first need to select a data source, which we will do by picking **AcmeTables**.

Let's start by configuring the crew leader lookup. First, select the **WorkerList** table. Then, configure the rule like this:

```
When EMPLOYEE NAME matches with data source column WORKER NAME  
Fill CREW LEADER with data source value CREW LEADER
```

This is a very basic lookup rule that will automatically select the employee's crew leader from the **Crew Leader** dropdown when there is a matching value. There won't be one when the form launches at first, since our user name is "Forms" and we don't have a row for it in the database, but if we change it to "David Harrison" the **Crew Leader** field should get populated with "Brad Williams".

But hold on... something seems off. If you configured everything the way I did, when the form launches, the **Crew Leader** dropdown will be empty. And when you type "David Harrison" as the employee name, the dropdown will be populated with the value **Brad Williams**, but it will be the only value in the list. What happened to the other two values?

The reason this happens is that Laserfiche Forms by default overrides all the values in dropdown fields when it performs the lookup. The reasons are complicated, but suffice it to say, this behavior is not ideal in our scenario for

two reasons. First, what if a brand new worker needs to fill out a Purchase Request form but their name is not yet in the database? The **Crew Leader** field is required, but it would be empty. Second, our workers should be able to change the **Crew Leader** field because occasionally they may be working under a different crew leader than the one they are assigned to (to fill in for sick or injured coworkers).

We need to tweak our form design. First we will add a Single-line field underneath **Employee Name** and make it our new (required) **Crew Leader** field. Then we will change the label of the **Crew Leader** dropdown field to “Temp Crew Leader” and uncheck the **Required** option. Lastly, we will go to the **Lookup Rules** tab and configure our rules like this:

```
1 When EMPLOYEE NAME matches with data source column WORKER NAME
  Fill CREW LEADER with data source value CREW LEADER

2 Fill TEMP CREW LEADER with data source value CREW LEADER
```

Now, when we preview our form, everything works well. Our new **Crew Leader** field, which is now a Single-line field, gets populated when the **Employee Name** field’s value has a match. New employees whose names are not yet in the database can still manually type in the names of their crew leaders. And if a worker is working under a different crew leader that day, they can select that from the **Temp Crew Leader** field, the values for which are coming from the **WorkerList** database table. Everyone is happy and the crowd applauds!

| | |
|---------------------|----------------------------------|
| Date | Date captured on form submission |
| Employee Name* | David Harrison |
| Crew Leader* | Brad Williams |
| Temp Crew Leader | Scott Copeland |
| Summary of request* | Need some more concrete. |

Figure 4.10: Finalized crew leader fields.

Box 4.6. Green fields

You may have noticed in the above screenshot, as well as on your own form previews, that some fields turn green when values are entered into them. This is how Laserfiche Forms communicates that a Required field's input is valid. If it was invalid, the color would be red. Fields that are not required don't change their color and remain white.

Let's now configure the lookup rules for the **Project Name** and **Project Number**. We are going to approach this slightly differently. Instead of expecting our workers to have memorized every single project name (we only have four, but in a real system there may be many more), we will turn the Project Name field, which is a Single-line field, into a pseudo-dropdown field. What do I mean by that? It's actually pretty straightforward.

We will switch to the Lookup Rules tab and add our rules like this:

```

3 Fill MATERIAL.>PROJECT NAME with data source value PROJECT NAME
4 When MATERIAL.>PROJECT NAME matches with data source column PROJECT NAME
  Fill MATERIAL.>PROJECT NUMBER with data source value PROJECT NUMBER

```

When we save our new rules and preview the form, the **Project Name** field should show us a list of projects. Pretty neat!

While we are at it, let's also add another Single-line field to our **Materials** collection and name it "Project Status", then add it to the fourth lookup rule so that it gets looked up when the **Project Name** is entered.

We are done with the Purchase Request Form for now. It's time to think about what happens when it is submitted.

4.2 Designing the Process Diagram

When a Purchase Request Form is submitted, it will go to the submitter's crew leader. This will be determined by the values in the **Crew Leader** and **Temp Crew Leader** fields. If the **Temp Crew Leader** field has a value, the form will go to them. Otherwise, it will go to the regular crew leader.

The crew leader will have two choices. They will be able to approve the request, in which case we might configure the process to do something like store the form in the LFTraining repository (we won't do that in this chapter, but you are welcome to add it to your process at the end). They will also be able to reject the request and send it back to the submitter, who will then be able to make the requested changes and send it back to the crew leader. It will be possible to repeat this back-and-forth loop as many times as needed until the request is approved.

4.2.1 Purchase Approval Task

In the Process Diagram, let's start by double-clicking the **Approval** User Task. This User Task was added for us when we created the Business Process using the **Form Approval** template. If it's not there, you can add it by dragging and dropping it from the toolbox on the left side.

We see that the Purchase Request form is assigned to this User Task, so when the approver opens the task they will see the submitted form. We also see that they have two choices (called "actions") by default: Approve and Reject.

We can change the wording of these - for example, instead of “Reject” we can use the term “Send back to submitter”. Let’s go ahead and do that now.

The last thing we will do in this tab is define a **Due Date**. It’s a good idea to do this in Business Processes that are time-sensitive. Specifically, we should make sure Purchase Requests are approved (or denied) on a timely basis, because projects that run short on materials may fall behind schedule. We will change the **Due Date** option to ‘Based on a variable’, and configure it to be 2 days after Task Start Time on 05:00 PM.

General Assign Outflow Documentation

Name

Approval

Form

Purchase Request Edit

Make form read-only for users the task is assigned to

Allow direct approval ?

Allow the task to be reassigned ?

Automatically load the next task if the same person is assigned to it ?

Save form locally on Laserfiche app devices

Show Save Draft button when viewing the task

Action Buttons ?

Submit Submit

Approve Approve

Reject Send back to su

Due Date

Based on a variable ▼

2 days ▼ after ▼ Task Start Time ▼ 05:00 PM CDT

Exclude weekends

Exclude configured time when variable has time value

Priority

No priority conditions are set. Priority will be set to No Priority.

Add Condition

Figure 4.11: Approval User Task general tab.

Next, we should specify who the task will be assigned to, which we can do by switching to the **Assign** tab of the User Task. User Tasks can be assigned to individual users or teams. For now, let's select the **Users** option, then use the variable menu to pick the **Crew_Leader_1** variable.

Box 4.7. Field labels vs. field variables

There is also a **Crew_Leader** variable in the list, which is for the **Temp Crew Leader** dropdown field. You may be wondering why the variable name does not match the field label. This is because variable names are based on the initial field label at the time the field is created. They do not change when the field label changes. This is so that backend systems, such as the Process Diagram or any workflows that may be referring to the field variable, continue to operate without needing to be reconfigured. It is possible to change the variable names manually, but for this Business Process we will keep them the way they are.

We should also send an email notification to the crew leader when this User Task arrives in their Laserfiche Forms Inbox. We will enable that option and use the following for our email:

```
Subject: A new Purchase Request has been submitted
```

```
*Check "attach form"*
```

```
Body:
```

```
Greetings {/dataset/Crew_Leader_1}!
```

```
A new Purchase Request has been submitted by {/dataset/_initiator_displayname}. You  
can refer to the details by double-clicking the attachment. Please take action  
within the next 48 hours.
```

It's important to mention that Laserfiche Forms assigns tasks to users based on the user name or team name that is specified in the associated field on the User Task. Since we used a field variable, we need to make sure that the values coming from that field match the user/team names of the assignees. If Laserfiche Forms can't find a matching user or team, it will suspend the task and record a warning.

If our user accounts have different usernames, such as the person's first initial plus their last name, we need to account for that when designing the process. One way we can do this is to use two fields: a displayed **Crew Leader** field that contains the crew leader's full name, as well as hidden a **Crew Leader User Name** field that would be populated with their user name in Laserfiche Forms. We would need to add that to our database table, or create a two-staged lookup. And then we would use the hidden field's variable in the User Task to assign the task to that user. At the end of the day, regardless of which approach we take, dynamic form routing (i.e. based on field variables) can add a certain amount of complexity to Business Processes that you have to plan for in advance.

We are now done configuring the Approval User Task. If we look at the Process Diagram, we see that both approved and rejected forms result in the process ending right away. This isn't very helpful - we want rejected Purchase Requests to go back to the submitter (the worker) so that they can make the necessary changes. How do we accomplish this?

4.2.2 Submitter Revision Task

Your first instinct might be to direct the Rejected arrow back to the green **Message Start Event**. That would not work though. Start Events simply start the process instance. They don't play any roles after that. In order to give the original form submitter a chance to make edits to their forms, we need to use a second User Task. Let's go ahead and insert that into our Process Diagram, then connect the Rejected arrow to it and double-click it to start making edits.

Whenever we have more than a few items in the Process Diagram, it's a good idea to make sure they have descriptive names. We will name this User Task "Submitter Revision". Then we will select the Purchase Request form. We also need to uncheck the option that says **Make form read-only for users the task is assigned to** because submitters need to be able to make revisions. It is, after all, their form!

Then we need to enable at least one **Action Button**. Which one(s) should we enable? It depends on what we are trying to accomplish. Should the workers

only be able to send the form back to their crew leader? Or should they also be able to discard it? In our scenario, the latter makes more sense. Maybe the materials being requested are already on the way the construction site, and there is no need to purchase more. So once the crew leader sends the form back to the worker, the worker can discard it. (Alternatively, we could give the crew leader the ability to do this. Since this is an imaginary scenario, there is no one right way to do it.)

We will enable the Submit and Reject buttons. We will change the first label to “Resubmit to crew leader” and the second one to “Discard”. We also need to assign the task to the submitter, which we can do by switching to the **Assign** tab, picking the **Users** option and selecting “Initiator” from the dropdown menu.

Let’s also enable the email notification for this User Task and configure it as such:

```
Subject: Your Purchase Request has been rejected
Body:

Hello {/dataset/_initiator_displayname},

Your crew leader rejected your Purchase Order request. Below is their comment:

{/dataset/_comment}

You can make the requested changes and send it back to them, or discard the request.
```

Each User Task enables the user performing that task to enter comments into a built-in **Comments** field. We will see that when we test our Business Process, but for now let’s refer to it in our email so we don’t have to come back and insert it later.

If we try to validate our Process Diagram now, we will get an error message saying the **Submitter Revision** User Task we added is not connected to an end event or other objects. Let’s go ahead and do that now. We will select the task and drag the tiny arrow on its top right corner to the Approval task. This will be the “Resubmit for approval” outflow path. We will then double-click the arrow and check the **This is the default outflow path** option. What

this means is that if none of the other outflow path conditions are met, the process will go down this one.

We also need to define the **Discard** path, which will end the process. We can either drag an **End Event** from the toolbox on the left side of the Process Diagram, or select our User Task and dragging the first red circle onto the diagram. Then we will click its arrow and define its conditional expression. (We can't define this as the default outflow paths because we already did that for the path connecting back to the Approval task.)

We already have some experience with conditional expressions from [Chapter 3](#). Let's define this expression as:

```
Last User Action = Discard
```

We will also change the name of this path to “Discard” and click Done.

Box 4.8. Conditional expressions and button labels

You may have noticed we defined the conditional expression for the Discard branch as **Last User Action = Discard**. This is because the label of the corresponding button on the **Submitter Revision** User Task is **Discard**. This is an important aspect of conditional expressions: whenever you change Action Button labels in User Tasks, you need to make sure the outflow paths for those buttons have matching conditional expressions.

Now when we validate the Process Diagram, we should see a green success message.

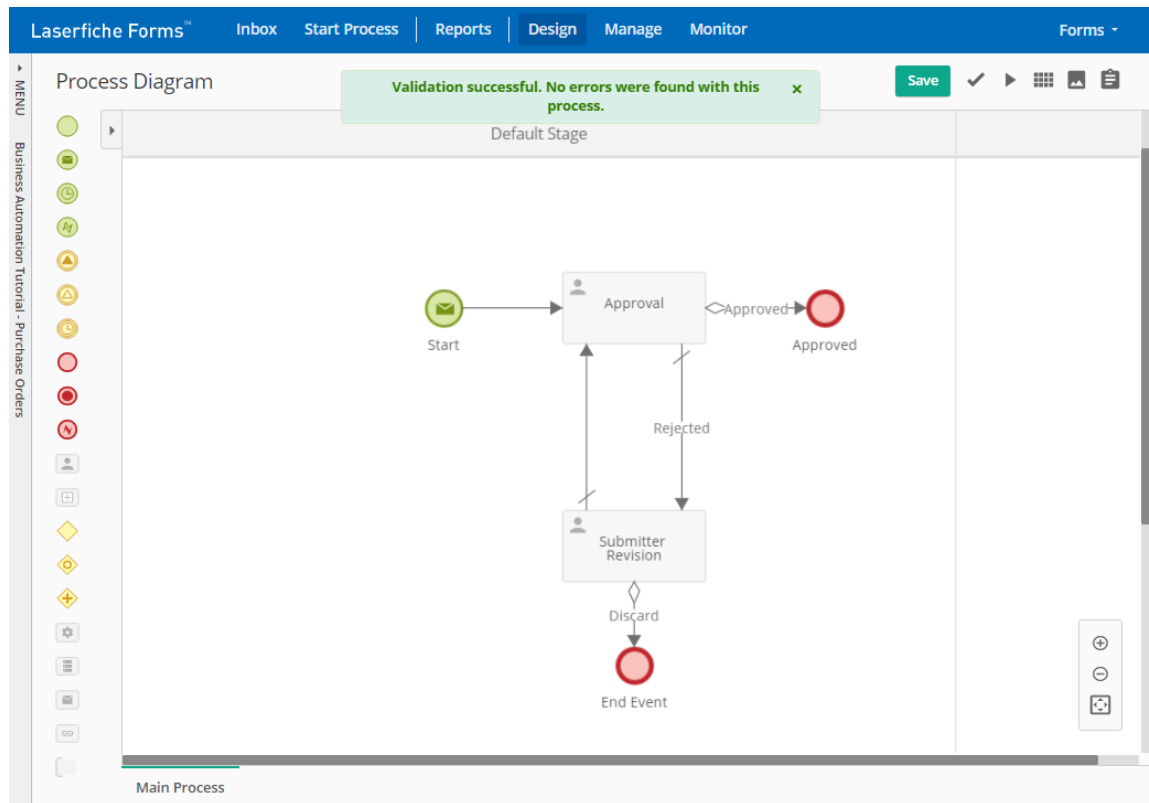


Figure 4.12: Process Diagram with two User Tasks.

We are almost ready to test the Business Process with our “real” users, David Harrison and Brad Williams. Before we can do that though, there are a few other things we need to take care of.

4.3 User accounts and security

4.3.1 User accounts

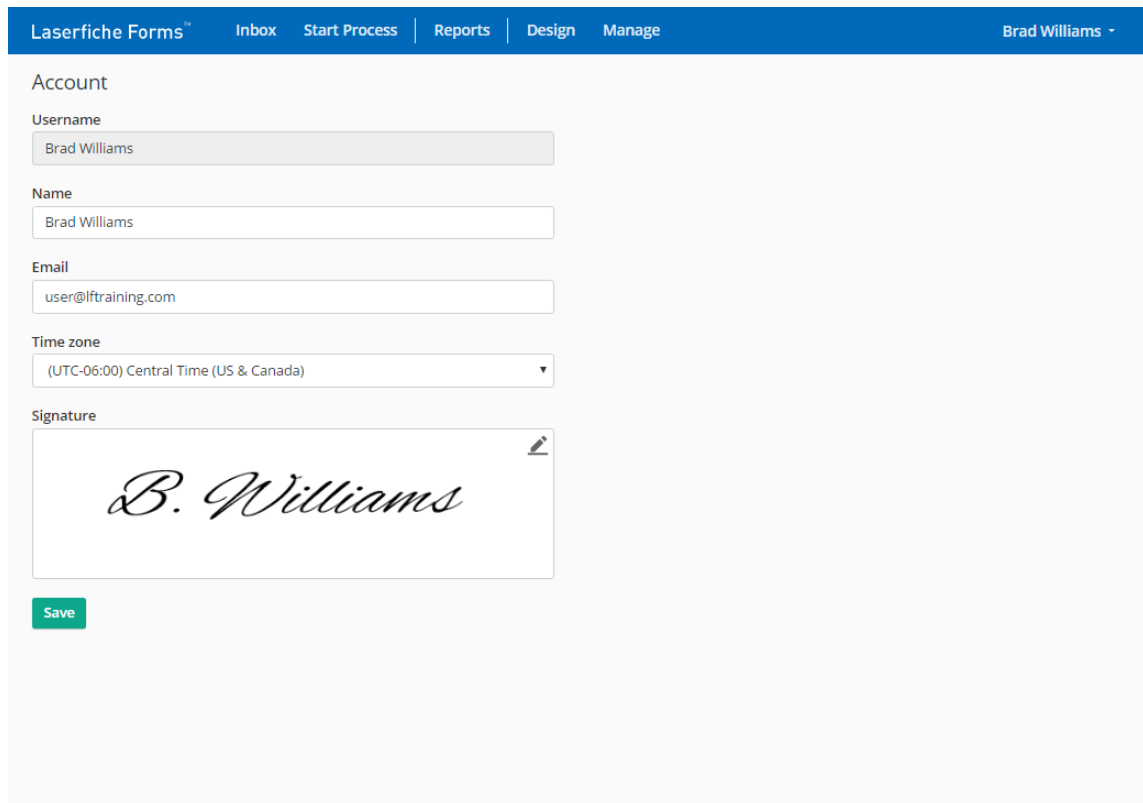
We have two User Tasks in our Business Process: the **Approval** task, which is assigned to the crew leader selected on the form, and the **Submitter Revision** task, which is assigned to the initial submitter. Each has email notifications configured because we want those users to be notified when a task gets as-

signed to them. The question is, how does Laserfiche Forms know where to send the email notification?

It's actually not magic. Every user account has several pieces of information associated with it, one of them being Email Address. Let's sign out of Laserfiche Forms now (we are logged in as the system administrator user, which is called Forms) and sign back in as David Harrison. Then, from the top right corner menu, click **Account**.

We should see David's account page. His Username and Name are populated, but his Email is blank. Let's fill that in with "user@lfttraining.com". Even though we have multiple user accounts in the system, we will have them share the same email address for simplicity. We will then sign off, sign back in as Brad Williams, and do the same thing.

You will also notice there is a Signature option. In Laserfiche Forms, each user account can have a default signature defined. This allows users to quickly place that on Signature fields on forms, as opposed to having to type or draw their signature from scratch every single time. We will go ahead and add a signature for Brad as well.



The screenshot shows the user account page for Brad Williams in the Laserfiche Forms application. The page has a blue header with the following navigation items: "Laserfiche Forms™", "Inbox", "Start Process", "Reports", "Design", and "Manage". The user's name "Brad Williams" is displayed in the top right corner. The main content area is titled "Account" and contains the following fields:

- Username:** A text input field containing "Brad Williams".
- Name:** A text input field containing "Brad Williams".
- Email:** A text input field containing "user@lfrtraining.com".
- Time zone:** A dropdown menu currently set to "(UTC-06:00) Central Time (US & Canada)".
- Signature:** A large text area containing a handwritten signature "B. Williams" in cursive. A small edit icon is visible in the top right corner of the signature area.

At the bottom left of the form, there is a green "Save" button.

Figure 4.13: User account page for Brad Williams.

Because we entered an Email address for our users, Laserfiche Forms will now be able to send us email notifications based on our user account.

Box 4.9. Forms accounts with Laserfiche Directory Server

If we were using Laserfiche Directory Server for user authentication (recall the configuration from [Section 2.1.4](#)), then we could configure each user's name, email and signature there. Once synchronized, Laserfiche Forms would pick it up automatically, eliminating the need to do what we just did manually. It is one of the many benefits of using Laserfiche Directory Server.

4.3.2 System Security

When configuring David and Brad's user accounts, you may have noticed that the top navigation bar contains **Reports**, **Design** and **Manage** links. We may not want to make those features available to David and Brad though, because they will be regular users and should not be able to create and manage processes. Let's fix this.

We will sign off as Brad and sign back in as the Forms user. Then, from the top right corner menu, we will go to the Administration page. On the Named User list, we can see that both David and Brad have their account roles set as **Inherit From Group**. In this particular case, the inherited role is the **Process Creator** role. Let's change that to **Basic User** for both accounts. Doing this won't remove the navigation links for David and Brad, but they will not be able to actually Design or Manage processes or access reports unless given explicit permission to them, which is what we want. For example, we may want Brad to be able to run reports on how many Purchase Requests each of his crew members submit every month. With this security configuration, we will be able to give his account that permission.

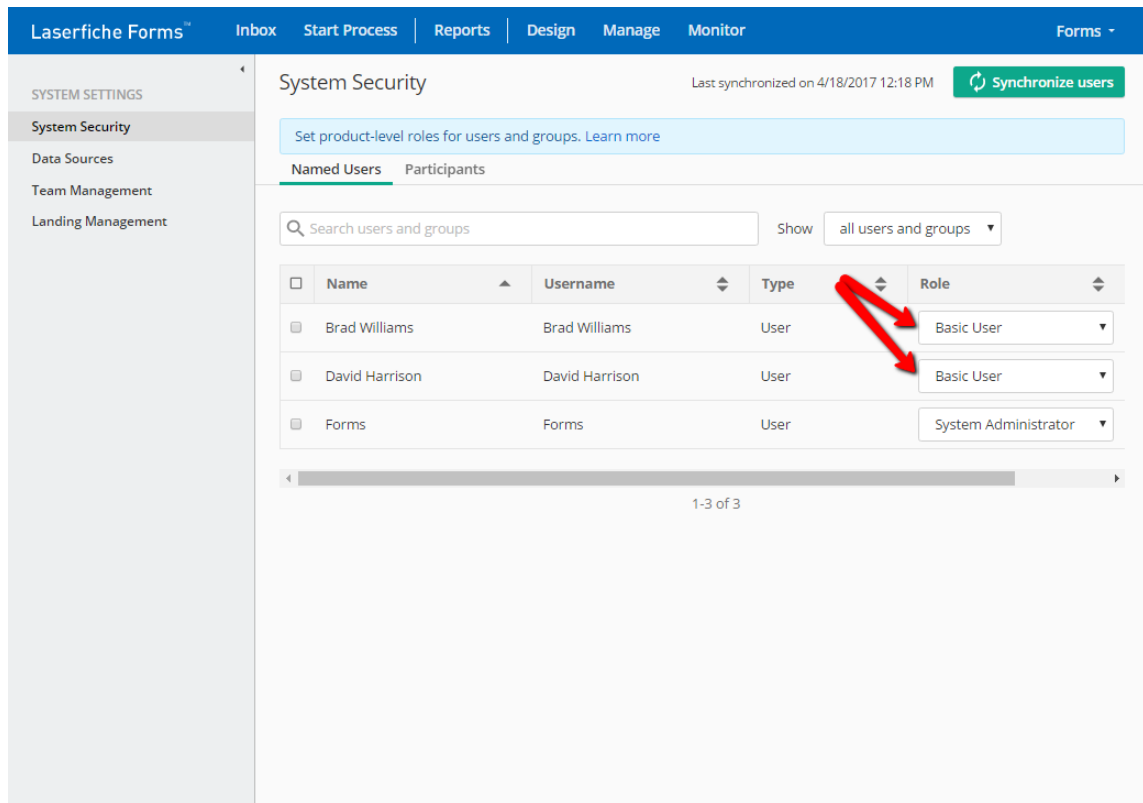


Figure 4.14: Changing David and Brad’s roles to Basic User.

4.3.3 Business Process access rights

The security settings we configured above were system-wide settings. We changed David and Brad’s roles to **Basic User**, which means by default they will have access to only the Business Processes we specify. We will give both of them access to the Purchase Order Business Process now. To do this, we will go to the Process Diagram and click the **Access Rights** link on the left-most pane.

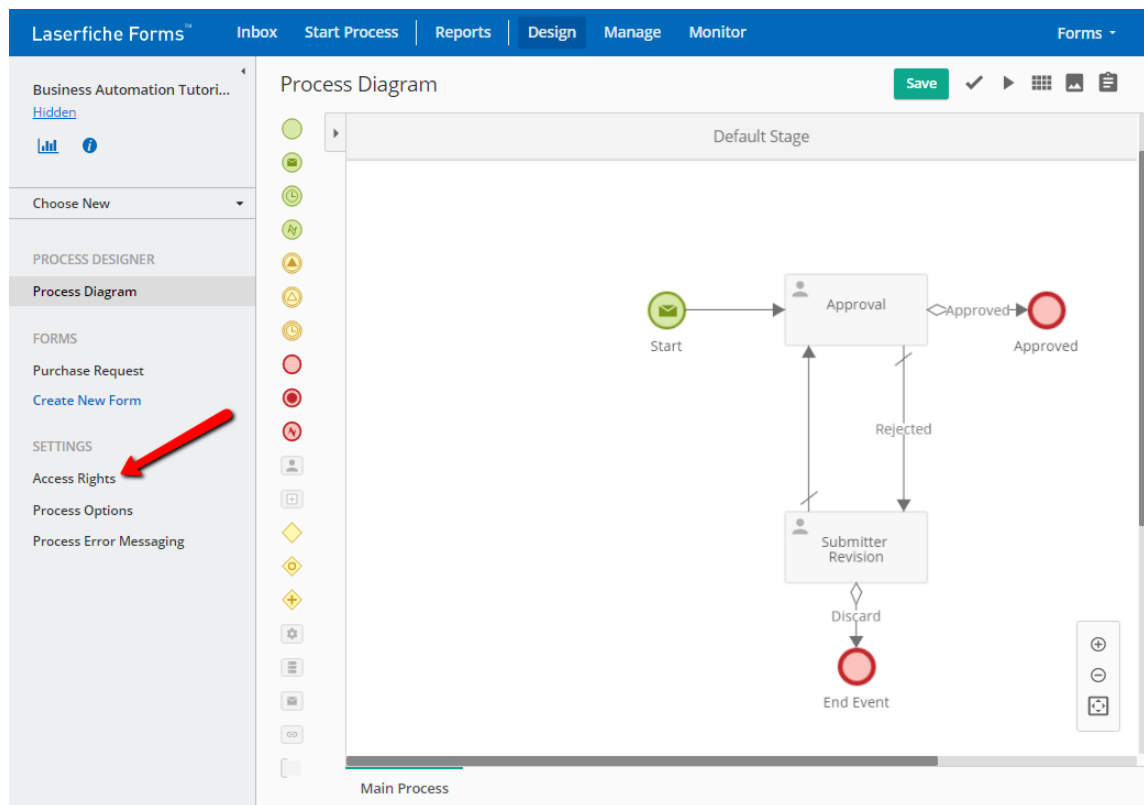


Figure 4.15: Getting to the Access Rights section of the Business Process.

The Access Rights currently contains only our Forms user, which is specified as the Process Admin. This essentially gives the user full control over the Business Process. Keep in mind that the user who creates the Business Process is given the Process Admin role over that process, but their role can be reduced by other System Administrators.

Let's add David Harrison and Brad Williams to the list. They will be added with the Submitter role, which is what we want: they will be able to submit forms and perform User Tasks, but nothing else.

4.4 Publishing and testing

The last thing we need to do before we can test the Business Process as David and Brad is publish it. We can do this from the **Process Options** page, which can be accessed from the left-most pane similar to the Access Rights page.

4.4.1 Publishing the Business Process

Newly created Business Processes are hidden by default. This allows Process Admins to create forms, design the Process Diagrams and make all other necessary changes before releasing them. Let's do that now by clicking the green **Publish** button.

Each Business Process is also given a unique link through which it can be started. The last part of the URL is always randomized, but it's a good idea to change that to something that is easy to distinguish. We will change it to "http://localhost/Forms/PurchaseRequest".

Next, we will configure the Instance Name. This determines the subtitle of the User Tasks that will appear in the Laserfiche Forms Inbox for each user. It is a good idea to make the instance names contain easily identifiable pieces of information from the process or the forms in it. We will look at why this is important when we test our process.

For now, let's go ahead and change the instance name:

```
Purchase Request - {/dataset/Employee_name} - ${/dataset/Total}
```

It's also worth mentioning that Laserfiche Forms can send email notifications if a process instance terminates. This is an advanced option we won't use right now, but for complex Business Processes that contain lots of moving parts (such as workflows), it's a good idea to enable this option so that the designated administrators can troubleshoot issues proactively.

4.4.2 Testing the Business Process

We are finally ready to test our Purchase Requests!

Let's sign in as David Williams, then click the **Start Process**. If everything was properly configured, we should see the **Business Automation Tutorial - Purchase Orders** process on the list, and it should have a **Start** button next to it. Let's click that and launch our form.

When the form launches, the **Employee Name** field should say **David Harrison**, and the **Crew Leader** field should automatically look up the crew leader, which is **Brad Williams**. We will keep the crew leader as Brad, pick "Greenland Heights" as the project, fill out the rest of the fields and submit the form.

Two things should happen. First, we should get an email notification addressed to Brad. The email should contain the PDF attachment of the submitted form, as well as a link to the Approval User Task. Second, if we log into Laserfiche Forms as brad, in our Inbox we should see the Approval User Task listed:

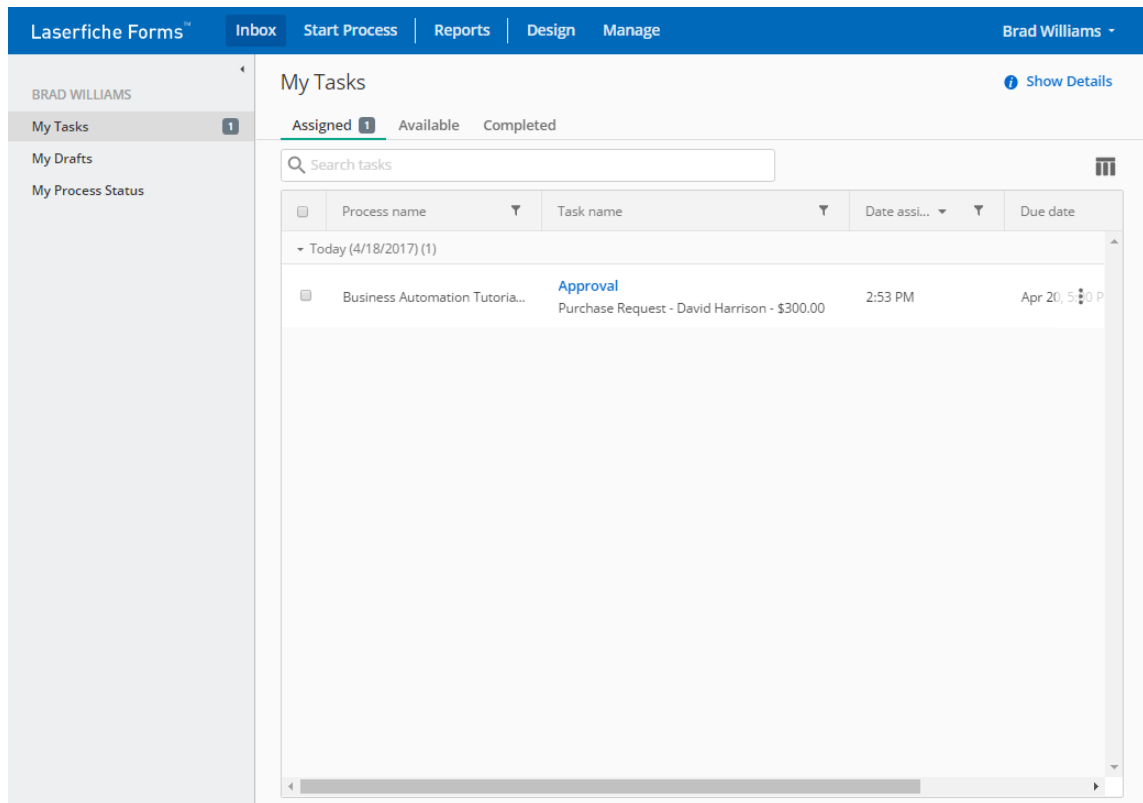


Figure 4.16: Brad’s Inbox in Laserfiche Forms.

The User Task can be opened both from the Laserfiche Forms **Inbox** and from the link in the email notification. Laserfiche Forms provides this flexibility because people have different work patterns: those who sit in front of their computer throughout the day can have the Laserfiche Forms Inbox open (and see the tasks appear as they arrive), while those in the field can rely on email notifications on their smartphones.

Performing the approval task

When we open the User Task, it will open in its own browser window. Unlike the initial form submission, this window will contain additional information. Here is what mine looks like:

The screenshot shows a web-based approval interface. At the top, it displays 'Approval' for a 'Purchase Request - David Harrison - \$300.00'. It is assigned to 'Brad Williams' with a 'Date assigned: 3:20 PM' and a 'Due date: Apr 20, 5:00 PM'. The priority is 'None'. A 'Save draft' button is visible in the top right.

The main form area is titled 'Purchase Request Form' and includes the ACME Industries logo with the slogan 'It Might Just Work!'. The form contains the following fields:

| | |
|---------------------|-------------------|
| Date | 4/18/2017 |
| Employee Name* | David Harrison |
| Crew Leader* | Brad Williams |
| Temp Crew Leader | Scott Copeland |
| Summary of request* | Need more hammers |

The 'Materials' section includes:

| | |
|-----------------|-------------------|
| Project Name* | Greenland Heights |
| Project Number* | 1023 |
| Item* | Hammers |
| Category* | Hand Tools |
| Cost | \$ 300.00 |
| Total | \$ 300.00 |

There is a 'Comments' text area and two buttons at the bottom: 'Approve' and 'Send back to submitter'.

The right-hand pane shows the 'Action History' for this approval. It includes a 'More details' toggle and a timeline of events:

- Approval** (Viewing): assigned to Brad Williams on 4/18/2017 3:20 PM. Status: In progress.
- Start**: completed by David Harrison on 4/18/2017 3:20 PM. Status: Submitted.

Figure 4.17: Brad's approval task.

The top pane contains the process summary, such as who the task is assigned to and the date on which it was assigned. It also has a **Save draft** button, which allows the approver to save their work and come back to it later. This may be beneficial if the approver is expected to perform a lot of activities during the approval task, such as look up information in external systems and fill out a series of additional fields.

The right-hand pane contains the Action History, which lists the steps the process instance has gone through up to that point. This is particularly useful in more complex processes where the approver needs to be able to see who has

performed the previous steps and when. In our case, it will only have the Start event, which we can see was completed by David Harrison a few minutes ago.

Finally, you will notice that the form fields are not editable. This is because we left the **Make form read-only for users the task is assigned to** option enabled when configuring the User Task (Figure 4.11). Depending on the Business Process and the step, we may want to give users the ability to make edits on the form. In this scenario though, the crew leaders are pretty busy, and if edits need to be made they will simply send the form back to the submitter.

That is what we will do now. Let's first enter something into the **Comments** field. Remember that this field is built into every User Task, and allows users to more easily collaborate. We will enter the following comment:

`This project is closed, David.`

Then we will send the form back to David, who should receive an email notification containing the comment from Brad.

4.5 Optimizing the Business Process

Our Business Process works, but there are a few things about it that we can optimize.

1. In the test scenario we just ran, Brad rejected the purchase request because it was for a closed project. It would be better if we had a way of preventing the submission of purchase requests for closed projects.
2. The Approval User Task is assigned to the crew leader, but what if the worker has a temporary crew leader that day? In those cases, the task should probably be assigned to that crew leader instead of the main one.

Let's make these changes now.

4.5.1 Preventing purchase requests for closed projects

If you recall from [Section 4.1.2](#), when we created our ProjectList database table, we also added a **Project Status** column. We haven't utilized it in our form thus far, but we will now.

Let's open the form's layout and insert a new Single-line field labeled "Project Status". Make this field Required and Read-only. Then, switch to the **Lookup Rules** tab and add a new line item to the fourth lookup rule so that it also populates the **Project Status** field. When we save the rule and preview the form, the **Project Status** field should now be populated when the project is picked from the **Project Name** list.

Remember though, we don't want to simply communicate the project status. We also want to prevent the form submission if it is a closed project. How do we do this?

Error Messaging

Laserfiche Forms 10.2 has a new feature that allows customizing error messaging on forms. We will utilize this feature now by switching to the **Error Messaging** tab of the form.

We will configure our first error messaging rule like this:

```
Display: Purchase requests cannot be made on closed projects.  
When: Regular expression does not match.  
For: These individual field(s)  
Materials > Project Status
```

Let's now define a regular expression for the **Project Status** field. Save the rule we just created and go back to **Layout**. Then, edit the field and switch to the **Advanced** tab, and enter "Open" as the **Regular expression for validation** option field.

Now when we log in as David Harrison and try to submit a Purchase Request for Greenland Heights, we will see our error message appear underneath the **Project Status** field, and the submission will be prevented.

The screenshot shows a web form titled "Materials" with the following fields and values:

- Project Name*: Greenland Heights
- Project Number*: 1023
- Project Status*: Closed
- Item*: Screwdrivers
- Category*: Hand Tools
- Cost*: \$ 200.00

A yellow highlight covers the Project Status field and the error message: "Purchase requests cannot be made on closed projects." Below the form is a "Total" field showing \$ 200.00 and a "Submit" button.

Figure 4.18: Submission will be prevented if the project is closed.

4.5.2 Generating a unique purchase request number for each form

In most organizations, purchase orders are given a unique number. This allows the accounting staff to associate them with invoices that arrive later for that purchase. Currently, our form has no such number. In this section, we will use a SQL stored procedure for it and configure our form to invoke it.

Box 4.10. Instance IDs

It is worth noting that Laserfiche Forms already generates a unique number for each process instance, called the **Instance ID**. However, the **Instance ID**

counter is shared across all Business Processes. This means that if we have multiple Business Processes in production, submitting a Customer Survey form would increment the counter.

In addition, an **Instance ID** is generated only when the instance starts, which, in the case of our Purchase Requests, happens when the initial form is submitted. That means the initial submitter does not see the **Instance ID**, unless we specifically email it to them using an Email Service Task.

It is for these reasons that using a stored procedure may be preferable. It depends on your use case. As a rule of thumb: if all you need is a unique number to assign to the form, use the **Instance ID**. But if you want more control over the unique numbers, use a stored procedure.

Okay, let's launch SQL Management Studio and log into our SQL server. We will go to the Programmability section of our AcmeTables database, then right-click Sequences and create a new sequence object.

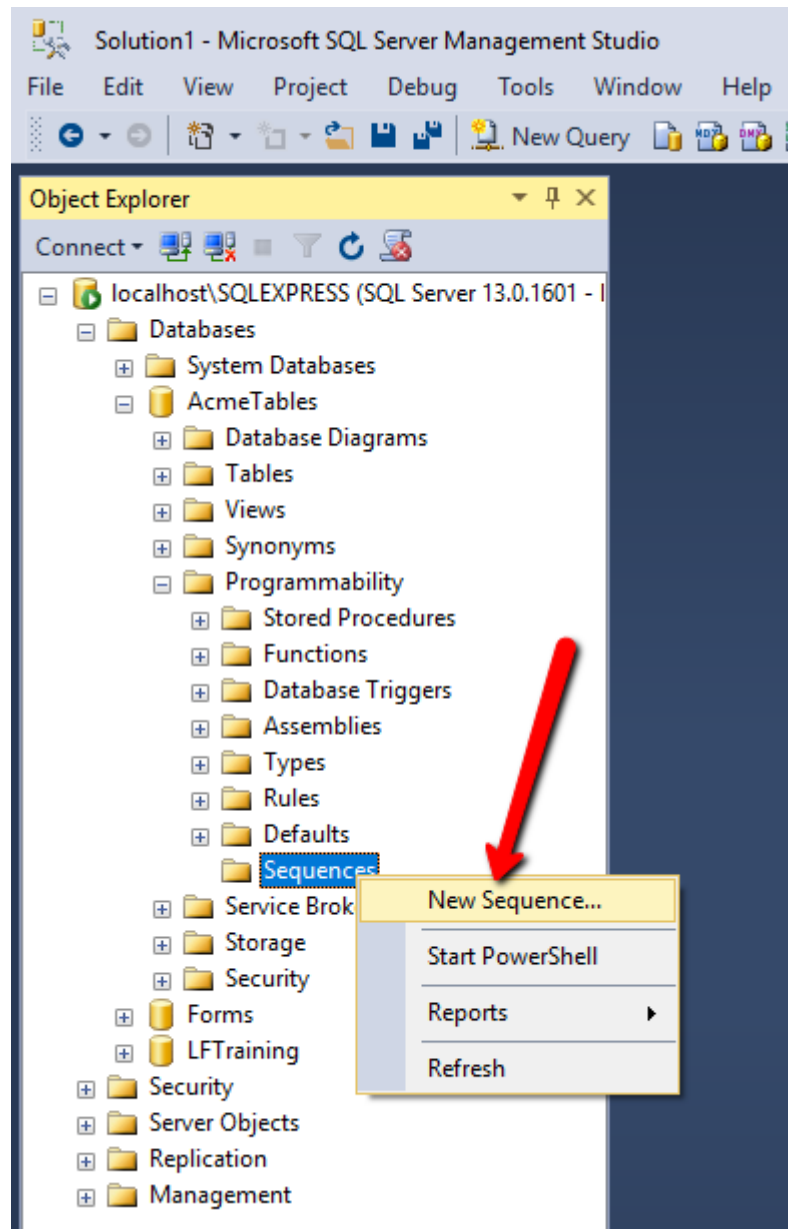


Figure 4.19: Creating a new sequence object.

Box 4.11. Sequence objects require SQL 2012 or later

Sequence objects are only available on Microsoft SQL Server 2012 or later. If you are using an earlier version, you will not be able to complete this section of the tutorial.

We will name our sequence “POSequence”, set the start value at 1000 and have it increment by 1. We could also set a minimum and maximum and have it cycle when it reaches the maximum, but we won’t do that. We will, however, set the **Cache option** to “No cache”.

Sequence objects are internal to SQL Server. They cannot be directly invoked by outside programs. So we need to use a stored procedure to get values from the sequence and pass them to Laserfiche Forms. Let’s right-click the Stored Procedures node and create a new stored procedure.

The query editor will open with a bunch of boilerplate code in it. Go ahead and delete it all and replace it with the following:

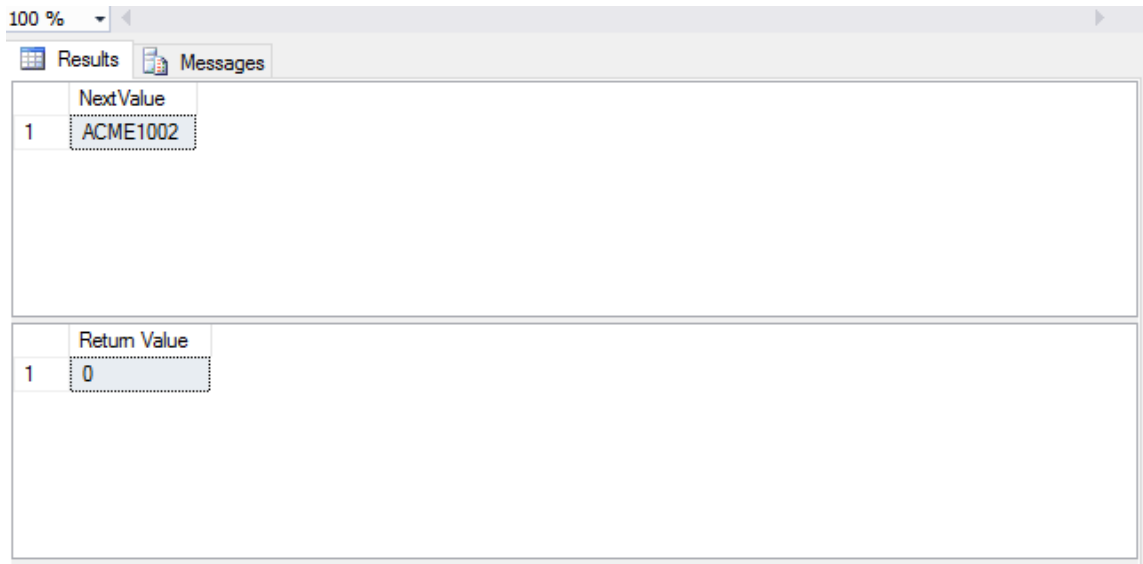
```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE dbo.GetNextPONumber
AS
BEGIN
    SELECT 'ACME' + Cast((NEXT VALUE FOR POSequence) as nvarchar(50)) AS 'NextValue'
END
GO
```

When you press F5, you should see a “Command(s) completed successfully” message in the output pane.

We aren’t going to cover how to write SQL in this tutorial, but I want to briefly explain what the above code does. It basically creates a new stored procedure that grabs the next value from the PONumber sequence we created, then prepends the word “ACME” to it and makes it available under the “NextValue” alias. You will see what that means in a few minutes.

First though, let’s make sure our new stored procedure works. We can do this by first right-clicking the Stored Procedures node in the Object Explorer

and selecting Refresh, then right-clicking our stored procedure and selecting “Execute Stored Procedure”. You should see the following output (your PO number may be different):



| | NextValue |
|---|-----------|
| 1 | ACME1002 |

| | Return Value |
|---|--------------|
| 1 | 0 |

Figure 4.20: The result we get by executing our new stored procedure.

Now let’s utilize this stored procedure in Laserfiche Forms. We have already added our **AcmeTables** data source to it. However, whenever new tables, views or stored procedures are added to the data source, it needs to be refreshed so that Laserfiche Forms becomes aware of them. So let’s go to the **Administration** page in Laserfiche Forms, switch to the **Data Sources** section, open the **AcmeTables** data source and click the **Refresh** button on the top right corner. Then we will also use **Add/remove processes** and give our new Business Process access to this data source.

Then we will hook up our stored procedure to the Purchase Request form. First, let’s add a Single-line field labeled “Request Number” to the top of our form, and make it **Read-only**. Then, switch to the **Lookup Rules** tab and add the following rule:

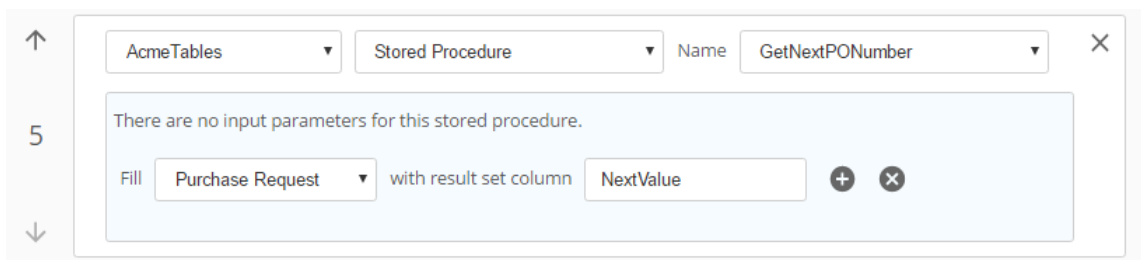


Figure 4.21: Adding a lookup rule that uses our stored procedure.

Now when we save our new rule and preview the form, the Request Number field should automatically grab the next number in the sequence:

A screenshot of a web form for a purchase request. The form has several fields: 'Request Number' with the value 'ACME1004', 'Date' with the placeholder 'Date captured on form submission', 'Employee Name*' with the value 'Forms', 'Crew Leader*' which is empty, 'Temp Crew Leader' which is empty, and 'Summary of request*' which is empty. A red arrow points to the 'Request Number' field.

Figure 4.22: Purchase request gets the next PO number.

Stored procedures are incredibly powerful, and can get the job done where simpler lookups based on tables and views fall short. They require a fair amount of SQL knowledge though, so unless you are already proficient in SQL, you may need to talk to your company's IT team to have them write the stored procedures for you. From there, you can simply connect them to Laserfiche Forms. If you are interested in learning SQL, I have provided some resources for it at the end of the book.

4.5.3 Routing the forms to the correct crew leader

Next, we will make some changes to our Process Diagram. When a Purchase Request form is submitted, it needs to be routed to the correct crew leader. Specifically, if the **Temp Crew Leader** has a value, the form should be routed to them instead of the worker's regular crew leader.

We are not going to spend too much time on this. In fact, I am going to leave this to you to perform as an exercise. You should be able to use everything we have covered so far to make this work. Good luck!

Box 4.12. Exercise hint

When testing this feature with David Harrison after making the necessary changes to the Process Diagram, you may want to change the value of the **Crew Leader** field from the looked up value "Brad Williams" to something else, and then pick "Brad Williams" from the **Temp Crew Leader** dropdown. That way, you don't have to create and configure new user accounts.

4.6 Chapter summary

We covered a lot in this chapter. Let's quickly review what we learned:

- Synchronizing Laserfiche Forms user accounts with the authentication source (Laserfiche Server or Laserfiche Directory Service)
- Configuring and utilizing database tables as lookup sources in forms
- User Tasks and actions
- Advanced forms options such as formulas and error messaging rules
- System security and Business Process Access rights

- Advanced lookups using SQL stored procedures

In [Chapter 5](#), we will create our most complex Business Process yet. In addition to the features we have covered so far, it will have multiple forms in it, and each form will utilize advanced customizations using CSS and JavaScript.

Chapter 5

Business Process: Job Applications

In this chapter, we will build on what we learned in previous chapters and build a fairly complex Business Process that will allow prospective Acme employees to apply to various positions we have open. In addition to their personal and background information, applicants will be able to submit their resumes and cover letters. When submitted, the package will first be screened by a recruiter, who will be able to send it to multiple departments for review (in case the candidate meets the criteria for multiple positions). Once each department makes their decision, the recruiter will receive a notification and be asked to contact the prospective employee for an on-site interview.

Note about this chapter

This chapter will have a bit of HTML, CSS and JavaScript. As mentioned at the beginning of this book, I don't assume any familiarity with these technologies. I'll do my best to give you a crash course on them and make sure to explain everything in detail as we go. However, if you have never done any web design and/or programming before, it is likely that you will run into error messages or unexpected behavior. My advice if that happens is this: don't give up. Laserfiche Forms by itself is quite powerful, but in my opinion its real strength is its openness to customization using these technologies. So if you learn the basics,

you will be in a really good position to make some amazing improvements to Business Processes you build.

The other thing I want to note here is that, while in previous chapters I was very explicit about what link to click on which part of the screen to get to where you need to go. I expect that that has become second-nature to you by now, so in this final chapter I will simply say stuff like “go to the Forms Administration page” rather than giving detailed instructions with screenshots.

With that out of the way, let’s begin!

5.1 Creating the job application form

Our first order of business is to create a new Business Process, which we will name “Business Automation Tutorial: Job Applications”. I picked the Form Approval template when creating mine. You don’t have to, but it saves you a few clicks, so you might as well do the same.

Let’s go to the Starting Form. The first thing I always do is edit the Form name (which, remember, is different from the Form title). Open the Form Settings dialog and change the Form name to “Job Application”. Let’s change the labels to be aligned at the top of their fields. Then, we will also change the Backend Validation option to “no validation” (which we need to do to work around a very minor bug in Laserfiche 10.2).

Our form title will be “Job Application” and it will use the following as its description:

Thank you for your interest in Acme Industries. Please fill out this form and submit it. You will hear back from us within one week.

We will also change the form’s theme to the Acme Industries theme that we have been using throughout the book. We will actually make a few tweaks to this later in this chapter, but for now we can use it the way it is.

We will start with the following fields:

| Field label | Type | Required | Notes |
|-------------|------|----------|-------|
|-------------|------|----------|-------|

| | | | |
|--|-------------|-----|------------------|
| First Name | Single-line | Yes | |
| Last Name | Single-line | Yes | |
| Email | Email | Yes | |
| Primary Phone | Single-line | Yes | |
| Current Address | Address | Yes | |
| Are you 18 or older? | Drop-down | Yes | Choices: Yes, No |
| Date of Birth | Date | Yes | |
| Are you able to lawfully work in the U.S.? | Drop-down | Yes | Choices: Yes, No |

At this point you may be thinking, “hold on, I don’t think it is legal to ask job applicants for their date of birth”, and you would be correct. We will add a Field Rule so that field is shown only if the applicant indicates they are under 18 years of age.

5.1.1 Standardizing user input

Let’s talk about phone numbers. In the United States, there are multiple phone number formats that are commonly used. Some people may enter their phone number as “512-394-5049”, while others may enter it as “(512) 394-5049”. It is also common to forgo all hyphens and parentheses and simply enter numbers.

Often times, Laserfiche Forms Business Processes are connected to back-end systems, such as a Laserfiche repository or a line of business application. These systems may impose their own constraints. As such, being able to standardize data up-front (i.e. on the form) can be very desirable, because it eliminates the need to reformat the data later on.

This is what regular expressions are used for. We briefly used them back in [Section 4.5.1](#), but the way we used them there was more of a neat trick. For our Phone Number field, we will define a “real” regular expression and specify how we want users to format their input. Let’s go to the **Advanced** tab of the field and enter the following as the **Regular expression for validation** option:

```
\d{3}-\d{3}-\d{4}
```

Here, the “\d” stands for “any digit”, and the number inside the curly brackets indicates the number of digits we expect for that group. Based on the regular expression, we are saying the phone number should be “three digits, hyphen, three digits, hyphen, four digits”. So “(512) 394-5049” would be invalid, whereas “512-394-5049” would be valid.

We should also enter some text above the field to communicate the expected format to our users. We will enter:

```
Example phone number format: 123-456-7890
```

We will also configure an error messaging rule so that if the **Phone Number** field’s value does not match the regular expression, a friendly error message gets shown. I will leave that up to you to configure. (If you need a reminder, revisit [Section 4.5.1](#))

5.1.2 Form pagination

Let’s start the next part of our form: collecting education and work history. This is one of the most important parts of a job application: we want to make sure our applicants have relevant degrees and work experience before hiring them!

We will begin by inserting a Page Break. Page Breaks split forms into multiple pages, which can be a great way to break long forms into smaller chunks and make them easier to fill out. In addition, when forms that have page breaks are stored in the Laserfiche repository, each form page is stored as a separate page, making the document easier to navigate. We will title our page “Education and Work History”. In addition, we will go to the pagination settings and change the Pagination Style to a progress bar, which is both more stylish and also fairly common in online job applications.

Now we need to populate our new form page. First, let's insert a dropdown field labeled "Highest level of education completed". This will have several choices:

- High school
- Associate's degree
- Bachelor's degree
- Master's degree
- Ph.D.

Then we are going to insert a Collection, labeled "Work History". It will contain the following fields:

| Field label | Type | Required | Notes |
|--------------------|-------------|----------|-------|
| Company Name | Single-line | Yes | |
| Job Title | Single-line | Yes | |
| From | Date | Yes | |
| To | Date | Yes | |
| Reason for Leaving | Single-line | No | |

It's also a good idea to change the collection's range to have a minimum of 0 items. This is so that applicants without work history, such as new college graduates, can submit applications. (However, if we were only interested in experienced applicants, we could leave this minimum at 1, or increase it.)

We will then add a third and final page, labeled "Position Information". This page will start with a Custom HTML field that contains some legal text:

Acme Industries is an equal opportunity employer and will not distinguish against any employee or applicant on the basis of age, color, disability, gender, national origin, race, religion, sexual orientation, veteran status, or any classification protected by federal, state and local law.

Then we will add the following fields:

| Field label | Type | Required | Read-only |
|---|-------------|----------|-----------|
| Position Desired | Drop-down | Yes | No |
| Position Status | Single-line | No | Yes |
| Salary Desired | Currency | No | No |
| When can you start? | Date | Yes | No |
| Type | Drop-down | Yes | No |
| Resume | File Upload | Yes | No |
| Tell us what makes you a good fit for Acme Industries | Multi-line | Yes | No |

5.1.3 Displaying a warning for closed positions

You may have noticed that we have a **Position Status** field that is set to **Read-only**. This field has a purpose: we will populate it with a database lookup based on the position that is picked. Then, if the value is “Closed”, we will display a warning message to the user. Unlike what we did back in [Section 4.5.1](#) however, we still want users to be able to make form submissions, even if the position they are applying for is currently closed. The reason is pragmatic: it would be useful to have a bunch of resumes handy when the position is opened in the future.

Let’s begin by creating a database table in the **AcmeTables** database that we created back in [Chapter 4](#). If you need a refresher for the specific steps, refer to [Section 4.1.2](#). Our table will be called “PositionList” and have the following columns and values:

| Position Name | Position Status | Department |
|---------------------|-----------------|-------------|
| Account Manager | Open | Sales |
| Marketing Associate | Closed | Marketing |
| Test Engineer | Open | Engineering |
| Software Developer | Open | Engineering |

Nice. Now our HR team can modify this table when a position becomes open or closed, or new positions are created in Acme Industries.

Recall from [Section 4.5.2](#) that, before we can use our new table on our Job

Application form, we need to tell Laserfiche Forms about it. So we will do that from the **Administration** page first.

Next, let's go back to the form and configure a few lookup rules. We will have two of them connected to the **PositionList** table:

- 1 Fill POSITION DESIRED with data source value POSITION NAME
- 2 When POSITION DESIRED matches with data source column POSITION NAME
Fill POSITION STATUS with data source value POSITION STATUS

This should be pretty self-explanatory at this point: we populate the **Position Desired** drop-down list with the values from the **Position Name** field in our database table, then update the **Position Status** when the user picks one from the list. (We will use the Department column later in this chapter.)

Okay, now how do we display the warning? If you recall from [Section 5.1.2](#), we used a Custom HTML field to enter static text at the top of our form's third page. We will do the same thing here. However, instead of plain text, this time we will add some styling by modifying its HTML. Let's first type the following into our new Custom HTML field using the Visual editor:

```
Please note that the position you picked is currently closed. We still encourage you to complete your application and submit your resume. We will notify you once the position becomes open.
```

Then we will switch to the **Html** tab of the editor, and change the first HTML tag from:

```
<p>
```

to:

```
<p class="alert alert-warning">
```

When we do, our warning message should get a nice yellow background as well as a thin dark yellow border.

Box 5.1. Built-in styling

What just happened? It looked like we made a slight modification to an HTML tag, and doing so caused the element to automatically change its look-and-feel.

The answer is that we added a [CSS class selector](#) to an HTML element. CSS classes are used to group and classify styling rules. In this case, we added two classes, “alert” and “alert-warning”, to the paragraph inside our Custom HTML field, causing its styling to change. These classes (along with a host of others) come from a third-party CSS library that Laserfiche Forms uses, called [Bootstrap](#). We will play with CSS more and create our own styles later in this chapter.

The last thing we need to do is configure a couple of field rules. We will add the following:

The image shows a configuration interface for field rules. It consists of two rule boxes. The first rule box is for the 'Hide' action. It has a dropdown menu set to 'Hide', a field for 'Position Status', and a condition 'Ignore the data when' with a plus sign. Below this is a dropdown menu set to 'Always'. The second rule box is for the 'Show' action. It has a dropdown menu set to 'Show', a field for '<p class="alert alert-w', and a condition 'Ignore the data when' with a plus sign. Below this is a dropdown menu set to 'When any' followed by 'of the following is true:'. There is a sub-condition with a dropdown menu set to 'Position Status', a dropdown menu set to 'is', and a text field containing 'Closed'. There is also a plus sign icon at the bottom left of the second rule box.

Figure 5.1: Field rules for Position Status.

We configured the **Position Status** field to be always hidden since it’s redundant. Lookups will still populate that field though, and when its value is “Closed”, our warning message will be displayed.

Preview Mode: Data entered will not be submitted.

ACME INDUSTRIES
It Might Just Work!

Job Application

Thank you for your interest in Acme Industries. Please fill out this form and submit it. You will hear back from us within one week.

Position Information
Page 3 of 3

Acme Industries is an equal opportunity employer and will not discriminate against any employee or applicant on the basis of age, color, disability, gender, national origin, race, religion, sexual orientation, veteran status, or any classification protected by federal, state or local law.

Position Desired*
Marketing Associate

Please note that the position you picked is currently closed. We still encourage you to complete your application and submit your resume. We will notify you once the position becomes open.

Salary Desired
\$

When can you start?*

Type*

Resume*

Figure 5.2: Our fancy warning message.

Great! We are done with the job application form. Let's move on to the next one.

5.2 Creating the Recruiter Checklist form

Our job application will be hosted on the Acme Industries website, meaning it will be open to the public. As such, we need to account for spammers as well as people who take the “shotgun approach” to job hunting. So we are going to create a second form that Acme recruiters can use to screen applicants and make sure their applications meet Acme's quality standards.

This form will be a bit different than the forms we have created so far. Specifically, we will make ample use of Field Variables by inserting them straight into our form. This will allow the information the applicant submits via *their* form to be displayed on the Recruiter Checklist form. You will see why this may be desirable.

Go ahead and set the form's name to "Recruiter Checklist". Just like we did with the Job Application form, we will also change the **Backend Validation** option to "no validation". Then, let's also set the title as "Recruiter Checklist" to match the form name and enter the following description:

Use this form to ensure the application meets Acme's quality standards. If it does, please route it to the appropriate hiring managers.

We will also use the Acme Industries theme so that the look-and-feel is consistent with the first form.

5.2.1 Applicant's Basic Information

The Recruiter Checklist form will have two sections. The first section will contain the applicant's information, and the second section will contain a checklist as well as the list of departments to route the application to.

Let's start by inserting a new Section and setting its title to "Applicant's Basic Information". Then, instead of inserting new fields, we will insert Field Variables instead. We can do this by switching to the **Variables** tab.

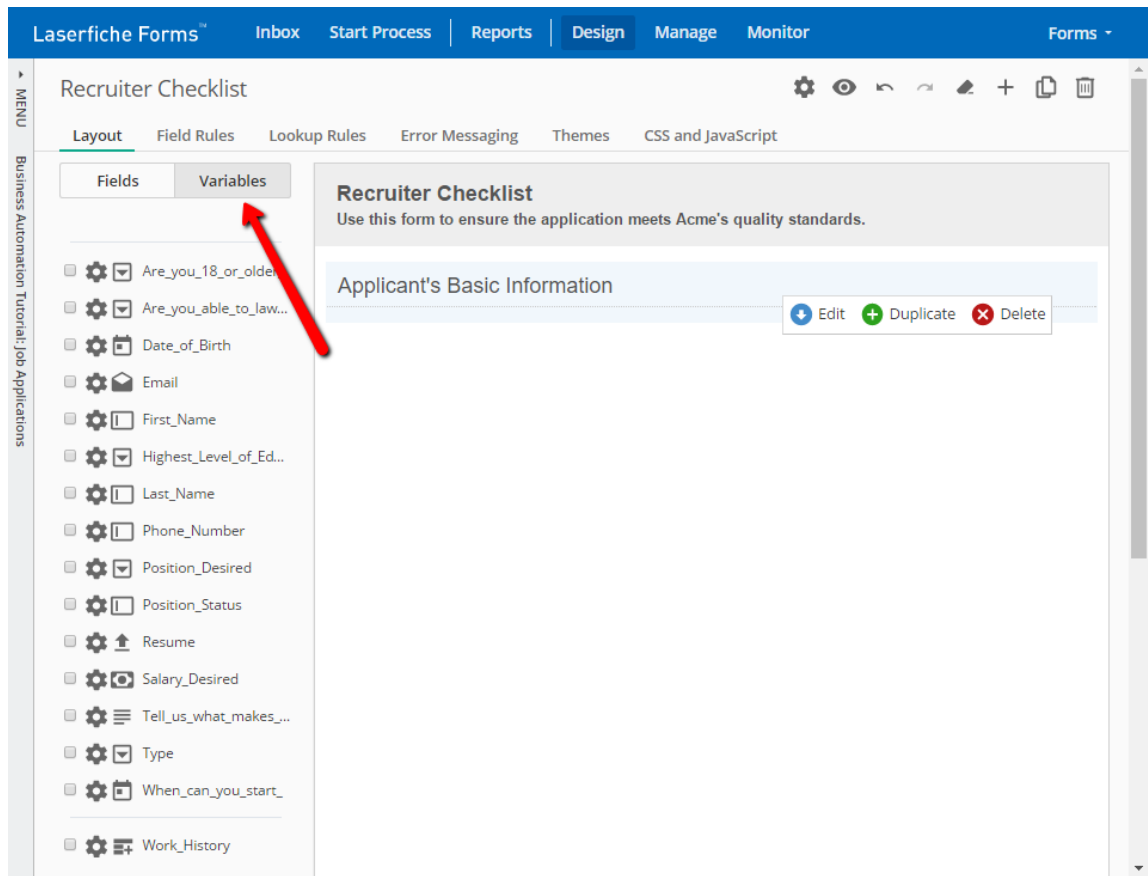


Figure 5.3: The Variables tab is available on the left pane.

We will insert the following Field Variables into the form:

- First Name
- Last Name
- Are you 18 or older?
- Are you able to lawfully work in the U.S.?
- Highest Level of Education Completed
- Tell us what makes you a good fit for Acme Industries

- Resume

You will note that each field is already set as **Required**. This is because we set them as such on the Job Application Form. *However, anything we change on the fields now will not be reflected back on the Job Application.* They will continue to refer to the same variables, but it is possible to set their options and labels differently.

This is a bit confusing, so let's give it a shot. First, let's uncheck the **Required** option for each field, and check the **Read-only** option. Then, let's change the multi-line field's label to "Applicant's self-summary". Now when we preview the form we should see the changes we made, *but* if we go back to the Job Application form and preview it, we see that the fields on it have retained their original options.

If you think about it, this makes sense: it allows different users to use the information in different capacities. Applicants should be able to type their information into the form fields and recruiters should be able to view them, but not change them.

Box 5.2. Different ways to make information read-only

You may recall from [Section 4.4.2](#) that User Tasks also have the option to make their designated forms read-only for the users participating in that task. Why did we not simply use that option, and instead painstakingly marked fields as read-only one by one?

The answer is that option makes the entire form read-only, and that's not what we want here. The recruiter will not be able to change the information the applicant submitted, but they will still need to enter their own data.

Alright, that is enough information for the recruiter to be able to tell if the applicant is promising. Note that the recruiter won't make the hiring decision. They just need to make sure the applicant is over 18, can lawfully work in the United States and uploaded a decent resume. (We could make the first few

criteria determined automatically by Forms and reject those applicants, but in my experience it's almost never a good idea to perform rejections automatically in HR settings, due to both technical and legal reasons.)

5.2.2 Recruiter Screening

Let's now insert a new Section into the form and set its title to "Recruiter Screening". This section will start with a checklist, titled "Screening checklist" that has the following options:

- Applicant is over 18
- Applicant has work authorization
- Applicant's resume is acceptable
- Applicant sounds enthusiastic

If the applicant meets all the above criteria, the recruiter will route the resume to the appropriate department. But which department is that?

We could simply route the applicant based on their desired position, which is a Required field on the Job Application form. But we want our recruitment process to be a bit more robust than that. Often times, people applying to a specific position turn out to be eligible for multiple positions. For example, an Account Manager might be able to also work as a Marketing Associate depending on their background. So we should figure out a way to route this application to multiple departments.

How do we do that? The simplest approach would be to use a checklist field that has a list of departments. But that's problematic, because our list of departments is determined by our **PositionList** database table. We could use a dropdown list, but that has a shortcoming too: by default, only a single item can be selected on it, whereas we want to be able to route the form to multiple departments.

We have hit a snag. We have a business requirement that cannot immediately be met with the tools available to us out of the box. Fortunately, Laserfiche Forms offers an astounding level of customizability. With a little bit of

JavaScript magic combined with advanced conditional expressions, we can accomplish what we need. Let's do that now.

5.2.3 Multi-select drop-down fields

By default, drop-down fields in Laserfiche Forms allow picking only a single value. However, Laserfiche Forms is based on HTML5, which means it uses modern web standards and can be easily customized.

First though, let's set up the stage. We will insert a drop-down field and label it "Departments" and also make it Required. Then, we will hook it up to our **PositionList** database table by adding a new lookup rule:

```
1 Fill DEPARTMENTS with data source value DEPARTMENT
```

When we save the lookup rule and preview the form, we should see that drop-down field get populated with our list of departments.

Peeking into the CSS and JavaScript rabbit hole

It's time to sprinkle some CSS and JavaScript on our form! Open the **CSS and JavaScript** section of the form and let's get started.

This section consists of three panes. The top-left pane is the CSS pane. CSS stands for "Cascading Style Sheets". Recall from **Box 5.1** that it is used to style web pages and change their look-and-feel. The bottom-left pane is the JavaScript pane, which allows customizing a web page's behavior. Lastly, the right pane is the built-in preview pane. It updates automatically as CSS and JavaScript changes are saved, which makes it a very handy tool.

The other thing that makes the built-in preview pane useful is that it shows what are called "CSS Selectors". For example, my **First Name** field has the following selectors:

```
id="q2", class="form-q label-left"
```

These CSS Selectors offer an easy way for us to refer to specific elements on the page when we write styling rules and JavaScript. For example, let's say that we want to programmatically set the value of our **First Name** field to "Bobby". We can do this using this bit of JavaScript:

```
$(document).ready(function() {  
    $("#Field2").val("Bobby");  
})
```

When we save our changes, the built-in preview pane should show the results of the script.

Web design and programming are vast, complex topics, and we are not going to go into too much detail here. Instead, we will take a pragmatic approach: I will provide the code and explain what it does. You are strongly encouraged to experiment with it however. To this end, I have provided some resources at the end of this chapter aimed at those looking to "get their learn on", as one of my ex-coworkers used to say.

Box 5.3. Field identifiers

You may have noticed that the identifier we used in our JavaScript snippet above is different than the CSS selector shown on the built-in preview pane. One says **#Field2** whereas the other says **#q2**. The reasons for this are complicated, but one way to think about it is that "#q2" refers to the entire field element (the label, any text above and under the field, etc.) whereas **#Field2** refers to only the field itself, which in this case is an **input** element represented by something like the HTML below:

```
<input type="text" id="Field2" name="Field2" readonly="True"  
class="singleline noborder cf-medium">
```

If you are familiar with HTML, great. If not, don't worry: we aren't going to do anything crazy with it in this tutorial. However, learning HTML is a great start

to learning web programming in general. Make sure to check out the resources at the end of this chapter if you are curious.

So let's take a closer look at the code snippet we wrote above. What does it mean exactly? Well, it actually consists of two parts. The first part is known among JavaScript programmers as the **document.ready block**. It is an event listener that says, "run the code inside this block when the page is ready to be interacted with". The simpler the form, the smaller its footprint, and therefore the quicker the page (i.e. the HTML document) will become "ready". The second part, which is line 2, simply says, "set the value of the element with **id** of **#Field2** to "Bobby".

There are many other event listeners. For example, the "change" event triggers when the element it is attached to changes. Here is an example for **#Field11**, which is my **Departments** field (you are encouraged to follow along, by the way):

```
$(document).ready(function() {  
    $("#Field11").on("change", function() {  
        alert("You just picked a department!");  
    });  
})
```

Now when we save it and pick a department, we should get a pop-up that states the obvious.

Okay, that's enough about overly simplistic examples. Let's do something useful. As mentioned previously, we want to change our **Departments** field from a single-select to a multi-select. Here is how we do it.

```
$(document).on("onloadlookupfinished", function() {  
    $("#Field11").attr("multiple", true);  
})
```

When we save our code, we should see that the **Departments** field has changed. It is now a list and has scrollbars!

It's a bit too short though. We should also change the field's height:

```
#Field11 {  
    height: auto;  
}
```

Now when we save our changes and preview the form, we should see an expanded dropdown whose height matches the contents. And here is the cool thing: if we hold down the Ctrl key, we can select more than one item!

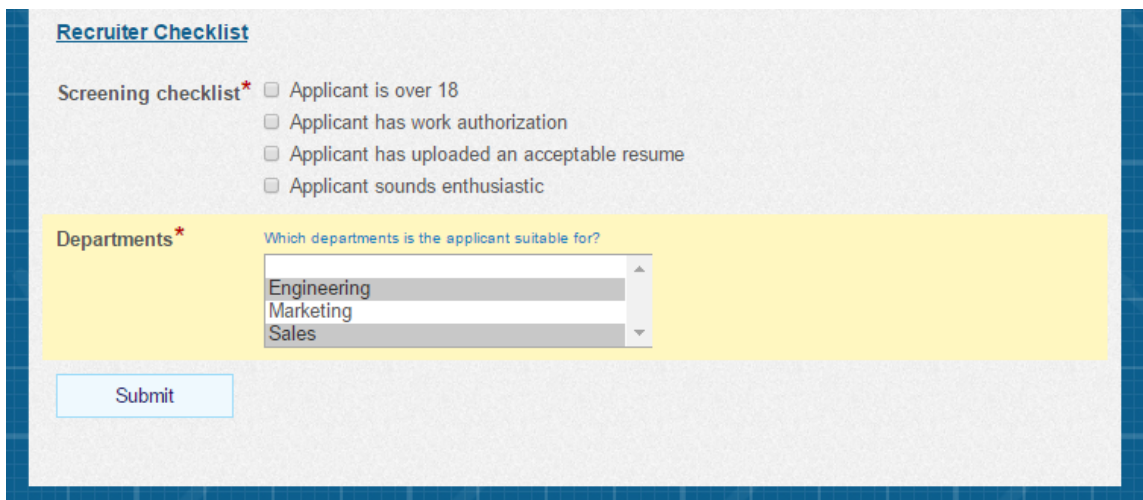


Figure 5.4: Selecting multiple items on a dropdown field.

This is one of the many advantages of Laserfiche Forms being HTML5-based. We used JavaScript, a web programming language, to add the **multiple** attribute to a `<select>` element, which represents a drop-down field. What this means is that, if we learn HTML, CSS and JavaScript, we can utilize those skills to add many fancy customizations to forms we build in Laserfiche.

5.2.4 Advanced conditional routing

Okay, now our recruiter can send the form to multiple departments... or can they? We used JavaScript to change our department drop-down to a multi-

select field, but what happens when they actually select multiple departments and submit the form?

Inclusive Gateway

The first thing we need to do is make sure the Message Start Event leads to a Recruiter Review User Task. This User Task should have the Recruiter Checklist form, and it should be assigned to the **Forms** user. We will assign all of the User Tasks to this user in this chapter, because doing that is easier than creating a bunch of user accounts, which we don't want to get bogged down with.

The Recruiter's task should have two outflow paths: one for routing the form to different departments, and another for discarding (which should end the process). We have already covered how to perform this type of routing. If you need a refresher, refer to [Section 3.2.3](#).

So let's think about where the approval outflow path should lead. Since we want our form to route to potentially several departments, we can't use an Exclusive Gateway like we did in [Chapter 3](#). With an Exclusive Gateway, only one of the branches can run - the first one that matches. If we want the process to be able to route to more than one outflow path, we need to use an Inclusive Gateway.

Let's add one, then connect it to three separate User Tasks, one for each department. We will name our User Tasks "Sales Review", "Marketing Review" and "Engineering Review", assign each one to the **Forms** user and pick the original Job Application form as a read-only form.

Box 5.4. A stripped down job application form

In a more realistic setting, we may want to create a third form that is a stripped down version of the job application. Since the applicant will have been screened by the time it reaches the hiring managers, they probably don't need to see that the applicant is over 18 and has work authorization. In addition, we may want to exclude the **Position Desired** field from this stripped down form so as to not

introduce bias into the hiring decision. If the Engineering Manager sees that the applicant desires a sales position, they may decline the applicant right away!

This is what my process looks like so far:

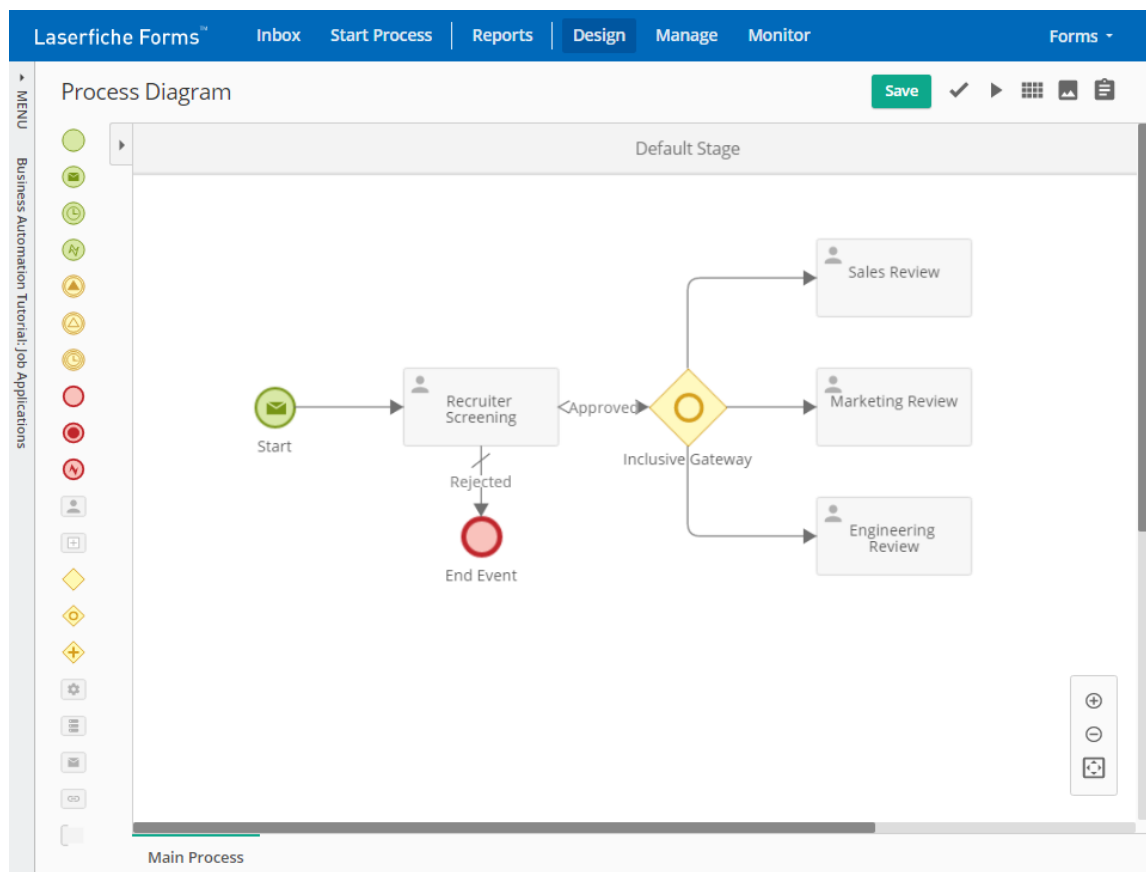


Figure 5.5: Partially-built Process Diagram with an Inclusive Gateway.

Let's configure the details now.

XPath Expressions

Now the million-dollar question: how do we configure the branch conditions?

With a single-select dropdown, the answer is obvious: the field's variable would contain the selected value, which we can match using the built-in “=” operator. With a multi-select dropdown, it's a bit more complicated. When a form is submitted with multiple items selected on the dropdown field, they are combined together with a comma separator. For example, if we select Sales and Engineering in the **Departments** list and submit, the variable will have the following value:

```
Sales,Engineering
```

What this means is that, when we configure our conditional expressions, we can't simply use the “=” operator. For example, if we use the conditional expression below:

```
/dataset/Departments="Sales"
```

It would work if and only if Sales was selected. If both Sales and Engineering were selected, the condition would return false, since “Sales,Engineering” is not equal to “Sales”.

The good news is that there is a solution. Laserfiche Forms conditional expressions are based on something called [XPath](#), which is a query language for selecting nodes from XML documents. XPath itself is pretty complicated, and unless you work with XML documents a lot, you probably won't gain too much value out of learning it. However, in Laserfiche Forms it can be used to create more complex conditional expressions when the ones that are built-in are insufficient.

We need a way to specify our conditions such that they will return true if the **Departments** variable *contains* that department. We can do that with the following XPath statement, which we can type into the Conditional Expression field:

```
contains (/dataset/Departments, "Sales")
```

With this expression, even if the Field Variable's value is "Sales,Engineering", the form will be inclusively routed to the Sales department for review. Let's go ahead and configure the other two outflow paths using the **contains** operator.

5.2.5 Merging gateways

There are three types of gateways: Exclusive, Inclusive and Parallel. Each gateway can be a *splitting* gateway or a *merging* gateway. The gateways we have covered so far were all splitting gateways: they split the process flow into multiple paths. Depending on the process, these paths may also need to be merged.

Let's think about how the rest of the process should flow. After each hiring manager receives the job application for review, they should have two options: approve or reject. If they reject the applicant, their part of the process flow should end. But if they approve, the recruiter should be notified so that they can contact the applicant for an interview. However, each department's interview will be at a different time, so it makes sense to schedule them as each department makes their decision (besides, it doesn't make sense for the recruiter to wait for straggling hiring managers, as the applicants can receive job offers at other companies).

So we are going to merge the three branches of our process using an Exclusive Gateway. This gateway will allow the data from each incoming path to proceed to the outflow path when it reaches the gateway. Let's go ahead and insert one now and connect our three User Tasks to it. Then, we will connect it to a final User Task labeled "Schedule Interview". This User Task will also be assigned to the **Forms** user, and it will have the Job Application form. It will also be connected to an End Event.

Lastly, we want to give each department the ability to reject the application. So we will connect each of their User Tasks to an End Event as well.

I am intentionally refraining from telling you every single thing that needs to be done. Use the **Validate** button to validate the process and fix any errors

that it may raise. Here is what my finished Process Diagram looks like:

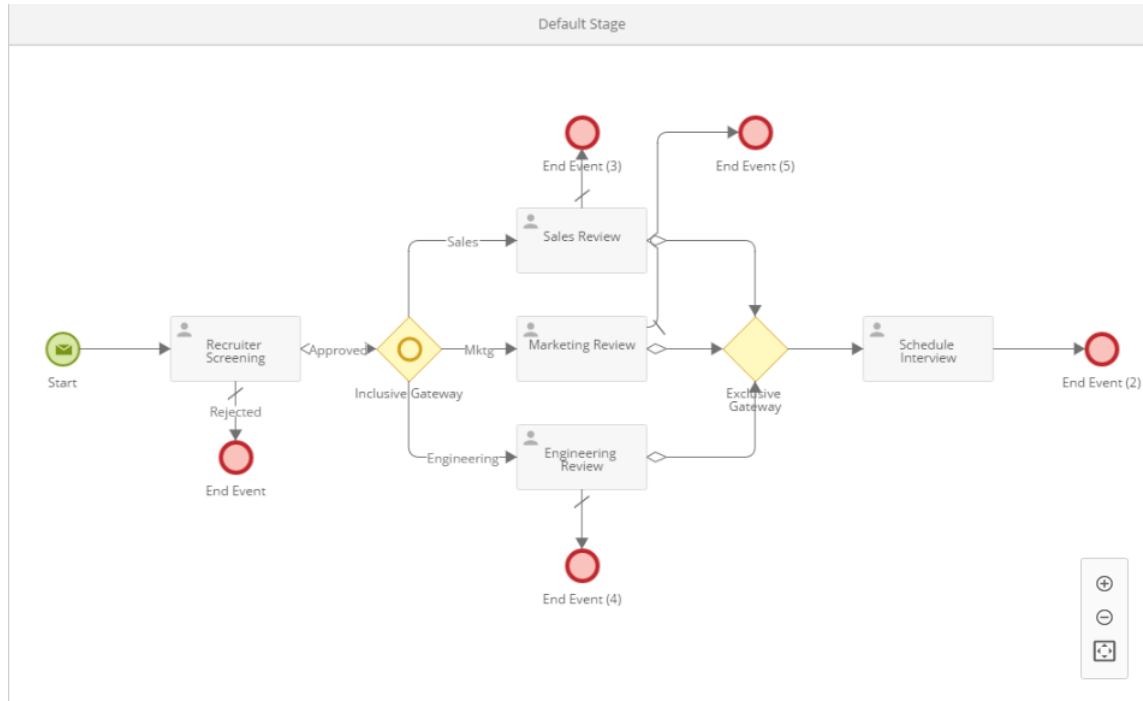


Figure 5.6: Finished Process Diagram.

By the way, Laserfiche’s documentation on different types of gateways is excellent. If you want to be able to design complex processes in Laserfiche Forms, you should definitely [check it out](#).

5.3 Testing the Job Application Process

Let’s go ahead and publish our job application, making sure that its link is a friendly URL (such as “<http://localhost/Forms/JobApplication>”) and that the instance name is easy to distinguish for someone looking at their Laserfiche Forms Inbox. For this Business Process, it might make sense to use the following:

```
Job Application - {/dataset/First_Name} {/dataset/Last_Name}
```

Our Forms user is already a Process Admin, so we can start the Business Process right away.

If you configured everything correctly, the following should happen:

1. When filling the Job Application form, if we pick “Marketing Associate”, we should see our fancy warning message.
2. When filling the Recruiter Checklist form, if we pick more than one department from the **Departments** field (which should be a multi-select list field), it should create User Tasks for all of those departments.
3. When reviewing the application as each department, the entire form should be read-only (which will also make it so that it is displayed on a single page).
4. If a department rejects the applicant, their process should end. A Schedule Interview task should appear in the Forms Inbox *only if* a department approves their task.

Box 5.5. Possible validation errors

If you get validation errors when you submit the Job Application or the Recruiter Checklist forms, make sure you have turned off the **Backend Validation** option from form settings by setting it to “no validation”.

If everything went well, great! Feel free to go back and make other improvements to the process, such as by adding email notifications to various tasks, and creating a third (and maybe even a fourth!) form to assign to the department review tasks and the schedule interview task. Regardless though, make sure to pat yourself on the back. The process you created is not a simple one. In fact, I think you can even show it off to your HR team and see what they think!

5.4 Chapter summary

Even though this chapter was a bit shorter than [Chapter 4](#), we probably covered as much if not more ground. We learned about:

- Page breaks on forms and their benefits
- Custom error messages with fancy styling
- Utilizing multiple forms in a single Business Process to control information access
- Changing a dropdown field to a multi-select list using JavaScript
- Advanced conditional expressions using XPath
- Splitting Inclusive Gateways and Merging Exclusive Gateways

That said, we only skimmed the surface on some of these. XPath has a lot of different options, and JavaScript and CSS are a world unto their own. At the end of [Chapter 6](#), I have provided several resources for further learning. Make sure to check them out if you want to improve your skills as a business process automation expert.

Chapter 6

Reporting and Optimization

Welcome to the final chapter, in which we will modify the Job Applications Business Process we built in [Chapter 5](#) by optimizing it for reporting and efficiency. This chapter is going to be shorter than the others, but I expect the information to be as valuable, if not more so. Information is the lifeblood of today's business, and being able to quickly provide it to key decision-makers in the company can be a very valuable skillset.

Without further ado, let's jump in.

6.1 Process Stages

We will open the Process Diagram for our Job Applications Business Process. Here is what mine, and hopefully yours, look like:

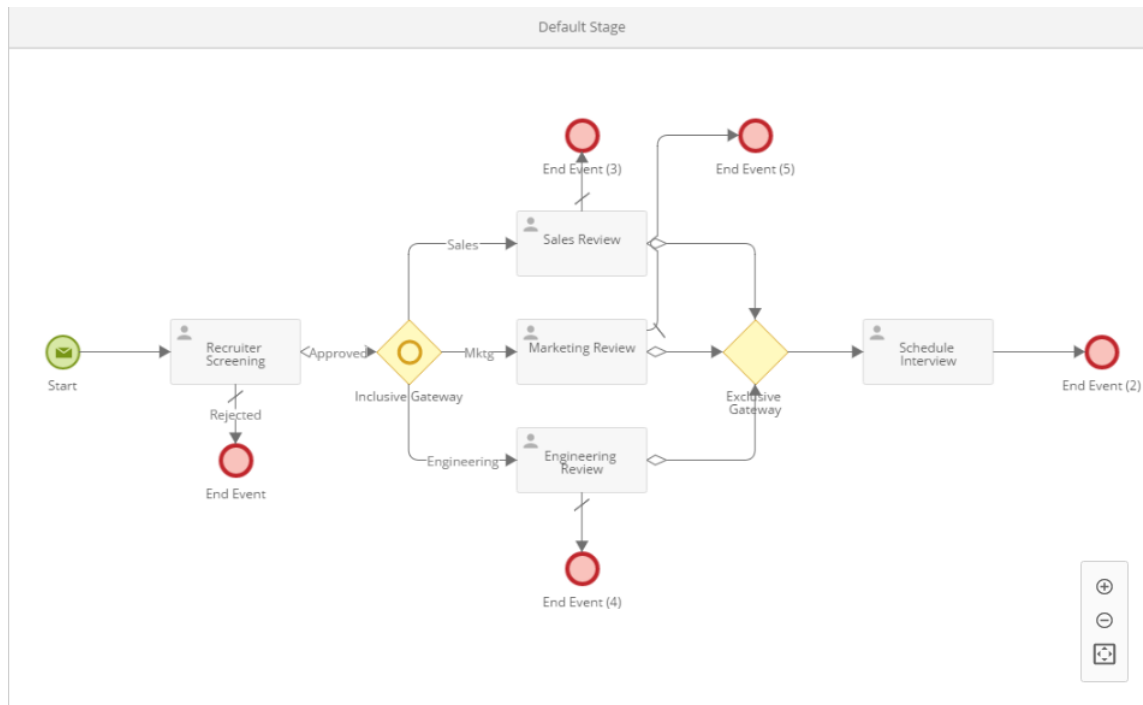


Figure 6.1: Finished Process Diagram with a single stage.

Let's say that we publish this process and let it run for a few months. One day, the Director of Human Resources comes to our office and complains that Acme Industries has been growing quickly and we need to accelerate our hiring. We know that the Business Process we published has been a big help (the old process was paper-based... yuck!), but we believe we can make improvements. Here is the problem though: how do we identify bottlenecks?

The way our Process Diagram is currently designed does not allow for deep analysis of what tasks may be taking the longest. Is our recruiter taking a long time screening applications (perhaps because too many are coming in), or are the hiring managers taking too long to review applications? Yet another possibility is that we call potential applicants up to three different times to schedule their interviews with different departments, which may be giving a bad impression and leading them to take jobs with other companies.

Let's figure out if we have bottlenecks first. Our Process Diagram essentially has three stages: Screening, Application Review and Interview Schedul-

ing, but they are all lumped into one stage, called Default Stage, seen at the top. Let's go ahead and split it into three:

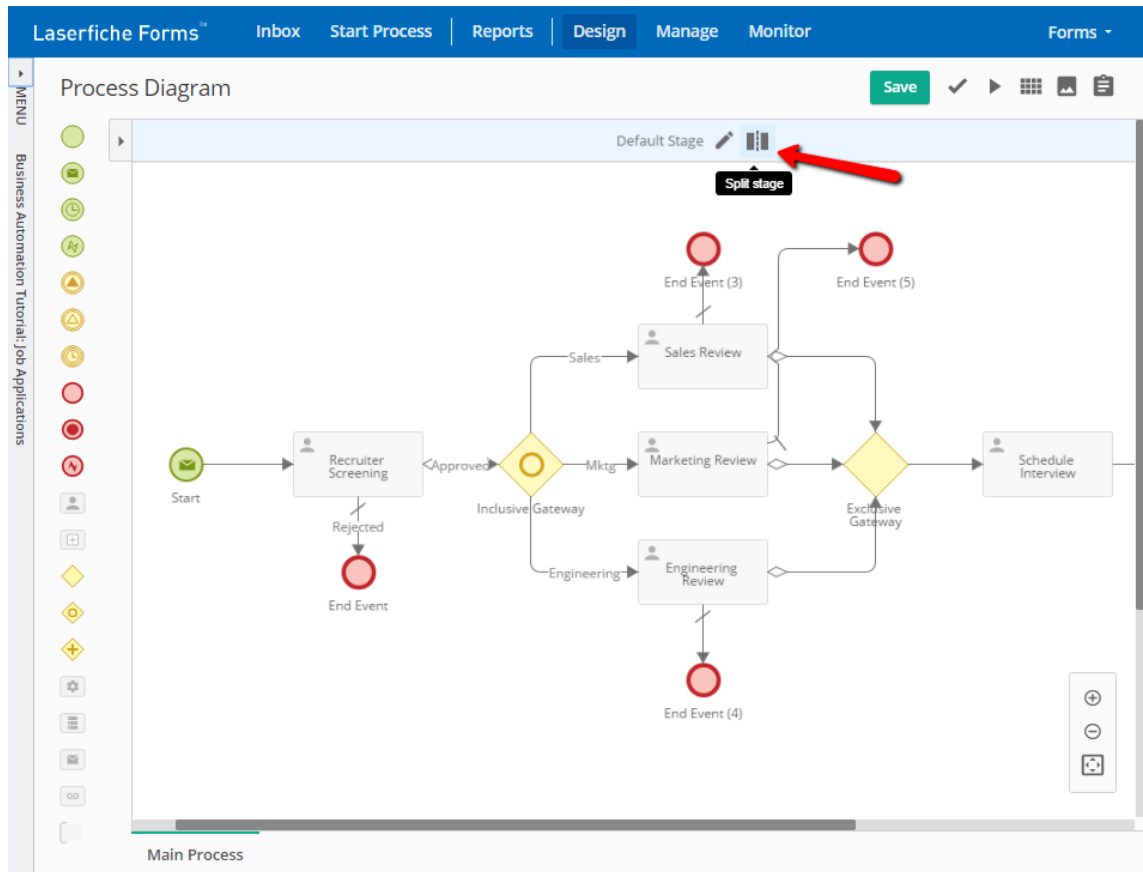


Figure 6.2: Splitting a stage.

Laserfiche Forms tries to be helpful when splitting stages by showing several different locations where the stage can be split. Once we rename the stages appropriately (which you can do by double-clicking the stage name), we will also reorganize the User Tasks on the Process Diagram inside the corresponding stage. The Recruiter Screening task will be in the Screening stage, the three department review tasks will be in the Application Review stage and the Schedule Interview task will be under Interview Scheduling. Below is what it should look like (I reorganized the Start and End Events to fit the screenshot):

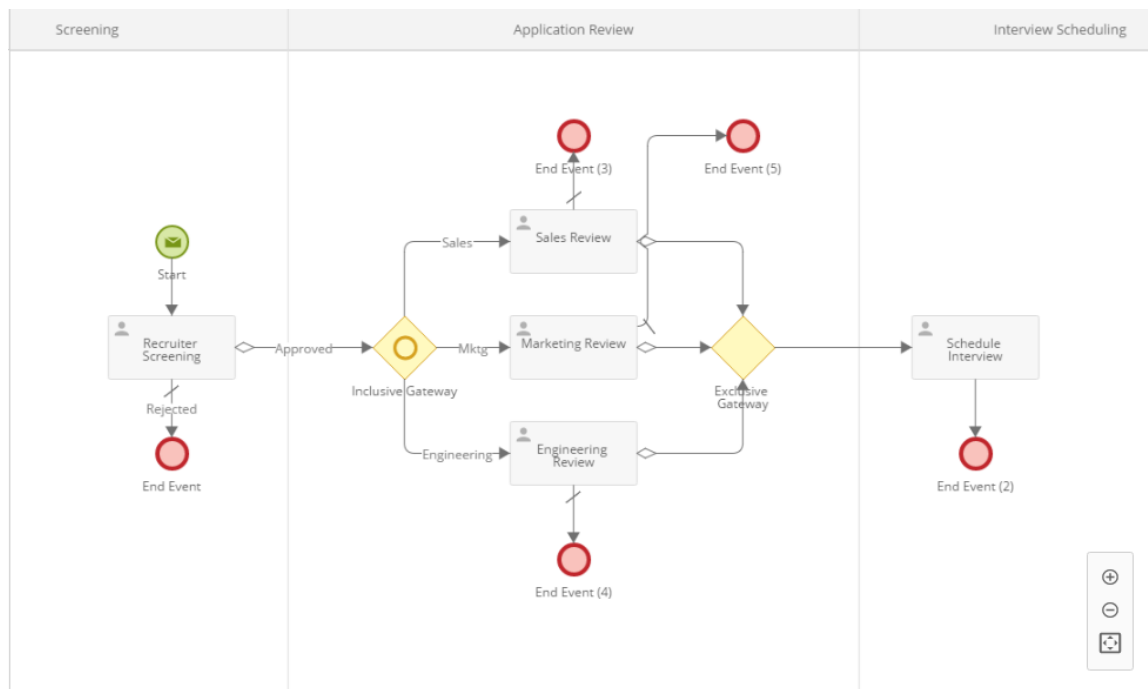


Figure 6.3: Process split into three stages.

The Business Process will continue to run the way we designed it, and running instances will not be affected. But now, Laserfiche Forms will be able to provide detailed breakdowns of how long each stage is taking and show us where the delays are using the Auto Reporting feature.

6.2 Auto Reporting

As the name implies, Auto Reporting provides automatic, process-level reports. In our current Business Process, it will give us some insight into the different stages we configured. Let's see this in action by switching to the process dashboards. This can be done by clicking the **Reports** link on the top navigation bar and selecting the Job Applications Business Process.

The Auto Reporting page has four sections.

6.2.1 Operational Dashboard

The Operational Dashboard shows analytics for instances of the Business Process that are currently running. In our case we have none (or a few, if you have not yet completed some of the User Tasks from [Section 5.3](#)). I'm going to submit two job application forms and screen one of them to populate this page with some data.

When we come back to the Operational Dashboard, we see some useful overview statistics:

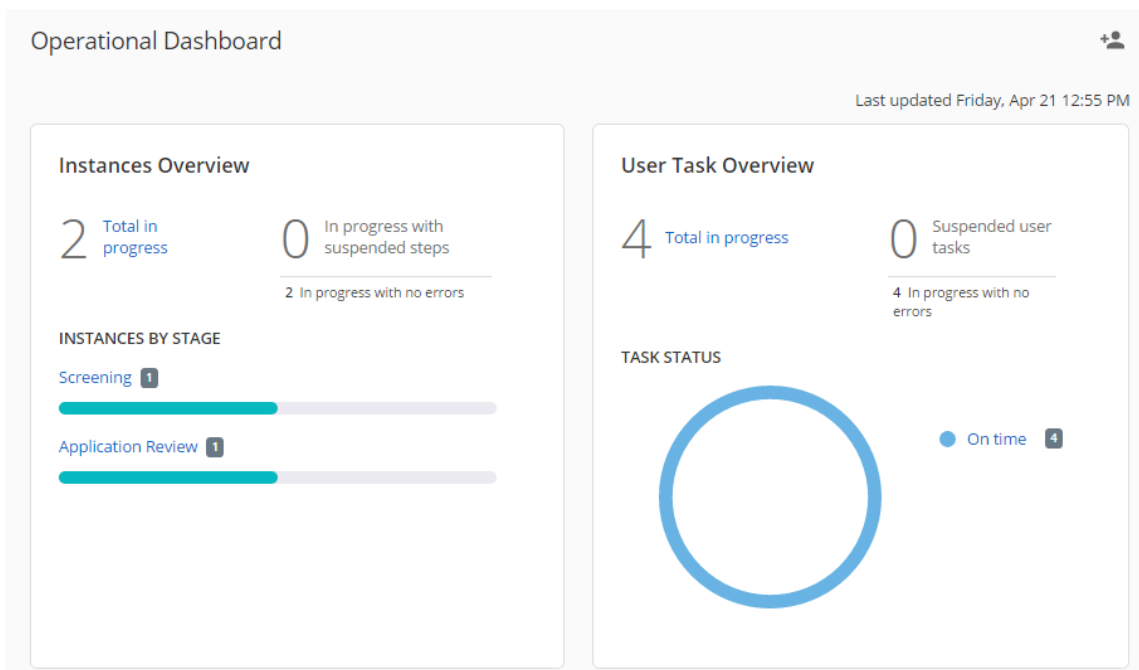


Figure 6.4: Overview metrics in the Operational Dashboard.

We have two instances in progress, and Forms shows us a breakdown by stage using the stages we configured in [Section 6.1](#). We can also see an overview of User Tasks, of which I have four. In our case, all of our tasks are on time. If we had configured due dates for our User Tasks, then some of them could potentially be shown as overdue. We will do this later in this chapter when we optimize our Business Process.

Let's also take a look at the User Task Distribution:

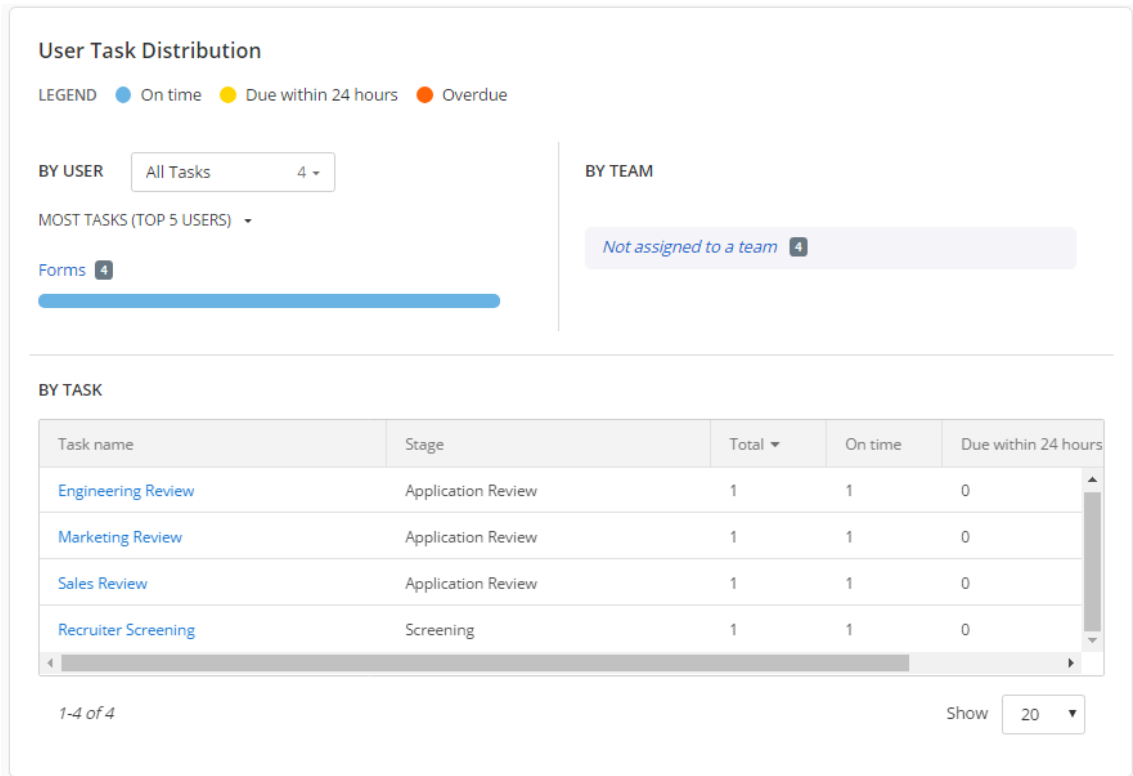


Figure 6.5: A distribution of active User Tasks by user.

This section is not particularly useful to us right now, because all of our User Tasks are assigned to the **Forms** user. However, in a Business Process that is in production and being used by many users, it can offer a lot of insight into workload. For example, the User Task overview may show that 40% of tasks are overdue, and you can look at the User Task Distribution to find out that it is because one of your recruiters are doing most of the screening and is overwhelmed. That could offer a valuable opportunity to optimize the process by evening out user workloads.

6.2.2 Performance Dashboard

In contrast to the Operational Dashboard, the Performance Dashboard provides insight into the Business Process based on completed instances. In addition to a short summary at the top of the page, it displays the duration of completed instances:

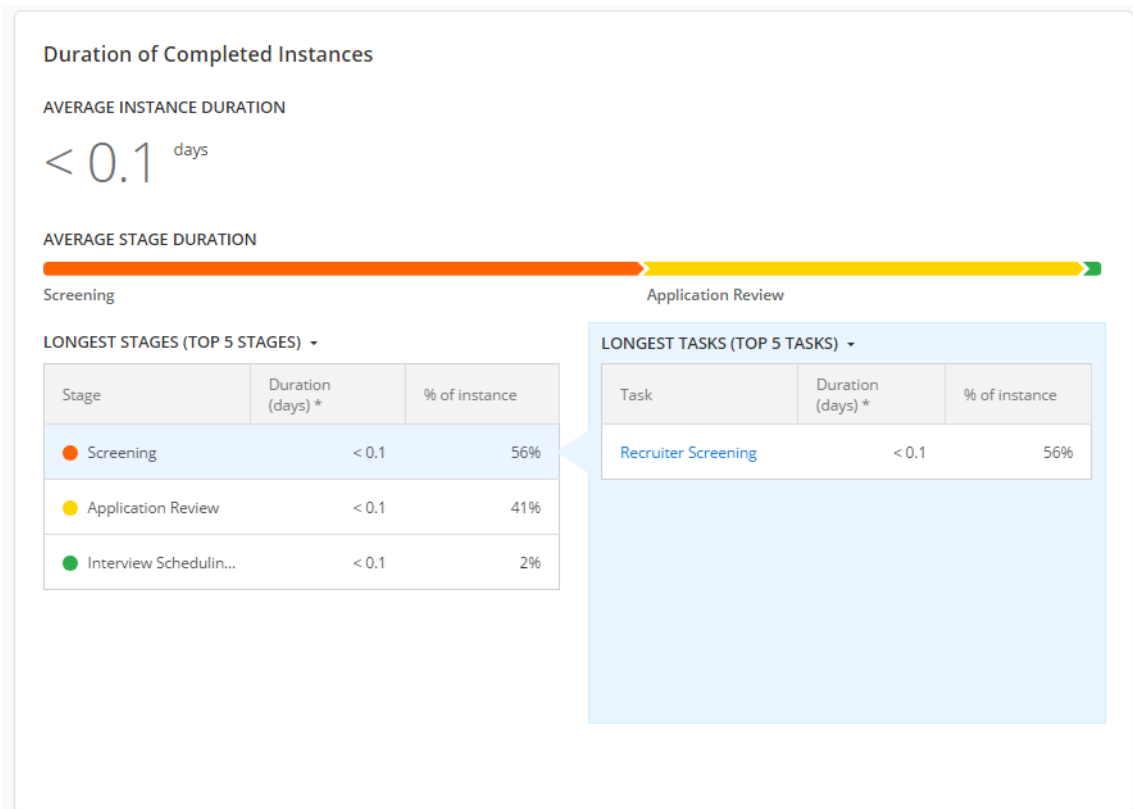


Figure 6.6: Duration of completed instances by stage and task.

We have been completing our User Tasks fairly quickly since this is a tutorial setting, but in a Business Process that is in production, this section offers useful information about how long each stage and task takes to complete. Note that an even breakdown of durations is unlikely, since the User Tasks in each process stage may require different levels of time investment. For example, screening a job application may take 2-5 minutes, but actually going through

the application, reading resumes and cover letters and discussing it with fellow coworkers may take 30 minutes or longer. Just because a stage or task takes a long time does not necessarily mean there is anything wrong with it.

On the other hand, seeing that a particular User Task takes a long time can be an opportunity for optimizing it. For instance, you may be able to break down a single “screen job application” task into two tasks, one for performing the background check and another for checking the applicant’s references. I call this the “assembly line approach”: by breaking down a process into its smallest possible units, you allow your employees to build expertise in that area, improving the efficiency of the overall process.

At the bottom of the Performance Dashboard, we can see the occurrence levels:

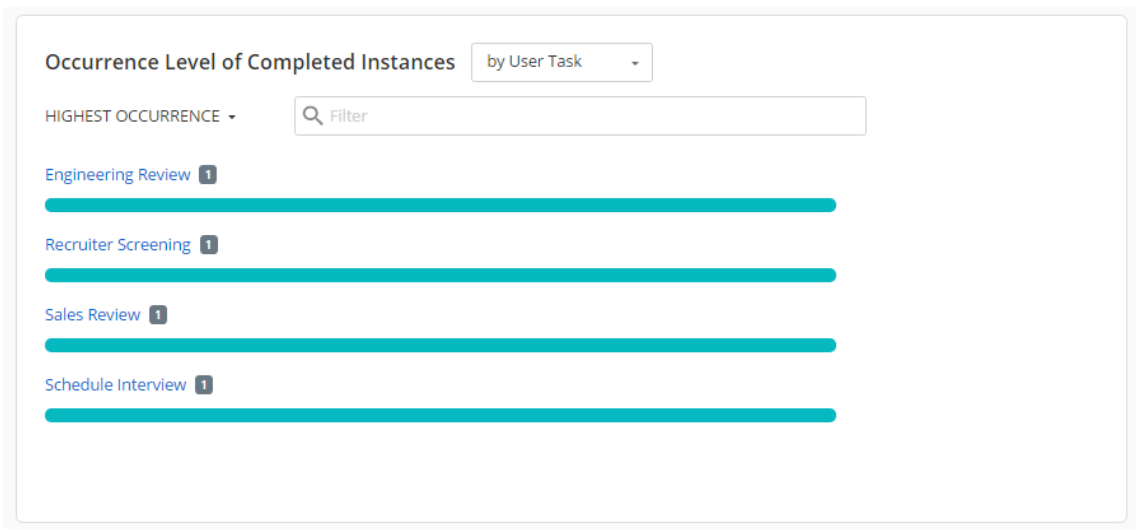


Figure 6.7: Occurrence levels of User Tasks.

This can be useful in complex Business Processes with many Inclusive and Exclusive Gateways. In our case, we can see an overview of how many job applications are routed to each department, which may help us notice a shortage on applicants of a specific type and spark conversations about applicant sourcing. For example, maybe our recruiters need to attend career fairs at engineering schools, or focus their efforts on contacting promising sales and marketing

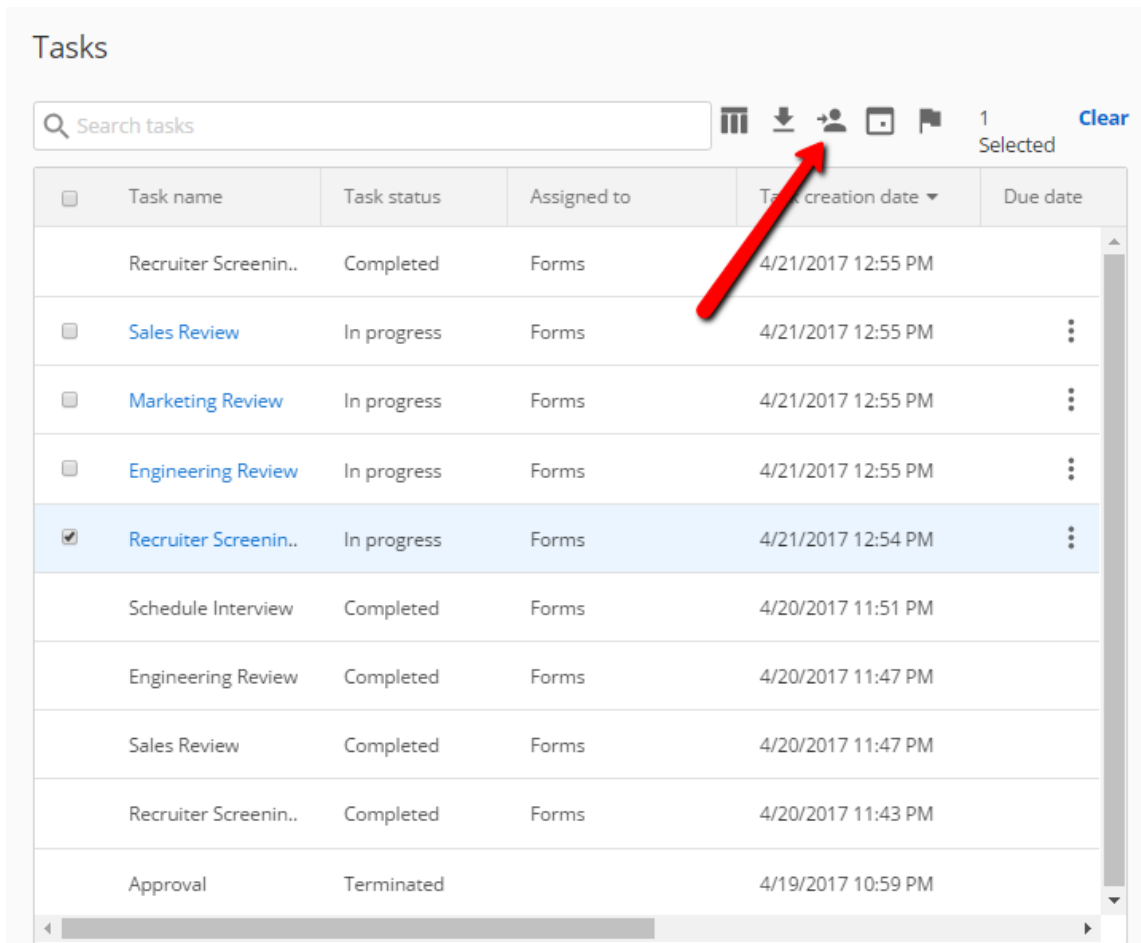
candidates.

Box 6.1. Reporting date ranges

By default, the Performance Dashboard displays data from the last 30 days. This date range can be changed from the top right corner.

6.2.3 Instances and Tasks

In addition to the two dashboards, we can also look at specific instances and User Tasks in detail. This allows us to examine the instance steps and open the User Tasks as a process administrator. In addition, process instances can be terminated (which also cancels all of their active tasks) and User Tasks can be assigned to other users. For example, if one of our recruiters has a rather heavy workload, we can come to the Tasks section and reassign some of their tasks to another recruiter.



Tasks

Search tasks

1 Selected Clear

| <input type="checkbox"/> | Task name | Task status | Assigned to | Task creation date | Due date |
|-------------------------------------|----------------------|-------------|-------------|--------------------|----------|
| <input type="checkbox"/> | Recruiter Screenin.. | Completed | Forms | 4/21/2017 12:55 PM | |
| <input type="checkbox"/> | Sales Review | In progress | Forms | 4/21/2017 12:55 PM | |
| <input type="checkbox"/> | Marketing Review | In progress | Forms | 4/21/2017 12:55 PM | |
| <input type="checkbox"/> | Engineering Review | In progress | Forms | 4/21/2017 12:55 PM | |
| <input checked="" type="checkbox"/> | Recruiter Screenin.. | In progress | Forms | 4/21/2017 12:54 PM | |
| <input type="checkbox"/> | Schedule Interview | Completed | Forms | 4/20/2017 11:51 PM | |
| <input type="checkbox"/> | Engineering Review | Completed | Forms | 4/20/2017 11:47 PM | |
| <input type="checkbox"/> | Sales Review | Completed | Forms | 4/20/2017 11:47 PM | |
| <input type="checkbox"/> | Recruiter Screenin.. | Completed | Forms | 4/20/2017 11:43 PM | |
| <input type="checkbox"/> | Approval | Terminated | | 4/19/2017 10:59 PM | |

Figure 6.8: Reassigning a User Task from the Tasks page.

Lastly, the Instances page is particularly useful if a process instance gets suspended and terminated, which can happen if the process is incorrectly configured or an external service task (such as a Laserfiche Workflow that is invoked from from the Business Process) errors out. Keep in mind that while Suspended tasks can be retried, Terminated tasks cannot be recovered.

6.3 Forms Reporting

In addition to automatic reporting, Laserfiche Forms also allows creating custom reports for more ad-hoc reporting. Let's give this a try by clicking the **Create Report** link on the left pane of the Reports page.

One of the fields in our Job Application form asks the applicant if they are allowed to legally work in the United States. Previously, we were hiring only those with work authorization. However, as of yesterday, Acme Industries sponsors work visas for exceptionally skilled foreign applicants. So we want to create a report that will allow us to see a list of those applicants so that we can contact them to apply again.

Before we can utilize this report however, we need to make sure we have actually submitted a few job applications and picked "No" on the work authorization field, and also specify the applicant has a Master's degree or a Ph.D. Otherwise our report will be empty! So, if you haven't submitted any such applications yet, make sure to do so now, then come back to this page.

Let's name our report "Foreign Applicants Report", then switch to the **Detailed Data Set** tab. We will remove the default set of columns, then add the following columns and labels:

```
* First_Name (First Name)
* Last_Name (Last Name)
* Email (Email)
* Phone_Number (Phone)
* Start_date (Applied on)
```

We need to also set a sort order in the Column Options section at the bottom. I set mine to sort by Start Date.

Then we will switch to the **Data Filter** tab and add a filter for the work authorization field's Field Variable, with a value of No. We also want to only contact foreign applicants who have Master's degrees or higher, and have applied recently. So we will set the appropriate criteria for those as well. Here is what my data filters look like:

← Edit Report Save

General Detailed Data Set Data Filter

Include instances that meet the following criteria ×

| | | | |
|----------------------------|-------------|-----------------|---|
| Are_you_able_to_lawfully.. | is selected | No | × |
| Start date | is after | 2017-01-01 | × |
| Highest_Level_of_Educati.. | is selected | Master's degree | × |
| Highest_Level_of_Educati.. | is selected | Ph.D. | × |

Figure 6.9: The data filters for our Foreign Applicant Report.

When we Save our report, Laserfiche Forms will redirect us to the **Reports** section. When we refresh that page, we should see our report. When we click it, it should give us a list of the applicants we are looking for.

6.4 Process Optimization

In this section, we will cover several advanced options available in Process Diagrams. These will allow us to optimize our process and meet Acme Industries' performance benchmarks.

6.4.1 Deadlines using Timer Events

One of the more common methods of making sure users complete their tasks on time is to configure deadlines. These deadlines can be one-time (“if deadline passes, re-route the flow to another branch”) or repeatable (“do X until the activity is finished”).

In our Job Applications, we have a potential bottleneck: sometimes department managers take a long time to review job applications that are sent to

them by the recruiters. This slows the entire process down and results in potentially awkward scenarios, such as an applicant being contacted twice within the span of two weeks as the second department approves their job application and wants to interview them. Ideally, each department should have a deadline for their task.

Let's go to our Process Diagram and right-click on one of the department review User Tasks, and select "Attach timer event". This will attach a yellow clock to the User Task, which we can start configuring by double-clicking it. We will name it "1-day deadline".

We can configure the wait option in three different ways: we can have it wait until a specific date, wait for a certain period of time, or wait based on a Field Variable. All three of these options are useful under different circumstances. In our case, we will configure our timer to wait for 1 day, and exclude weekends.

Then, we configure what should happen when the deadline passes. We have two options:

1. We can choose to interrupt the attached activity, which will end the User Task and cause the process to continue down the default outflow path. This can be useful if we aren't interested in waiting for a response from the department if they don't review the job application within 1 day.
2. We can choose to not interrupt the attached activity, which allows us to repeat the deadline timer until the user completes the task.

We will go with the first option. Then, we will make a few modifications to the outflow paths. This part is a bit tricky, so refer to the screenshot below (I attached my timer object to the bottom-right corner of the User Task):

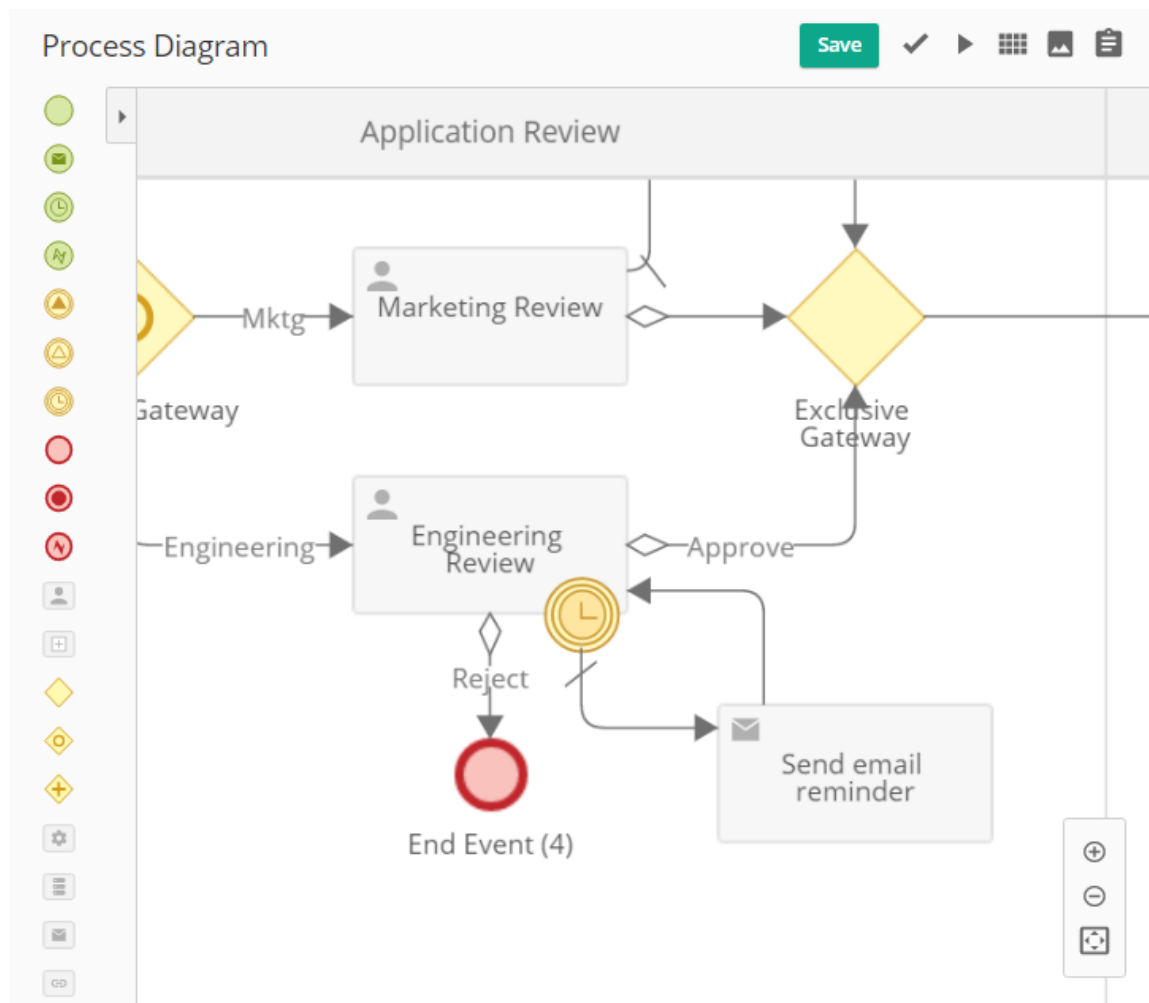


Figure 6.10: Timer event configuration.

First, we have the two outflow paths coming out of the Engineering Review User Task. Both of these have the appropriate conditional expressions (e.g. “Last User Action = Reject”). Then, we have a third outflow path, this one coming out of the timer (not the User Task itself). It is set as the default outflow path and connected to an email service task, which loops back to the Engineering Review User Task.

The result of this configuration is that the user gets a certain amount of time to complete their task. If the deadline passes, they are sent an email reminder

and the task is re-assigned to them.

Box 6.2. Deadlines and reporting

While timer events are very useful, when used as deadlines they affect average durations. Specifically, in our configuration above, the User Task is interrupted when the deadline passes, then it is re-assigned to the user after the email reminder is sent. This can potentially reduce the average duration for that task, since the maximum duration the task can run for is going to be based on the deadline.

For practice, I recommend configuring deadlines for some of the other User Tasks as well. Keep in mind however that timers you add to a Process Diagram will only take affect for new process instances, so you will have to submit new job applications to test them. Also, make sure to turn them off after testing, otherwise your email inbox may get full!

6.4.2 Signal events

Running User Tasks or service tasks in a Business Process run in parallel can be useful for efficiency. However, sometimes you need events that happen in one branch to affect other branches. For example, let's say that after the recruiter screens the application and routes it to multiple departments, we want to implement a "first-come-first-served" logic whereby the first department that reviews the application and approves it prevents other departments from doing so. While you may not want to do this in a production setting, for this tutorial we will experiment with it.

In Laserfiche Forms, this is accomplished using Signal Events. There are two kinds: Signal Throw and Signal Catch. The idea is that signals thrown from one part of the process are caught in other parts.

In our Process Diagram, we will insert a Signal Throw Event on the Approve branch of the Engineering Review User Task. Then we will configure it and add a new broadcast signal named "Engineering Approved". This

signal will be thrown when the Engineering department approves the application.

We also need to catch this signal. Specifically, we want it to interrupt any reviews that may be going on for this applicant in the Sales or Marketing departments. We can do this by right-clicking each User Task and selecting “Attach signal event”, which we then configure to listen for the Engineering Approved signal we are throwing from the Engineering approved branch.

Similar to the timer event, Signal Catch Events can be configured to interrupt or not interrupt their attached activity. In our scenario, we will interrupt it, then create an outflow path from the Signal Catch Event and end the process. When the “Engineering Approved” signal is caught by this Signal Catch Event, the outflow path leading out of it will be followed, and it will end the process. Below is how I have mine configured:

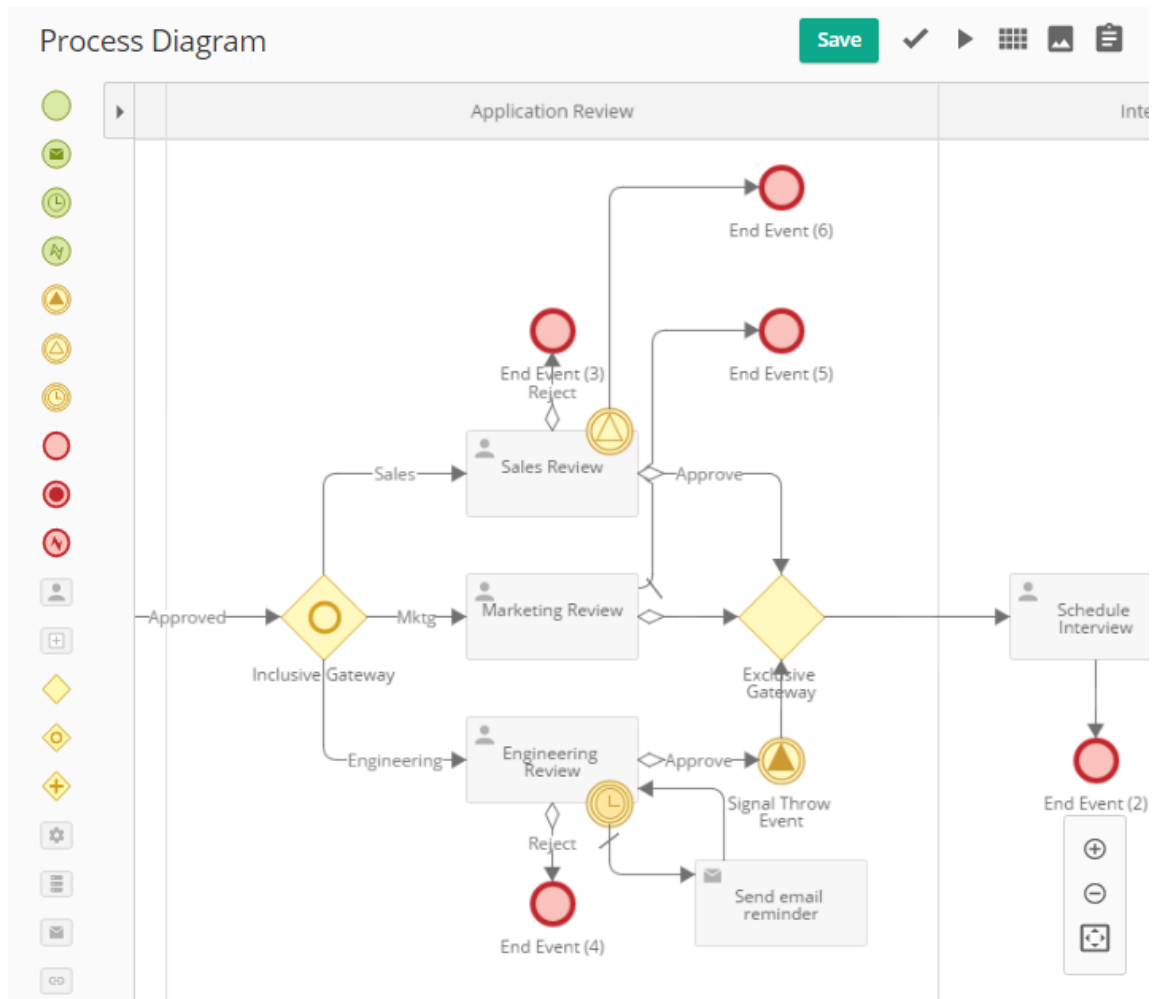


Figure 6.11: Configuring Signal Throw and Signal Catch Events.

The Signal Throw Event located on the Engineering Review User Task’s approved outflow path throws the “Engineerign Approved” signal, which is caught by the Signal Catch Event attached to the Sales Review User Task. This ends that task if it is currently active and follows the outflow path coming out of the Signal Catch Event.

Signal Events can be extremely powerful. Not only can they increase process efficiency (by making sure tasks are not unnecessarily performed), but they can also improve collaboration between process participants. That said,

they also add a certain level of complexity to the process, which can make it more difficult to change later. So use them sparingly and only when needed.

6.5 Conclusion

With the addition of the Timer and Signal events to our Process Diagram, we are now finished with the Job Application Business Process. This one ended up being significantly more complex than the previous two we created, although hopefully not overwhelmingly so.

We have covered all the major features of Laserfiche Forms 10.2. Despite this, there is still a lot to learn! As a first step in this process, this section contains some suggestions for further learning.

6.5.1 Guide to further resources

The web provides a wealth of resources to learn just about any topic. When it comes to Laserfiche, there are several.

- [Laserfiche Online Help Files](#): Laserfiche provides very detailed documentation about all of their products, including Laserfiche Forms. When I want to learn about a specific feature, this is where I start. Help files contain everything from system requirements for various Laserfiche components to descriptions and usage examples of specific features.
- [Laserfiche Certified Professional Program](#): The Laserfiche CPP program offers online courses to become officially certified on various Laserfiche-related topics. Each CPP offers a comprehensive product manual as well as videos.
- [Laserfiche Empower](#): The annual Laserfiche Empower conference is an incredibly well-organized, week-long training conference. It brings together Laserfiche users and experts from all over the world and features

classes taught by Laserfiche engineers about the very products they develop. Registrations usually open around early Summer, so make sure to check the page regularly for various early-bird specials.

In addition, below are some free resources for further learning about complimentary technologies we have briefly covered in this tutorial:

- [Learn HTML & CSS: Part I](#): This free course teaches HTML and CSS, which are considered the foundations behind all web pages, including forms built using Laserfiche Forms. Learning these would be a great start.
- [Learn JavaScript](#): This free course teaches the fundamentals of programming with JavaScript, which can help you take your Laserfiche Forms to the next level.
- [Learn jQuery](#): This free course teaches jQuery, an extremely popular JavaScript library that makes it a breeze to add interactivity and animations to web pages. jQuery is already included in Laserfiche Forms - in fact we used some in this chapter!
- [Learn SQL](#): This free course teaches the fundamentals of Structured Query Language, more commonly known by its acronym SQL. Learning SQL can be very beneficial, not just in your Laserfiche career, but elsewhere.

So go ahead... dive in and see how deep the rabbit hole goes!