

FlexFoundation Blueprints:
Passport Application Approval System

FlexFoundation™ 

 **ezgov.**

transforming the business of government™

www.ezgov.com

FlexFoundation Blueprints: Passport Application Approval System

Version 3.0

BTM - Blueprint - 033104 - 3.0

Copyrights

© 2004, EzGov, Inc. All rights reserved. Any unauthorized use, reproduction, adaptation, distribution, display, or disclosure of this material, or any part thereof, is strictly prohibited and is a violation of the Copyright Laws of the United States (17 U.S.C. Section 101 et.seq.)

Trademarks

EzGov, the EzGov logo, and EzGov FlexFoundation are registered trademarks of EzGov, Inc. in the United States and/or other countries. FlexFoundation and *transforming the business of government* are trademarks of EzGov, Inc. in the United States and/or other countries.

All other brand names and product names used in this document are trade names, service marks, or registered trademarks of their respective owners.

Notices

This product (software and/or documentation) is furnished under a License/Purchase Agreement and may be used only in accordance with the terms of such agreement. Reverse engineering of the software is prohibited.

EzGov, Inc. reserves the right to make changes in specifications at any time and without notice. The information furnished by EzGov, Inc. in this document is believed to be accurate and reliable, but is not warranted to be true in all cases.

Links and addresses to Internet resources are tested and inspected thoroughly prior to release, but the ever-changing nature of the Internet prevents EzGov, Inc. from guaranteeing the content or existence of the resource. When possible, the reference contains alternate sites or keywords that could be used to acquire information by other methods.

Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by EzGov, Inc.

U.S. GOVERNMENT RESTRICTED RIGHTS. If the software and software documentation are licensed for use by the United States or for use in the performance of a United States government prime contract or subcontract, Licensee agrees that the software and software documentation are delivered as "commercial computer software" subject to FAR 12.212 and DFARS 227.7202. The use, duplication, and disclosure of the software and software documentation by the Department of Defense shall be subject to the terms and conditions set forth in the separate license agreement. All other use, duplication and disclosure of the software and software documentation by the United States shall be subject to the terms and conditions set forth in the separate license agreement. The licensor is EzGov, Inc., 101 Marietta Street, N.W., Atlanta GA 30303.



Table of Contents

	How to Use This Guide	5
	Who should read this documentation	5
	Organization of this documentation	5
	Related documentation	6
	Conventions used in this documentation	6
Chapter 1	Project Overview	
	The Passport Application Approval System	7
	What is the PAAS?	8
	The Runtime Framework and Development Environment	9
	Business Transaction Manager	9
	Developer Studio	10
	The Project Artifacts	10
	The Process used to create the PAAS	11
Chapter 2	Initialization	
	Initializing the PAAS	14
	Initializing Business Transaction Manager Components	15
Chapter 3	The Model	
	The Form Schemas	16
	Login Form Schema	17
	SS11 Form Schema	17
	SS91 Form Schema	17
	The Form Elements	18
	Defining the Form Elements	18
	Defining the Response Values	19
	The Business Rules	20

Organizing the Rule Sets	20
Defining the Display Formats	21
Defining the Validation Formats	21
Defining the Rule Scripts	22

Chapter 4 User Interface Considerations

Application Requirements	24
Understanding Form Schema Management	24
Understanding Design Requirements	25
Understanding application flow	26
Defining Pages for the PAAS	26
Defining Page Elements for the PAAS	27
Defining Response Choices for the PAAS	27
Defining Styles for the PAAS	28
The Page Styles	29
The Header Styles	32
The Footer Styles	33
The Navigation Bar Styles	33
The Page Element Styles	34
The Page Specifications for the PAAS	36
Login Page Specifications	36
My Work List Page Specifications	38
Manage Form SS91 Page Specification	40

Chapter 5 Application Flow

Form Schema management	43
Implementing the events in the application flow	44
Naming conventions	45
Identifying Repeated Events	45
Application flow through the PAAS	47
Getting to the Login Page (2.0)	47
Login Page (2.0)	48
Create Username Page (2.1)	50
Create Username button on the Create Username Page	51
My Work List Page (3.0)	53
Process buttons on the My Work List Page	54
Form SS11 - Pages 1 through 4 (3.1 - 3.4)	55
Manage Form SS91 Page (3.5)	60

	Form SS91 - Pages 1 and 2 (3.6 - 3.7)	64
	Assign Passport Application Page (3.8)	67
	Approve Passport Application Page (3.9)	68
	Calling the Events from Buttons	69
	Navigation Properties	69
	Default Chain Properties	70
	Default Authentication Interaction	70
	Default Error Interaction	71
	Implicit Chain Properties	71
Chapter 6	External Data Integration	
	Data Integration in the PAAS	72
	External Data Interactions in the PAAS	73
	Get External Data Interaction	73
	Map External Data Interaction	75
	Update External Data Interaction	75
Chapter 7	Advanced Presentation Techniques	
	Advanced presentation techniques in the PAAS	77
	Dynamic Response Values	77
	Application Status dynamic Response Values list	78
	Select Officer Dynamic Response Values List	80
	Repeating form data	82
	Repeating form data in the PAAS	82
Appendix A	Logging into the PAAS	
	Existing User Credentials for the PAAS	84
	Creating New User Credentials for the PAAS	85
	Valid Employee IDs in the PAAS	85
Appendix B	Use Cases for the PAAS	
	Project Artifact: Login Use Case	87
	Use Case Title: Login	87
	Project Artifact: Approve Passport Application Use Case	90
	Use Case Title: Approve Passport Application	90
Appendix C	Screenflow for the PAAS	

	Screenflow	95
Chapter 8	Wireframes for the PAAS	
	2.0 Login	97
	2.1 Create Username	98
	3.0 My Work List	99
	3.1 Form SS11, 1st page	100
	3.2 Form SS11, 2nd page	101
	3.3 Form SS11, 3rd page	102
	3.4 Form SS11, 4th page	103
	3.5 Manage Form SS91	104
	3.6 Form SS91 1st page	105
	3.7 Form SS91, 2nd page	106
	3.8 Assign Application	107
	3.9 Approve Application	108
Appendix D	Business Rules for PAAS	
Appendix E	Mockup for the PAAS	
	Mockup	148



How to Use This Guide

This guide describes the design decisions and implementation process for the Passport Application Approval System, a FlexFoundation Blueprint application. It describes how the application was designed and how it was built using Developer Studio. It also provides an overview of the application, describes the project artifacts used to document the application requirements, and provides information about accessing the application.

Who should read this documentation

This guide addresses technical personnel who participate in the architecture, implementation, and planning of customizing solutions for customer environments.

This document assumes that you are familiar with the technologies listed in the System Requirements section of *Creating Business Transaction Manager Applications*.

Organization of this documentation

The information in this documentation is presented in 7 chapters:

- ❖ **Chapter 1, *Project Overview***, provides an overview of the Passport Application Approval System application, the artifacts, the process, and the development tools used to build the application.
- ❖ **Chapter 2, *Initialization***, discusses how we initialize the PAAS.
- ❖ **Chapter 3, *The Model***, provides the design considerations and implementation process for creating the Model for the PAAS.
- ❖ **Chapter 4, *User Interface Considerations***, provides the design decisions and implementation process for the Pages, Page Elements, Response Values, and Styles on the PAAS.
- ❖ **Chapter 5, *Application Flow***, discusses how we implemented navigation using interactions and the Business Rules.

- ❖ *Chapter 6, External Data Integration*, discusses how the PAAS integrates with an external database.
- ❖ *Chapter 7, Advanced Presentation Techniques* provides information about implementing dynamic dropdowns and displaying data from multiple Form Results in a table format.

Related documentation

Currently, the following documentation contains information to which you may need to refer:

- ❖ *Installing Developer Studio*
- ❖ *Administering Business Transaction Manager Applications*
- ❖ *Creating Business Transaction Manager Applications*

Conventions used in this documentation

To help you find important information, this documentation uses the following conventions:

This convention	Indicates
<i>bold italic</i>	A term included in the glossary.
boldface	The names of elements of the user interface, such as menus, buttons, and options.
bold monospaced	Text you should type exactly as shown, such as “Type shutdown -f at the prompt.”
monospaced	This style is used for code, file names, directory paths, and file listings.
NOTE	Notes that contain additional explanation.
Tip	Tips that contain shortcuts or special information.
Caution	Cautions that contain information to avoid errors.
Warning	Warnings that contain information about potentially damaging actions. Read them carefully.
See also	Cross-references that tell you where you can find additional information about a topic.
File > Save	The path to a menu command. In this example, the Save menu item on the File menu.
CTRL+v	Shortcut keys to a menu command. In this example, press and hold the Ctrl key while pressing the lowercase letter v, then release both keys.

Project Overview

The Passport Application Approval System (PAAS) is a FlexFoundation Blueprint application built on FlexFoundation™ Business Transaction Manager. It is called a blueprint application because it is a pattern that developers can follow when building their own Business Transaction Manager applications.

This guide accompanies the PAAS. It discusses functionality in the PAAS, key design decisions we made when building the PAAS, and recommended implementation patterns. The PAAS Blueprint application is available to you as a sample Project in FlexFoundation™ Developer Studio. The sample Project shows you all the files and objects used to create the PAAS. You can run the PAAS by launching it from Developer Studio, which deploys the PAAS into a pre-configured runtime environment.

This Project Overview chapter introduces the PAAS Project, discussing the development tools we used to build the Project, the artifacts we used as a basis to build the Project, and the process we used to build the Project.

The Passport Application Approval System

This section describes the PAAS, its functionality, who uses it, and how it works.

What is the PAAS?

The Passport Application Approval System allows government agency employees to process electronic applications for new passports and passport renewals. It is a fully functional sample Business Transaction Manager application that provides a working demonstration of how you can independently, quickly, and efficiently build a Business Transaction Manager application using FlexFoundation™ Developer Studio. The functionality we demonstrate in the PAAS includes:

Functionality demonstrated	Chapter
Login and user credential creation	Chapter 5 Login Functionality
Use of Styles to control the look-and-feel of application	Chapter 3 User Interface
Navigation through application pages	Chapter 4 Navigation
How we defined our Form Schemas	Chapter 2 Model
Basic workflow	Chapter 4 Application Flow
Use of Business Rules	Chapter 2 Model
Access of data in an external relational database	Chapter 4 Application Flow
Dynamic Response Choices	Chapter 6 Advanced Presentation Techniques
Repeating data rows	Chapter 6 Advanced Presentation Techniques

Who uses the PAAS?

All government agency employees with access to the PAAS have one of two roles: They are either a Passport Agent or a Passport Officer. Each user that logs into the system is identified as having one of those roles, and has access to certain functionality based on that role.

- ❖ **Passport Agent.** The Passport Agent can view and process applications for a passport that has a status of Submitted or In Progress. After reviewing a passport application, the Passport Agent can assign it to a Passport Officer.
- ❖ **Passport Officer:** The Passport Officer can view and process all applications, regardless of status. After reviewing a passport application, the Passport Officer can approve or deny it.

How does the PAAS work?

Citizens submit an application for a new passport or renewal of an existing passport via an online form called Form SS11.

When the Passport Agents or Passport Officers choose to process the application, they can view and modify all of the information that the citizen submitted on Form SS11. If the citizen's previous passport was lost or stolen, the government agency employee that reviews the application must attach a supplementary form, Form SS91, to explain in detail why the previous passport cannot be returned. The employee can add, edit, and delete one Form SS91 per application for a passport.

An application for a passport (Form SS11) has one of the following statuses:

- ❖ **Submitted** – The application was submitted to the government agency, but has not been processed.
- ❖ **In progress** – The application is being processed, and has not yet been assigned to a Passport Officer.
- ❖ **Assigned** – The application was assigned to a Passport Officer, but has not yet been approved or denied.
- ❖ **Approved** – The Passport Officer approved the application.
- ❖ **Denied** – The Passport Officer denied the application.

After the government agency employee logs into the PAAS, they reach a Page that displays their work list, which is all the applications for a passport available to them for processing. From this Page, they can select an application to process. After processing the application, they can either assign it, or approve or deny it, based on their role.

The Runtime Framework and Development Environment

This section discusses Business Transaction Manager and Developer Studio, which are the FlexFoundation products that comprise the runtime framework and development environment we used to build the PAAS.

Business Transaction Manager

Business Transaction Manager is the preferred software platform for creating thin-client government applications for revenue collection, benefits filing, and licensing applications. Business Transaction manager's unique combination of functionality provides pre-packaged components that enable delivery teams to quickly build applications that require capturing data, validating that data against Business Rules, and mapping that data to existing systems. The Business Transaction Manager framework includes the following components:

- ❖ **Rendering Engine** Dynamically assembles all application pages at runtime by merging the appropriate Styles with dynamically generated Pages to produce the user interface. Once the Page is rendered, filled in by the user, and submitted, Rendering Engine invokes Business Rules Engine, to apply Business Rules against the data.
- ❖ **Business Rules Engine** Enables the execution of Business Rules, including validations and calculations, defined in Developer Studio for Business Transaction Manager applications.
- ❖ **Interaction Manager** Handles the Interactions defined in the navigation component of your application.
- ❖ **Core Monitoring Facility** Contains the underlying software packages that provide data access to the components of all FlexFoundation applications. This includes logging, transaction count monitoring, and other services and utilities.
- ❖ **Document Repository Manager** Enables an application to store, retrieve, and delete files in the document repository.
- ❖ **PDF Generator** Allows applications to generate PDF documents from application data.

Developer Studio

Developer Studio is an integrated team-based development environment that facilitates designing, building, and testing Business Transaction Manager applications. Developer Studio is designed to simplify a complex development process by providing a series of integrated wizards and editors that facilitate rapid learning, allowing you to become immediately productive.

Developer Studio is built for team-based development. While developers are working on separate objects in parallel, Developer Studio constantly ensures consistency of each developer's work across the objects; therefore, errors are discovered and corrected before integration. Developer Studio provides direct access to all of the pre-built functionality in Business Transaction Manager, where the application architecture and runtime components for an application are already set. This enables delivery teams to get up and running quickly.

The Project Artifacts

All requirements used to create the PAAS Blueprint application are captured in project artifacts. In this guide, the term *project artifacts* means the documentation used to communicate requirements from the client to the implementation team.

These artifacts describe and demonstrate the client’s expectations of the functionality that the PAAS will provide. This can include specification of the look and feel for pages, the types of information that the PAAS must capture, and the types of users that require access to the PAAS.

You can find these project artifacts in the Appendices of this guide. The following table describes the project artifacts.

Artifact Name	Artifact Description
Use Cases	Documents the interaction of a user with the system. Each use case describes a sequence of interactions between the system and the user that yields an observable result that has value to the user. It includes the most common sequence of actions, the Basic Flow, and other alternative flows stemming from the basic flow. Two use cases were created for the PAAS.
Wireframes	Provides a visual representation for each Page in the application. This includes the text on the Page, Page Elements on the Page, and the navigational elements that are available to the user (for example, buttons). We created Wireframes for each Page in the PAAS.
Screenflows	Documents the Page flow throughout the application.
Mockup	Provides the design for a Page. This includes the specific colors used on the Page and the specific layout of text and Page Elements on the Page. Only one mockup was created for the PAAS. This mockup then provided the basis for designing all of the other Pages.
Business Rules	Documents the Business Rules for the PAAS and the associated error and warning messages. For example, it specifies the number of characters a user can enter for a Page Element and the valid characters a user can enter for a Page Element.

The Process used to create the PAAS

This section discusses the process we used to create the PAAS. We followed the process described in *Creating Business Transaction Manager Applications* to create the PAAS Blueprint application. To create the PAAS we,

Stage	Process
1	Reviewed and analyzed requirements.
2	Created the Project in Developer Studio.

Stage	Process
3	Created the Model.
4	Defined the user interface (UI).
5	Defined the application flow (navigation).

Review and analyze requirements

We started by reviewing and analyzing the requirements documented in the project artifacts. Part of the analysis process included determining:

- ❖ naming conventions for things like Page Elements and Interactions,
- ❖ the events in the application flow, and
- ❖ the number of Form Schemas to create.

These design decisions are documented in detail throughout the remainder of this guide.

Create the Project in Developer Studio

After completing the design process, we then created a Project in Developer Studio. The Project contains all the resources that make up an application.

NOTE Refer to *Creating Business Transaction Manager Applications* for more information about creating Projects using Developer Studio.

Create the Model

Next we created the Model for the application. This involved defining our Form Schemas, Form Elements, and Business Rules. We gathered most of the requirements for the Model from the Business Rules artifact. See [Chapter 3, The Model](#), for more information on the Model for the PAAS.

Define the user interface

In parallel to creating the Model, we defined the user interface for the application. First we created Pages for the application based on the Wireframes, and then associated each Page to a Form Schema. Second, we defined the Page Elements for each Page and associated them with Form Elements. We gathered most of these requirements from the Wireframes. Last, we defined Styles for the Pages and Page Elements based on the Mockup and the Wireframes.

After applying the Styles, we previewed each Page to make sure it looked the way we intended. See *Chapter 4, User Interface Considerations*, for information on how we defined the user interface (UI) for the PAAS.

Define control for navigation

Last, we defined control for the navigation through the application using Interactions. We built an Interaction Chain to provide the functionality for each event identified during our analysis of the requirements. We used the Handlers provided with Developer Studio for many of our Interactions, and we created a few custom Handlers to deal with the more complex navigation logic. See *Chapter 5, Application Flow*, for more information on how we implemented the navigation for the PAAS.

Initialization

This chapter discusses how we initialize the Passport Application Approval System.

Initializing the PAAS

We decided to initialize the Blueprint application upon startup of the servlet container so that the application is already running when the first user requests the first page of the application. To do this, we extended the `InitServlet` class in the `com.ezgov.flexfoundation` package, which is loaded on startup by the default `web.xml` web descriptor included with Developer Studio. Rather than changing the `InitServlet` directly, we opted to create an application-specific initialization class, the `BlueprintApp` class in the `com.ezgov.blueprint` package, to handle the Blueprint application initialization so that we would have more control over configuration of the application at runtime. To invoke the `BlueprintApp` class from the `InitServlet` at runtime, we included the following line in the `init()` method of the `InitServlet`:

```
BlueprintApp.getInstance();
```

This invokes the `BlueprintApp` class to initialize the Business Transaction Manager components for the application. In the `BlueprintApp` class, we externalized some configuration properties into a `BlueprintAppConfig` resource bundle so that we could override the properties at runtime with a configuration property resource file.

Initializing Business Transaction Manager Components

Within the `BlueprintApp` initialization class, we initialized the Business Transaction Manager components in the following required order:

Stage	Process
1	Initialized application logging, which enables us to report any error encountered during initialization of other components. <code>LoggingMonitor.initialize();</code>
2	Initialized String Manager, which reads in the externalized configuration properties we defined for the application. <code>configManager = StringManager.getManager ("com.ezgov.blueprint", "BlueprintConfig", Locale.getDefault ());</code>
3	Initialized the Data Manager to make it available for the Interaction Manager and Rendering Engine components because they require access to external databases. <code>DataMgrContainer container = DataMgrContainer.getInstance ();</code> <code>container.loadFromConfig (configManager);</code>
4	Initialized the Business Transaction Manager components. Example: <code>InteractionMgr.componentInit ();</code> <code>Eforms.componentInit ();</code>

The Model

This chapter discusses how we created the Model for the Passport Application Approval System. The Model contains the resources that define how your application data is validated, stored, and displayed. In the PAAS, the Model contains the data the application uses and stores as it processes passport applications. The Model also contains the Business Rules that ensure the passport application is valid and complete.

To define the Model for the PAAS, we took the following approach:

Stage	Process
1	Defined the Form Schemas.
2	Defined the Form Elements and Response Values.
3	Defined the Rule Sets.
4	Defined the Display Formats.
5	Defined the Validation Formats.
6	Defined the Rule Scripts.

The Form Schemas

For the PAAS, we created three separate Form Schemas to logically group the data the application uses or stores as it executes the passport application approval functionality. The first Form Schema captures the data used to log in the user, the second captures the data from the paper SS11 Form, and the third captures data from the paper SS91 Form.

Login Form Schema

The Login Form Schema defines the data captured and used in the PAAS through the login, create username, and user work list functionality. The data captured for this Form Schema is not stored in the Business Transaction Manager database. Instead, the PAAS uses information from this Form Schema to lookup and store information in another system (an external database) that manages user credentials for the PAAS.

We decided to create the Login Form Schema as a separate Form Schema because the login functionality is typically a good candidate for reuse across applications. The login functionality also provided a logical break in the application functionality if we wanted to have multiple developers working on Form Schemas.

SS11 Form Schema

The SS11 Form Schema defines the data captured and used in the PAAS for the paper Form SS11 and for the functionality to approve and assign a passport application. We wanted to capture and persist Form Results for the SS11 Form, so we set up a Business Transaction Manager database for the PAAS.

SS91 Form Schema

The SS91 Form Schema defines and captures the data for the paper SS91 Form, which contains information that is required if the passport applicant has lost a previously issued passport. The Passport Agent or Officer completes the SS91 Form, and always submits it with an SS11 Form.

There is a one-to-one relationship between the SS11 Form and the SS91 Form, meaning that one SS11 Form may have zero or one associated SS91 Form. The Form Results captured for the SS91 Form Schema are also persisted to the Business Transaction Manager database.

Because there is a one-to-one relationship between the SS11 Form and the SS91 Form, we could have combined these two Form Schemas; however, we chose to separate them so that we could have multiple developers working on the two separate Form Schemas simultaneously. If, instead, a Passport Agent or Officer could submit an “unlimited” number of instances of the Form SS91 with a Form SS11, we could not have combined the two Form Schemas. This is because a Form Schema must contain a fixed number of Form Elements.

NOTE Refer to *Creating Business Transaction Manager Applications*, Chapter 4, *Defining Form Schemas*, for more information on defining Form Schemas.

The Form Elements

For the PAAS, we created a Form Element for every data field in the application

- ❖ used to access or update other systems, or
- ❖ needed to store and validate.

We did this by reviewing the Wireframes and Business Rules project artifacts. For example, on Wireframe 2.0 (the Login Page), we determined that we needed three Form Elements, one for the username, one for the password, and one for the application language. The first two data fields are used to validate that the user should have access to the PAAS, and the last one is used to determine the language in which the application is presented to the user.

Defining the Form Elements

For each Form Element in the PAAS, we defined the data type, Validation Format, Display Format, and the maximum response length. Additionally, we defined whether a response is required.

We were able to reuse Form Elements that captured information on one Page and displayed the information on another Page within the same Form Schema. For example, the applicant's Last Name is captured on the Form SS11 - Page 1 of 4 (`ss11page1`) and displayed on the Assign Passport Application (`assign`) and Approve Passport Application (`approve`) Pages within the SS11 Form Schema. For this, we created the `last_name` Form Element and associated it to both Page Elements on the `assign` and `approve` Pages.

For Form Elements that capture the same type of data, and for which the same validation is applies, we created one Form Element and copied it as needed. For example, in the SS11 Form Schema there are two sets of address fields (the 'mail to' address fields and the 'permanent' address fields). We created one address Form Element for each line of the address and then copied and renamed these Form Elements. This allowed us to reuse the Form Elements.

Following is an example of the attributes we defined for a Form Element created in the SS91 Form Schema. This Form Element captures information about the efforts made to recover a lost or stolen passport. See Question 13 on Wireframe 3.7 (Form SS91, 2nd Page).

Attribute	What we entered in Developer Studio
Name	<code>effort_to_recover</code>
Data type	String
Validation Format	AlphaNumSpaceCommaApostHyphPer0to255

Attribute	What we entered in Developer Studio
Message Severity (<i>for the Validation Format</i>)	ERROR
Message to display to user (<i>for the Validation Format</i>)	%INVALID_EFFORTS_TO_RECOVER
Is response required from the user?	Yes
Message Severity (<i>if a response is required</i>)	ERROR
Message to display to user (<i>if a response is required</i>)	%ENTER_EFFORTS_TO_RECOVER
Encrypt response	No

We did not define the message text associated with each attribute in Developer Studio because all user messages for the PAAS are contained in a resource bundle. We did this to provide an example of how you would define messages for a localized application. In Developer Studio we entered a code associated with the actual message text in the resource bundle. The % symbol tells Developer Studio that the information is a code, not the actual message text. You can view the message resource bundles in the webroot directory of the Blueprint project in Developer Studio.

Defining the Response Values

For the PAAS, we created Response Values for the Form Elements that require a static, predetermined set of possible values. To create the Response Values for the application, we reviewed the options associated with the multiple-choice questions defined in the Wireframes (drop-down and radio button questions).

Most of the Response Values are the same as the Response Choices defined for the user interface. However, when we wanted to store a shorter value in the database, we created Response Values that are different than the Response Choices. For example, question 3 on Wireframe 3.1 contains options that allow the user to select the gender of the passport applicant. To implement this, we created the following two Response Values for the Gender Form Element in the SS11 Form Schema:

Attributes of the 'Male' Response Value

Attribute	What we entered in Developer Studio
Name	gender
Response Value	m

Attributes of the 'Female' Response Value

Attribute	What we entered in Developer Studio
Name	gender
Response Value	f

The PAAS contains two Form Elements that require dynamic Response Values: Application Status (`app_status`) in the Login Form Schema and Passport Officer (`selectOfficer`) in the SS11 Form Schema. We did not define Response Values for these Form Elements in Developer Studio because they are dynamically defined at runtime. See [Chapter 7, *Advanced Presentation Techniques*](#), for more information on the dynamic Response Values implemented in the PAAS.

NOTE Refer to *Creating Business Transaction Manager Applications*, Chapter 4, *Defining Form Schemas*, for more information on defining Form Elements.

The Business Rules

Business Rules represent the core business logic of an organization. They validate data captured in an application and can help determine the application flow. You can create Business Rules to address any subject matter, from simple comparisons to complex tax computation. In the PAAS, we defined our Business Rules by first creating Rule Sets, then defining the Validation Formats, and then defining the Rule Scripts. The PAAS does not require Display Formats.

Organizing the Rule Sets

For the PAAS, we created two Rule Sets that organize our Business Rules by type of rule. You can create one Rule Set that includes all of your Business Rules for an application, or you can create multiple Rule Sets organized by Form Schema, type of Business Rule, or team structure. The following table shows the structure of the Rule Sets in the PAAS project:

Rule Set	Description
PassportRules	Contains all Display Formats and Validation Formats for the application.
PassportRegistration	Contains all of the Rule Scripts created for the application.

Defining the Display Formats

A Display Format specifies a text pattern for displaying data to the user of your application. For the PAAS, we reviewed the requirements defined in the Business Rules to determine if any Display Formats were needed for the application. We determined that we did not need to create any Display Formats.

Defining the Validation Formats

A Validation Format is a regular expression—a series of character classes, character escapes, metacharacters, and constructs, used to validate or modify data. For the PAAS, we reviewed the Business Rules artifact and determined which syntax validations we could implement using Validation Formats. In general, we implemented Validation Formats for the Business Rules that validate the format of the data entered by the user (i.e. those elements that could only contain letters).

For example, we created the `CitizenNum` Validation Format to validate that a response entered for the `citizen_id_number` Form Element in the SS11 Form Schema can only contain numbers and hyphens.

Attributes of the `CitizenNum` Validation Format:

Attribute	What we entered in Developer Studio
Name	CitizenNum
Regular expression	[0-9-]{0,11}

We created several reusable Validation Formats and applied them to multiple Form Elements. For example, we created one Validation Format to validate the data captured for all questions in the application that ask the user to enter a “day”. For example, question 5 on Wireframe 3.1 asks the user to enter a date of birth.

The regular expression created for all the “day” Form Elements verifies that the response entered contains only numeric characters, and that the value is between 01 and 31.

Attributes of the `Day` Validation Format:

Attribute	What we entered in Developer Studio
Name	Day
Regular expression	(0[1-9] 1[0-9] 2[0-9] 3[0-1])?\$

After we created each Validation Format, we applied it to the appropriate Form Elements in the application.

Defining the Rule Scripts

Rule Scripts can define a wide range of Business Rules for syntax validation, basic value checking, transformations, navigation logic, and complex calculations. For the PAAS, we created Rule Scripts for the more complex Business Rules defined in the Project Artifacts.

For example, we created the Rule Script, `Validate_PassNum`, to validate that the passport number captured in the `applicant_passport_num` Form Element matches the number entered in the `recent_passport_number` Form Element. Following is the `Validate_PassNum` script.

```
SS11Passportnum = locals().get("Blueprint_form_ss11_recent_passport_num_"
+ Blueprint_form_ss91_form_ss11_formresultid)
if(Blueprint_form_ss91_applicant_passport_num != None):
    if(Blueprint_form_ss91_applicant_passport_num != SS11Passportnum):
        _SCRIPT_RESULT = 'PASSPORT_NUMS_NOT_EQUAL'
```

If a user enters a different passport number for the `applicant_passport_num` Form Element, the system displays the error message for `PASSPORT_NUMS_NOT_EQUAL`.

After creating the `Validate_PassNum` Rule Script, we applied it to the `applicant_passport_num` Form Element in the SS91 Form Schema. Because the rule script contains a Script Result, we assigned a Message Severity and entered a code that references the actual message text in the messages resource bundle.

Attributes for the `Validate_PassNum` Rule Script:

Attribute	What we entered in Developer Studio
Form Element name	<code>applicant_passport_num</code>
Rule Script name	<code>Validate_PassNum</code>
Script Result	<code>PASSPORT_NUMS_NOT_EQUAL</code>
Message Severity	<code>ERROR</code>
Message to display to users	<code>%PASS_NUM_DOESNOT_MATCH_SS11</code>

We created several reusable Rule Scripts (`validate_format` and `validate_spouse`) for the PAAS based on the Business Rules defined in the Project Artifact. For all Business Rules that performed the same validation we were able to create one Rule Script and apply it to multiple Form Elements.

For example, we created the `Validate_Spouse` Rule Script to validate the responses for several Form Elements. For these Form Elements, the `Validate_Spouse` Rule Script specifies that if a user selects **Yes** for the “Have you ever been married?” question in Form SS11, then a response is required for the Form Element. If no response is entered for the Form Element, a Script Result is assigned to the Form Element. The following code sample shows the `Validate_Spouse` Rule Script:

```
if(Blueprint_form_ss11_married == 'yes'):
    if(_TARGET == None):
        _SCRIPT_RESULT = 'MISSING_VALUES'
```

This Rule Script first determines if the response to the `married` Form Element is Yes. If the response is Yes, then the Rule Script determines if the user entered a response for the `_TARGET` Form Element, which is the Form Element to which this Rule Script is applied. If no response is captured for the `_TARGET` Form Element, then the Rule Script assigns the `MISSING_VALUES` Script Result.

We applied the `Validate_Spouse` Rule Script to the following Form Elements in the SS11 Form Schema:

```
spouses_first_name      spouses_birth_year
spouses_middle_name     spouses_citizen
spouses_last_name       marriage_day
spouses_birth_place     marriage_month
spouses_birth_day       marriage_year
spouses_birth_month     widowDivorce
```

We then defined the appropriate attributes for the `MISSING_VALUES` Script Result for all the Form Elements to which we applied the Rule Script.

NOTE Refer to *Creating Business Transaction Manager Applications, Chapter 5, Working with Business Rules*, for more information on defining Business Rules.

User Interface Considerations

This chapter describes how we implemented the user interface for the Passport Application Approval System. A user of the application interacts with the user interface, which displays information to the user and captures information from the user. To define the user interface for the PAAS we created Pages, Page Elements, Response Choices, and navigation controls (for example, buttons) in Developer Studio. We also created Styles to control the appearance of the user interface.

To implement the user interface for the PAAS, we took the following approach:

Stage	Process
1	Reviewed the Form Schemas and how they are managed.
2	Defined all Pages, Page Elements, and Response Choices.
3	Determined the Pages, Page Elements, and Response Choices that required custom functionality.
4	Defined the Styles for all Pages, Page Elements, and navigation controls.

Application Requirements

To define the user interface for the PAAS we first had to understand the organization of the Model, the design requirements defined in the project artifacts, and the application flow.

Understanding Form Schema Management

Before defining the Pages, Page Elements, and Response Choices for the PAAS, we reviewed the Form Schemas that make up the application to understand how they work together. There are three Form Schemas in the PAAS: the Login Form Schema, the SS11 Form Schema, and the SS91 Form Schema. See [Chapter 3, *The Model*](#), for more information on Form Schema management in the PAAS.

- ❖ The Login Form Schema contains Form Elements associated with three Pages defined in the Wireframes: the Login Page, the Create Username Page, and the My Work List Page. The Pages created in the Presentation for these Wireframe Pages are associated with the Login Form Schema.
- ❖ The SS11 Form Schema contains the information in a passport application submitted electronically by a citizen. All of the Form Elements created for this Form Schema map to seven Pages defined in the Wireframes: Form SS11 Page 1 of 4, Form SS11 Page 2 of 4, Form SS11 Page 3 of 4, Form SS11 Page 4 of 4, Manage Form SS91, Assign Passport Application, and Approve Passport Application. The Pages created in the Presentation for these Wireframe Pages are associated with the SS11 Form Schema
- ❖ The SS91 Form Schema is conditionally required based on information in the SS11 Form Schema, and is always associated with an SS11 Form Schema. The Form Elements defined in this Form Schema map to two Pages defined in the Wireframes: Form SS91 Page 1 of 2 and Form SS91 Page 2 of 2. The Pages created in the Presentation for these Wireframe Pages are associated with the SS91 Form Schema.

It is important to understand the relationship between the Form Schema and the Presentation portion of your application.

Every Page that you create in Developer Studio is associated with a Form Schema, most Page Elements are associated with a Form Element, and every Response Choice is associated with a Response Value. The Form Elements and Response Values determine how the data captured in the Page Elements and Response Choices are validated and, if persisted to a Business Transaction Manager database, how the value is stored.

Understanding Design Requirements

The design requirements defined in the Wireframes and Mockups specify the layout of the Pages and the appearance of the application.

For the PAAS, we used the Wireframes and Mockups to determine which Pages had the same basic layout. Based on our review, we determined that we needed five Page Styles, and that two of those Page Styles require custom functionality in the Style. The My Work List (`user_worklist`) Page includes a table that contains repeating groups of data and the Manage Form SS91 Page contains additional button logic. More information about how we determined that we needed five Page Styles, and how we implemented the custom functionality is provided later in this chapter.

Next we reviewed the Page Elements defined on each Page in the Wireframes to determine if we needed to create any custom Page Element Styles. We found that there are several Page Elements that are displayed on the same line, and that most of the Page Elements are numbered. To correctly display these Page Elements, we needed to create custom Page Element Styles based on the sample Styles included with Developer Studio.

We then reviewed the header area of each Wireframe Page, and found that the information contained in the header area is the same on every Page. As a result, we determined that we could add the header information to each Page Style instead of creating Header Styles.

We also reviewed the footer area of each Wireframe Page, and found that there is no information in the footer area of the Pages. So we did not create Footer Styles for the PAAS.

Understanding application flow

Application flow describes the users' route, or navigation, through the application. We define navigational controls for the user interface, such as buttons, hyperlinks, hyperlinked images, and tabs, to allow the user to navigate through the application. For the PAAS, we reviewed the Wireframes and Mockups to determine how a user navigates through the application.

We found that most navigation through the PAAS is done via buttons, so we then created button images that represent the navigation controls defined in the Wireframes. We also created a custom Navigation Bar Style to control the placement and appearance of the buttons and to determine the action that occurs when the user selects the button. More information on defining the navigation controls for the PAAS is included in the Styles section of this chapter.

Defining Pages for the PAAS

Pages display information and collect information from a user. A Page consists of Page Elements, Response Choices, instructional text, a header area, a footer area, and navigation controls. You can define Page Styles to control the look and feel of the Page.

To define Pages for the PAAS, we took the following approach:

Stage	Process
1	Created the Pages in Developer Studio based on the Pages defined in the Wireframes.
2	Associated each Page to the appropriate Form Schema in the Model.
3	Defined all attributes except for the Page Styles associated with the Pages.
4	Created the Page Styles.
5	Applied the Page Styles to the appropriate Pages.

For example, we created the Login (`login`) Page in Developer Studio and associated it to the Login Form Schema defined in the Model. In the Page Editor, we defined the text to appear in the browser title bar as "2.0 - Login" and the Style attribute 6 (border text) as "Welcome to the Passport Application Approval System". We then created the `Login.jsp` Page Style for the Page, and applied it to the Page.

Defining Page Elements for the PAAS

Page Elements prompt the user for information and present information to a user on a Page.

To define Page Elements for the PAAS, we took the following approach:

Stage	Process
1	Defined Page Elements for each field specified in the Wireframes.
2	Associated each Page Element to a Form Element in the appropriate Form Schema.
3	Defined the necessary attributes for the Page Elements.
4	Created the Page Element Styles.
5	Applied the Page Element Styles to the appropriate Page Elements.

Special Considerations

When defining your Page Elements, you should consider how you want to implement the instructional text. You can include instructional text on a Page in a text block Page Element, a read-only text field Page Element, or add it directly to the Page Style. We determined the approach for including the instructional text in the Page by reviewing the Wireframes. If the instructional text appeared on a Page that we knew required a unique Page Style, we added the text to the Page Style. For Pages that were not unique (e.g., Form SS11 pages 1 through 4), we added read-only text field Page Elements to the Pages containing the instructional text.

For example, we added the instructional text that appears on Form SS91 Page 1 of 2 (`ss91page1`) as a read-only text field Page Element to the Page (`instructionalText`).

Defining Response Choices for the PAAS

Response Choices are options that you can add to multiple-choice Page Elements (dropdowns and radio buttons) that offer a user one choice out of a group of mutually exclusive choices.

To define Response Choices for the PAAS, we took the following approach:

Stage	Process
1	Reviewed the Wireframes to determine the Response Choices to create.
2	Created Response Choices in Developer Studio for the multiple-choice Page Elements with static, pre-defined options.
3	Associated the Response Choices to the appropriate Response Values defined in the Model.

Special Considerations

If the Response Values for a multiple-choice Page Element are determined dynamically at runtime, we did not add Response Choices to the Page Element in Developer Studio. There are two Page Elements in the PAAS that contain Response Values that are generated at runtime:

- ❖ the Application Status (`app_status`) Page Element on the My Work List Page and
- ❖ the Select the officer (`selectOfficer`) Page Element on the Assign Passport Application Page.

The Response Values for the Application Status (`app_status`) dropdown list are based on the user's role, Passport Agent or Passport Officer, which the system determines when the user logs into the PAAS. The Select the officer (`selectOfficer`) dropdown list is populated at runtime with all the Passport Officers registered with the PAAS. See [Chapter 7, *Advanced Presentation Techniques*](#), for more information on how we implemented dynamic Response Values in the PAAS.

Defining Styles for the PAAS

Styles are individual JSP files that contain the code to render the formatting for a Page, Page Element, Header, Footer, and Navigation Bar. The Styles provided with Developer Studio are used to render Pages in HTML. To support the operations performed in Styles, such as initializing a Style or initiating an action, the Styles contain Business Transaction Manager JSP tags. These tags are delivered in a library with Developer Studio, and consist of Java code that executes when a tag is used in a Style.

Once we defined the Pages, Page Elements, and Response Choices for the PAAS, we determined how these components appear in the application by defining Styles. We took the following approach to define the Styles:

Stage	Process
1	Defined Styles for the Pages.
2	Defined Styles for the Header.
3	Defined Styles for the Footer.
4	Defined Styles for the Navigation bar.
5	Defined Styles for the Page Elements.

The Page Styles

The Page Style defines the layout of a Page, identifying the location in the Page where Headers, Footers, Navigation Bars, and Page Elements are placed.

For the PAAS, we reviewed the sample Styles included with Developer Studio and determined that we needed to modify the sample Styles to create the following five custom Page Styles:

- ❖ `Login.jsp`
- ❖ `Register.jsp`
- ❖ `RepeatingGroups.jsp`
- ❖ `ManageForm.jsp`
- ❖ `FormPages.jsp`

We did not create a unique Style for every Page in the application because we were able to reuse the `FormPages.jsp` Page Style for multiple Pages. We applied this Page Style to all Pages with the same appearance and general layout.

All of the custom Page Styles we created included the header information for the Page. The Page Styles all also use Style attribute 6 to define the border text to be displayed on the Page. Each Page Style defines where the border text should appear on the Page. Using this Style attribute allowed us to apply the same Page Style to multiple Pages. For example, we defined Style attribute 6 as "Form SS11- Page 4 of 4" for the `ss11page4` Page.

Creating the ManageForm.jsp Page Style

We created the ManageForm.jsp Page Style for the Manage Form SS91 (form_manage_SS91) Page. The ManageForm.jsp Style contains logic that determines which buttons to display on the Page. If a Form SS91 already exists in the passport application, a user will be presented with the Edit and Delete buttons. If a Form SS91 does not exist, and is required, a user will be presented with the Add button. The ManageForm.jsp Style also includes the Close Passport Application button, which saves the application data and returns the user to the My Work List (user_worklist) Page. In addition, the ManageForm.jsp Style contains instructional text to display on the Page.

The ManageForm.jsp Style contains the following code that displays the Add button if a Form SS91 has not been created:

```
<table cellpadding="3" cellspacing="0" border="0" bgcolor="#666666">
<% String add = formResult.getResponse( "form_ss91_formresultid" ).getResponse();
if ("".equals(add) || add == null) { %>
  <tr>
    <efoms:setInputName var="add" save="true" validate="true" complete="false"
interaction="Add_managess91"/>
    <td><input type="image" src="ezsuite/efoms/styles/images/custom/
btn_add_ss91.jpg" height="17" name="<efoms:expr value="$add"/>" border="0"
width="99" alt="Add Form SS91"></td>
  </tr>
</table>
```

At the top of the Style we included the `<%@ taglib uri="/efoms" prefix="efoms" %>` directive which is required in order to use the tags in the Business Transaction Manager Tag Library.

Additionally, we included the `<pageStyleInitialize>` tag, which must exist in a Page Style in order to render a Page and to use other tags included in the Page Style. This tag initializes a collection of variables that are made available to each Style.

The `<% String add = formResult.getResponse("form_ss91_formresultid").getResponse(); if ("".equals(add) || add == null) { %>` tag accesses the Form Result ID of the Form SS91 to determine if a Form SS91 already exists. If the Form Result does not exist, the PAAS displays the Add button. If the Form Result does exist, the PAAS displays the Edit and Delete buttons.

Creating the Login.jsp Page Style

We created the Login.jsp Style for the Login Page in the PAAS. As defined in the Wireframes, the Login Page required a layout unique to the rest of the application. To create the Login.jsp Style, we modified the packaged Sample_LoginPage.jsp Style.

The `Login.jsp` custom Style includes

- ❖ the text to display on the Page,
- ❖ the HTML buttons used for navigation, and
- ❖ the code used to render the Page.

We included the buttons used for navigation in this Page Style because the Log In button is positioned in the middle of the Page. To include this unique button placement in the Style created for navigation would require additional custom code; therefore, it was easier to create a custom Page Style that includes the buttons.

Creating the Register.jsp Page Style

We created the `Register.jsp` Page Style for the Create Username (`create_username`) Page. The `Register.jsp` Page Style is similar to the `login.jsp` Page Style, except that it does not include the buttons used for navigation. To create the `Register.jsp` Style, we modified the `Login.jsp` Page Style by removing the buttons and modifying the instructional text included in the Style. The buttons used for navigation on the Create Username Page are included in the Navigation Bar Style, `Blueprint_ImageNavbar.jsp`.

Creating the RepeatingGroups.jsp Page Style

We created the `RepeatingGroups.jsp` Page Style for the My Work List (`user_worklist`) Page, which includes a table displaying read-only summary information about all the passport applications that are available to the user for processing. The data in the table is pulled from multiple Form Results from the SS11 Form Schema. To create the `RepeatingGroups.jsp` Page Style, we modified the packaged Page Style, `Sample_SummaryResultsPage.jsp`. This sample Page Style contains the presentation tags that support the creation of a list containing repeating rows of data from multiple Form Results.

To create the `RepeatingGroups.jsp` Page Style, we first created a table with one row and seven columns. We then copied the logic that displays the repeating form data from the `Sample_SummaryResultsPage.jsp` Page Style and added it around the table. In each column, the name of the Page Element to be displayed is included. In the table, we included the presentation tags copied from the sample Page Style to define the content of the table. See [Chapter 7, *Advanced Presentation Techniques*](#), for more information on how we implemented this table.

In addition to the table, the `RepeatingGroups.jsp` Page Style includes the text to display on the My Worklist (`user_workload`) Page, and the buttons used for navigation. The Filter and Process buttons on this Page are images, instead of HTML, which allows us more control over the appearance of the buttons.

Creating the `FormPages.jsp` Page Style

We created the `FormPages.jsp` Page Style for all of the remaining Pages in the PAAS. To create this Page Style, we modified the `TableBorderPage.jsp` Page Style provided with Developer Studio. The `FormPages.jsp` Page Style creates a border around all of the contents on the Page. In addition, it contains the information in the header area of the Page, and the Close Passport Application button, which saves the application data and returns the user to the My Worklist (`user_worklist`) Page.

We included the Close Passport Application button in the `FormPages.jsp` Page Style because it is displayed on all Pages to which we applied this Page Style.

However, we did not include the primary navigation buttons—Cancel, Back, and Next—in the Style because some of the Pages do not contain all of these buttons. For example, the Assign Passport Application Page contains only the Back and Cancel buttons. Instead, we defined these navigation controls in a navigation bar Style, `Blueprint_ImageNavbar.jsp`, and applied this Style to the Page. The `Blueprint_ImageNavbar.jsp` Style contains logic that determines which buttons to display for each Page.

We did not include the instructional text in this Page Style because the Pages that this Style was created for do not include the same instructional text. For the instructional text on these Pages, we added text block Page Elements to the Page.

The Header Styles

The Header Style controls the text, images, or other elements that are displayed in the header area at the top of a Page. The Page header could contain a Page title or a banner image. Header Styles are most often used for applications that have the same Page layout, but require different headers based on where the user is in the application. For example, if an application contains six Pages with identical layouts and each Page requires a different logo in the header, you would create one Page Style and six different Header Styles.

For the PAAS, we did not use Header Styles; instead, we defined all header images and text in the Page Styles. We included the header information in the Page Styles because all Pages in the PAAS contain the same header information.

The Footer Styles

The Footer Style controls the text, images, and other elements that display in the footer area at the bottom of a Page. For example, you could choose to display a copyright notice or the date when the Page was last modified. There is no footer information in the Pages in the PAAS, so we did not create Footer Styles.

The Navigation Bar Styles

The navigation bar provides controls to

- ❖ move to the next Page,
- ❖ move back to the previous Page,
- ❖ access online help,
- ❖ validate application data,
- ❖ save and submit information, and
- ❖ exit an application.

The Navigation Bar (navbar) Style dictates the application actions, layout, and formatting of the navigation bar and its buttons. Application actions allow the Style to issue processing directives to the Rendering Engine and to specify that a user action calls an Interaction Chain. Using the navigation bar Style is optional.

Defining the `Blueprint_ImageNavbar.jsp` Navigation Bar Style

For the PAAS, we created one custom Navigation Bar Style, `Blueprint_ImageNavbar.jsp`. Every Page in the PAAS uses the custom Navigation bar Style except for the Login Page and the My Work List Page. The custom `Blueprint_ImageNavbar.jsp` contains buttons and the logic for displaying the Next, Back, Cancel, and Create Username buttons on the Pages where it is applied. We modified the `Sample_ImageNavbar.jsp` delivered with Developer Studio to create the `Blueprint_ImageNavbar.jsp` Style.

The following code samples are included in the `Blueprint_ImageNavbar.jsp` Style, and define the Interaction Chains that are run if a user selects the Cancel or Create Username buttons on this Page. If a user selects the Cancel button, the `<eforms:setInputName var="cancel" save="false" validate="false" complete="false" interaction="Cancel_createuser"/>` tag specifies that the `Cancel_createuser` Interaction will run.

If a user selects the Create Username button, the `<eforms:setInputName var="create" save="false" validate="true" complete="false" interaction="Create_createuser"/>` tag specifies that the `Create_createuser` Interaction should be run. The next code sample shows how the Style determines which buttons to display on the Page based on the Page name.

```
<eforms:choose>
<!-- Create Username Page -->
  <eforms:when test="$curPage.pageAlias == 'createuser'">
    <eforms:setInputName var="cancel" save="false" validate="false"
complete="false" interaction="Cancel_createuser"/>
    <td><input type="submit" value="Cancel" class="blacktext" name="<eforms:expr
value="$cancel"/>"></td><br>
    <td></td>
    <eforms:setInputName var="create" save="false" validate="true"
complete="false" interaction="Create_createuser"/>
    <td><input type="submit" value="Create Username >" class="blacktext"
name="<eforms:expr value="$create"/>"></td>
  </eforms:when>
<!-- End Create Username Page -->
```

The Page Element Styles

The Page Element Styles define the appearance and attributes of a Page Element when it is rendered on a Page. The characteristics of the Page Element that the Page Element Style control include the tab index assigned to the Page Element and the placement of label text in relation to the Page Element.

For the PAAS, we modified the sample Styles included with Developer Studio to create custom Page Element Styles for the Page Elements. We needed to create custom Page Element Styles because the label for most Page Elements contains a number (see Wireframe 3.1 for an example).

The sample Styles do not include a column for a number to appear before the Page Element label. In addition, we needed to remove the background color custom attribute defined in the sample Styles because we defined the background color for all Page Elements in the Page Style.

Following is an example of the custom Page Element Styles we created.

Creating the multifiields/TextField.jsp Page Element Style

We created the custom `multifiields/TextField.jsp` Page Element Style for use with all date of birth, city/state, and name fields, which display several Page Elements on one line. For each field in the example below we added a separate Page Element to the Page and applied the `multifiields/TextField.jsp` Style to each.

1. Applicant Name:

For the First name (`first_name`) and Last name (`last_name`) Page Elements, we used Custom attribute 5 in Developer Studio to specify whether the Page Element should appear at the beginning of the line or at the end of the line. In this example, we entered “begin” in Custom attribute 5 for the first name, and “end” for the last name.

The following code sample comes from the `multifiields/TextField.jsp` Style and defines how the PAAS displays a Page Element if Custom attribute 5 is set to "begin". This code creates a new row within the table and includes the Page Element at the beginning of the row. Note that there is no `</tr>` tag closing the row. This code sample also defines the column that includes the number to appear before (preamble) the Page Element.

```
<eforms:if test="$curPageElement.styleAttribute5 == 'begin'">
<tr>
  <td valign="top">
    <eforms:choose><eforms:when test="$((curPageElement.preamble != null) &&
    (curPageElement.preamble != ''))"><span class="text"><eforms:expr
    value="$curPageElement.preamble" escaperesult="false"/></span></eforms:when>
      <eforms:otherwise>&nbsp;</eforms:otherwise>
    </eforms:choose></td>
  <td valign="top"><span class="text"><eforms:expr value="$curPageElement.question"
  escaperesult="false"/></span></td>
  <td valign="top">
    <table cellpadding="0" cellspacing="0" border="0">
      <tr>
</eforms:if>
```

If we define Custom attribute 5 in Developer Studio as "end", the following code in the Style is applied to the Page Element. This code displays the Page Element at the end of the row and also includes the `</tr>` tag to close the row.

```
<eforms:if test="$curPageElement.styleAttribute5 == 'end'">
  </tr>
  </table>
</td>
</tr>
<tr>
```

```
<td colspan="3"></td></tr>  
</eforms:if>
```

The Page Specifications for the PAAS

This section includes examples of Pages created in the PAAS and information about attributes for each Page. This section provides the Page specifications for the following Pages:

- ❖ Login Page
- ❖ My Work List Page
- ❖ Form SS11, Page One

Login Page Specifications

The Login (`login`) Page is the first Page presented to a user in the PAAS. On this Page, a registered user can log in to the PAAS and access the My Work List Page, or non-registered users can access the Create Username Page.

In Developer Studio, we first defined the Login Page, and then added the Username and Password Page Elements. We applied the `login.jsp` Page Style to the Page.

login_user Page Element

We added the `login_user` Page Element to the Login Page to capture the username for a user that is already registered with the application.

Attributes for the `login_user` Page Element:

Attribute	What we entered in Developer Studio
Name	login_user
Type	text field
Style	ezgov/rowLayout/twoColumnRow
Label Text	Username:
Associated Form Schema	login

Attribute	What we entered in Developer Studio
Associate a Form Element	login_user
Custom Attribute (box width)	15
Custom attribute (assistive technology)	User Id

login_password Page Element

We added the `login_password` Page Element to the Login Page to capture the password for a user that is already registered with the application.

Attributes for the `login_password` Page Element:

Attribute	What we entered in Developer Studio
Name	login_password
Type	text block field
Style	ezgov/rowLayout/twoColumnPasswordRow
Label Text	Password: (case sensitive)
Associated Form Schema	login
Associate a Form Element	login_password
Custom attribute (box width)	15
Custom attribute (assistive technology)	Password (case sensitive)

My Work List Page Specifications

The My Work List Page (`user_worklist`) allows a user to access passport applications, and modify or submit them for approval. The My Work List Page contains a dropdown list of the statuses a passport application can have. The user can select a status from the dropdown list to filter the passport applications that appear in the work list. The statuses that appear in the dropdown are based on the user's role, Passport Agent or Passport Officer, which the system determines when the user logs into the system.

- ❖ A Passport Agent can view and process applications for passports that have a status of Submitted or In Progress.
- ❖ A Passport Officer can view and process all applications, regardless of status.

We first created the `user_workload` Page in Developer Studio, and then added one Page Element (`app_status`). We did not add Response Choices for the `app_status` Page Element. Instead, we created custom code to retrieve the choices from a database at runtime. We applied the `RepeatingGroups.jsp` Style to this Page, which contains the Page layout information, the instructional text, the Filter and Process buttons, and the code to populate the data in the work list.



app_status Page Element

Below are the attributes for the `app_status` Page Element. The PAAS dynamically generates the options for this dropdown at runtime, so we did not create static Response Choices in Developer Studio.

Attributes for the `app_status` Page Element:

Attribute	What we entered in Developer Studio
Name	<code>app_status</code>
Type	dropdown
Style	<code>ezgov/fieldLayout/inputField</code>

Attribute	What we entered in Developer Studio
Associated Form Schema	login
Associate a Form Element	app_status

Manage Form SS91 Page Specification

The Manage Form SS91 (`manages91`) Page allows a user to add, edit, and/or delete a Form SS91. The Page Style for this Page contains custom logic to determine which buttons should be presented to the user. If a Form SS91 is required, but does not exist, the button to add a Form SS91 is presented. If a Form SS91 exists, the buttons to edit or delete the Form SS91 are presented.

We first created the `manages91` Page in Developer Studio, and then added two Page Elements, `trackingNumber` and `manageInstructions`. We applied the `ManageForm.jsp` Page Style to this Page, which contains the Add, Edit, and Delete buttons and the Create Passport Application button. We applied the navigation bar style, `Blueprint_ImageNavbar.jsp` to the Page, which contains the Back and Cancel buttons.



The screenshot shows the Passport Application Approval System interface. At the top, there is a logo of a globe and the text "Passport Application Approval System" and "A Blueprint for Business Transaction Manager Applications". On the right, there is a "Welcome" message and a "Sign Out" link. The main content area is titled "Manage Form SS91" and contains the following information:

- Tracking Number:** 3013
- Close Passport Application** button
- Text: "Form SS91 is required for passport applications where a previously issued passport is lost or was stolen."
- Text: "A citizen may bear only one valid or potentially valid passport at a time, except as otherwise authorized by the government. Therefore it is necessary to submit Form SS91 with an application for a new passport when a previous valid or potentially valid passport cannot be presented. Form SS91 must be used to set forth in detail why the previous passport cannot be presented."
- This application for a new passport requires Form SS91**
- Edit Form SS91** button
- Delete Form SS91** button
- Cancel** button
- Back** button

trackingNumber Page Element

We added the Tracking Number (`trackingNumber`) Page Element to the Manage Form SS91 (`form_manage_ss911`) Page in order to display the tracking number for the passport application. This text field only displays the tracking number and does not capture information from a user.

Attributes for the `trackingNumber` Page Element:

Attribute	What we entered in Developer Studio
Name	<code>trackingNumber</code>
Type	text field
Style	<code>ezgov/fieldLayout/inputField</code>
Associated Form Schema	<code>form_ss11</code>
Associate a Form Element	TrackingNumber
Custom attribute (read-only)	Yes
Custom attribute (background color)	<code>ffffff</code>

The tracking number is used for display purposes only, so we entered “Yes” in the Read-only custom attribute in Developer Studio to make the Page Element read-only.

manageInstructions Page Element

We added the `manageInstructions` text block Page Element to the Page, which contains the instructional text to appear at the top of the Page.

Attributes for the `manageInstructions` Page Element:

Attribute	What we entered in Developer Studio
Name	<code>manageInstructions</code>
Type	text block
Style	<code>custom/instructionalText</code>

Attribute	What we entered in Developer Studio
Label text	<pre>Form SS91</pre> <p>is required for passport applications where a previously issued passport is lost or was stolen.

A citizen may bear only one valid or potentially valid passport at a time, except as otherwise authorized by the government. Therefore it is necessary to submit Form SS91 with an application for a new passport when a previous valid or potentially valid passport cannot be presented. Form SS91 must be used to set forth in detail why the previous passport cannot be presented.</p>
Associated Form Schema	form_ss11
Associate a Form Element	manageInstructions

Application Flow

This chapter describes the design decisions and implementation considerations for implementing the application flow for the Passport Application Approval System. The application flow refers to the sequence of actions and changes in application state that occur as a user executes a process in an application. In the PAAS, the application flow is event-driven, meaning that the application is inactive until the user or some other system prompts the start of an event via an action or an input. The event may result in a change to information in an external database, the display of a new Page in the application, or the execution of Business Rules.

To implement the application flow in the PAAS, we took the following approach:

Stage	Process
1	Reviewed the Form Schemas and how they are managed.
2	Implemented the Events in the application flow.
3	Updated the buttons in the application to call the correct Chain Properties.
4	Set the Navigation properties.

Form Schema management

Before defining the application flow of the PAAS, we had to understand the Form Schemas that make up the application and how they work together. There are three Form Schemas in the PAAS: the Login Form Schema, the SS11 Form Schema, and the SS91 Form Schema.

- ❖ The Login Form Schema is transient, which means that no Form Results from this Form Schema are saved to the Business Transaction Manager database. This Form Schema contains three Pages: the Login Page, the Create Username Page, and the My Work List Page. The Form Result for this Form Schema is only created once per user session.

- ❖ The SS11 Form Schema contains all the information for one passport application. A user can choose to access an SS11 Form Schema from the Login Form Schema. The SS11 Form Schema is a persistent Form Schema, which means that Business Transaction Manager saves Form Results to the database.
- ❖ The SS91 Form Schema is conditionally required based on information in the SS11 Form Schema, and is always associated with an SS11 Form Schema. There is a one-to-one relationship between the SS11 Form Schema and the SS91 Form Schema, meaning that one SS11 Form Schema can only have one associated SS91 Form Schema. The SS91 Form Schema is also a persistent Form Schema.

It is important to understand the relationship between Form Schemas because your application flow must handle the creation and deletion of Form Results. It also manages the links between Form Schemas. More information on managing Form Schemas in the application flow is provided throughout the remainder of this chapter. See [Chapter 3, *The Model*](#), for information about the Form Schemas for the PAAS.

Implementing the events in the application flow

To understand how to implement the application flow for the application, we first identified the events in the application flow. An event occurs as the result of an input or action by the user or another application. In the PAAS, events are typically triggered when a user clicks a button in the application. During the event, some type of processing occurs, and there is an outcome from the event. In the PAAS, events typically result in the rendering of a new Page. We documented the PAAS events in a series of flow diagrams that are provided throughout the remainder of this section.

After identifying the events in the application flow, we took the following approach to implement the events:

Stage	Process
1	Established naming conventions for the Chain Properties and Interactions.
2	Identified the events that repeat in the application.
3	Created Chain Properties and Interactions to implement the events.

Naming conventions

The most important part of establishing naming conventions in your application is to ensure that the names are consistent and intuitive. For the application flow in the PAAS, we used two naming conventions:

- ❖ We named our Chain Properties according to the event that calls the Chain Properties and the Page on which the event occurs. For example, `create_createuser`.
- ❖ We named our Interactions according to the action they perform. For example, `Display_userworklist` Page.

Identifying Repeated Events

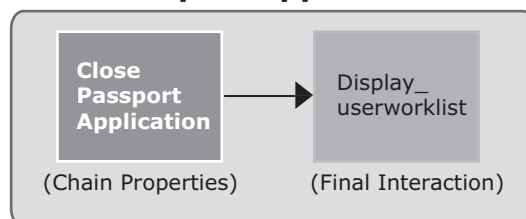
When reviewing the application flow in the PAAS, we found that some of the events repeat across several Pages of the application. It is important to identify the repeated events to determine if you can reuse an Interaction Chain in the application flow. When you reuse an Interaction Chain it reduces development time and helps to ensure that events are implemented consistently, making them easier to create and maintain. There are two events that repeat in the PAAS:

- ❖ Close Passport Application, and
- ❖ Sign Out.

Close Passport Application event

In the PAAS, the Close Passport Application button appears on every Page of the application, and always initiates the same event. When the user selects this button, the PAAS saves any changes made to the data and returns the user to the My Work List Page.

Close Passport Application Button



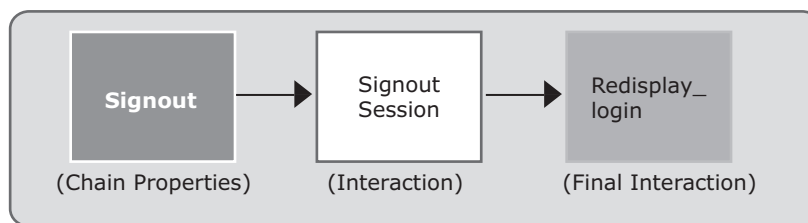
To implement the Close Passport Application event we,

Stage	Process
1	Created a Chain Properties, <code>ClosePassportApplication</code> , which starts the event.
2	Created an Eforms Request Interaction, <code>Display_userworklist</code> , which displays the My Worklist Page. The <code>ClosePassportApplication</code> Chain Properties calls the <code>Display_userworklist</code> Interaction.

Sign Out event

A Sign Out link appears on every Page of the application, performing the same action everywhere it appears in the application. When the user selects this link, the PAAS logs the user out of the application and displays the Login Page.

Sign Out



To implement the Sign Out event we,

Stage	Process
1	Created a Chain Properties, <code>Signout</code> , which starts the event.
2	Created a Custom Interaction, called <code>SignoutSession</code> , and a custom Handler, called <code>Signout</code> , which logs the user out of the application by ending the session.
3	Created a View Request Interaction called <code>Redisplay_login</code> that calls the <code>InitServlet</code> class, which in turn re-initializes the application and displays the Login Page.

Application flow through the PAAS

The application flow through the PAAS always starts at the Login Page. From that Page, users can either login to the application or create a new username.

After logging into the application, the user moves through the remaining Pages of the application:

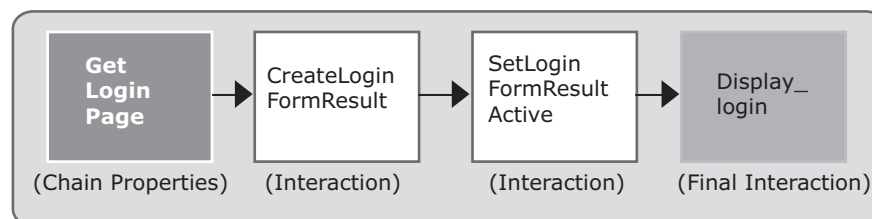
- ❖ My Work List Page, which provides a list of the SS11 Forms (passport applications) available for processing.
- ❖ SS11 Form Pages 1 through Page 4, which allow the user to view and update the four Pages of the SS11 Form.
- ❖ Manage Form SS91 Page, which allows the user to add, edit, or delete a Form SS91.
- ❖ Form SS91 Pages 1 through 2, which allows the user to edit the two Pages of Form SS91.
- ❖ Assign Page, which allows a Passport Agent to assign the passport application to a Passport Officer.
- ❖ Approve Page, which allows a Passport Officer to approve or deny the passport application.

The following sections provide detailed information on the application flow from each of these Pages. The Wireframe number for each Page is shown so you can refer to the Wireframes in the Appendix.

Getting to the Login Page (2.0)

When a user attempts to access the Login Page, the application creates a Form Result for the Login Form Schema and makes that Form Result the active Form Result.

Get Login Page



To implement the Get Login Page event we,

Stage	Process
1	Created a Chain Properties called <code>GetLoginPage</code> to start the event
2	Created the Form Result for the Login Form Schema using <code>CreateLoginFormResult</code> , which is a Create Form Result Interaction.
3	Created a Set Active Form Result Interaction, <code>SetLoginFormResultActive</code> , which sets the new Form Result as the active Form Result.
4	Created an Eforms Request Interaction, <code>Display_login</code> to direct the application to display the Login Page.

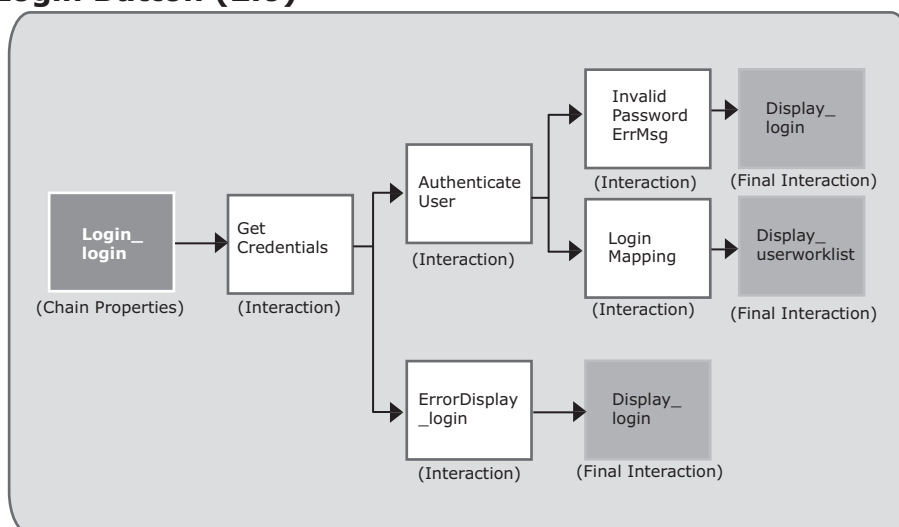
Login Page (2.0)

When the user enters a username and password on the Login Page and chooses the Log In button, the PAAS checks the information against a list of valid usernames and passwords in an external database. If the username and password exist in the database, the user is logged into the system. If the user enters invalid credentials more then two consecutive times, the user receives an error message with instructions to contact the system administrator for assistance logging into the application. There are two buttons on the Login Page, the Log In and the Create Username buttons.

Log In button

The Log In button directs the application to validate the username and password against an external database, and then logs the user into the application if the user credentials are valid.

Login Button (2.0)



To implement the Log In event we first created a Chain Properties to start the event, called `Login_login`. The `Login_login` Chain Properties calls a Get External Data Interaction, `GetCredentials`, to look up the username in an external database. For the `GetCredentials` Interaction we created a SQL query, the `GetCredentialsByUserName` query, to access the external database and return a result if the username is in the database. It also returns the role associated with the username. Any user that logs into the PAAS has a role of either “Passport Agent” or “Passport Officer”.

The `GetCredentials` Interaction defines a branch in the application flow based on the outcome of the `GetCredentialsByUserName` query.

- ❖ If no results are returned from the query, the `ErrorDisplay_login` Interaction runs. This Interaction allows us to set (and later display) an error message indicating that the query did not find the username.
The `ErrorDisplay_Login` is a Custom Interaction with configuration properties, and uses a Custom Handler, `SetMessage`, which keeps count of the number of times the user enters an invalid username. For the first two tries, the Handler directs the application to display the message associated with the error code `NOT_REGISTERED_USERNAME`. Once the user reaches the maximum number of attempts, the `SetMessage` Handler directs the application to display the message associated with the error code `INVALID_PASS_THREE`. Then the next Interaction, `Display_login`, an Eforms Request Interaction, displays the Login Page and shows the error message.
- ❖ If results are returned, the next Interaction, `AuthenticateUser`, runs. The `AuthenticateUser` Interaction is a Custom Flow Control Interaction that uses a custom Handler, `ValidLogin`, to validate the user's password, and to determine which branch to take in the application flow. The application flow differs based on whether the user's password is valid or is invalid.
 - ◆ If the password is invalid, the `ValidLogin` Handler returns the Lookup Key `invalidLogin`. This tells the application to run the `invalidPasswordErrMsg` Interaction. This is a Custom Interaction that uses a Custom Handler we already created, `SetMessage`.
 - ◆ If the password is valid, we load the Form Results for the SS11 forms for the list on the My Work List Page. See [Chapter 7, *Advanced Presentation Techniques*](#), for more information on how we generate and display this list. The `ValidLogin` Handler also checks the role associated with the user. If the role is Passport Agent, then the Response Choices in the Application Status dropdown field on the My Work List Page are dynamically set to "Submitted" and "In Progress". If the role is Passport Officer, then

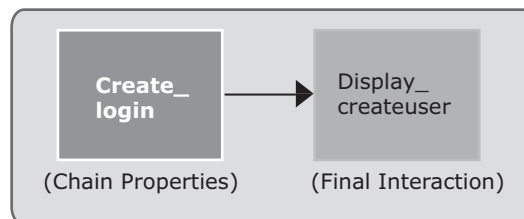
the Response Choices in the dropdown list on the My Work List Page are dynamically set to show all available statuses for a passport application. See [Chapter 7, *Advanced Presentation Techniques*](#), for more information about populating dynamic dropdowns.

The next Interaction in the chain, `LoginMapping`, is a Map External Data Interaction that stores the query results in the user session so that the application can reference the information at a later time. The final Interaction in the Chain, `Display_userworklist`, is an Eforms Request Interaction that displays the My Work List page.

Create Username button on the Login Page

When the user selects the Create Username button on the Login Page, the PAAS takes the user to the Create Username Page.

Create Username Button (2.0)



To implement the Create Username event we,

Stage	Process
1	Created a Chain Properties called <code>Create_login</code> to start the event
2	Created an Eforms Request Interaction, <code>Display_createuser</code> to display the Create Username Page.

Create Username Page (2.1)

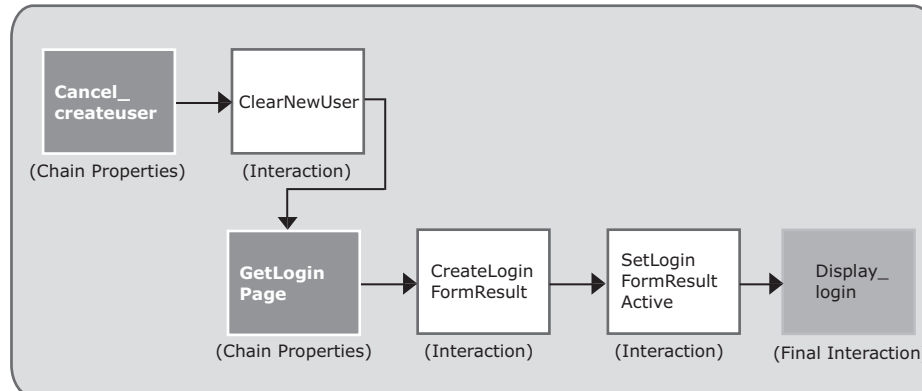
If the user chooses to create a new username from the Login Page, the PAAS displays the Create Username Page. On this Page, the user enters an employee ID, a name, a username, and a password. The PAAS checks the employee ID against a list of valid employee IDs that reside in an external database.

If the PAAS does not find the employee ID in the external database, and the user enters that ID more than three consecutive times, the PAAS issues an error message telling them to contact their system administrator for assistance in creating user credentials. If the employee ID exists, a username is created for the user, and they are returned to the Login Page. From the Login Page, they can enter their new username and password to log in to the application. There are two buttons on the Create Username Page, the Cancel and the Create Username buttons.

Cancel button on the Create Username Page

The Cancel button deletes any information entered on the Create Username Page and returns the user to the Login Page.

Cancel Button (2.1)



To implement the Cancel event on the Create Username Page we,

Stage	Process
1	Created a Chain Properties called <code>Cancel_createuser</code> to start the event
2	Created a Delete Form Result Interaction, <code>ClearNewUser</code> , to delete the Form Result created for the new user from the session. This deletes any information entered on the Create Username Page.
3	Created another Chain Properties, <code>GetLoginPage</code> . We reuse this Interaction Chain for another event later in the application flow.
4	Created a Create Form Result Interaction, <code>CreateLoginFormResult</code> , to reestablish the session and to create a new Form Result for the Login Form.
5	Created a Set Active Form Result Interaction, <code>SetLoginFormResultActive</code> , to set the Form Result for the Login Form as the active Form Result.
6	Reused the previously created Eforms Request Interaction, <code>Display_login</code> to display the Login Page.

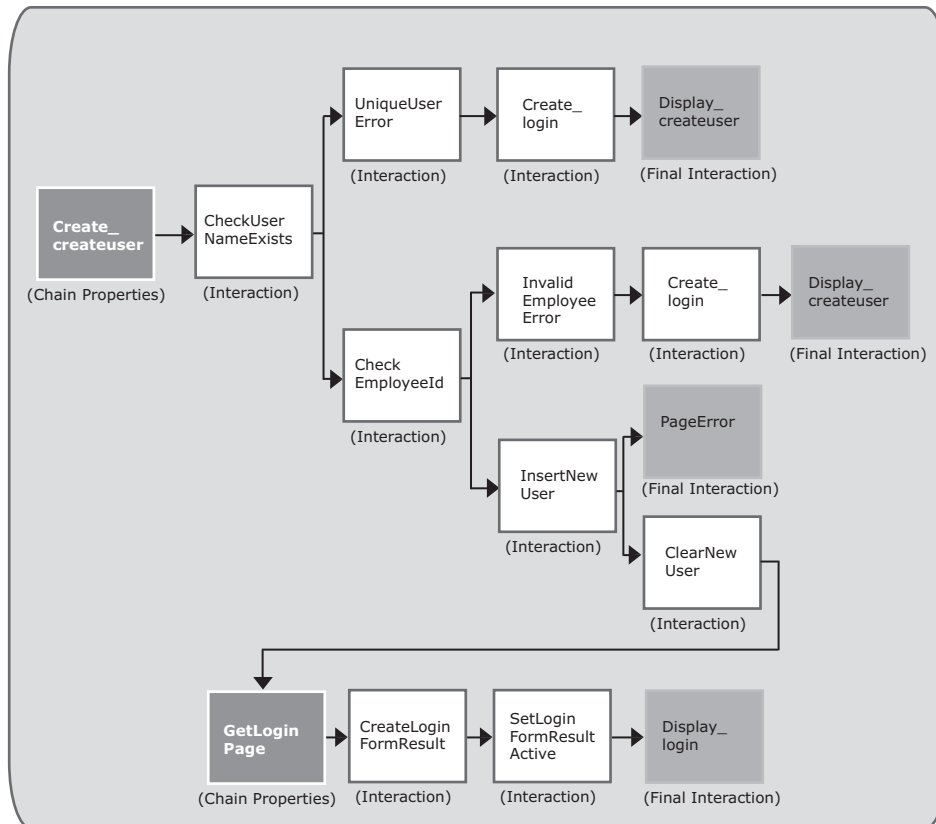
Create Username button on the Create Username Page

The Create Username button directs the application to:

- ❖ validate that the username does not already exist in an external database, and then
- ❖ validate that the employee ID exists in the database, and
- ❖ create the new username.

This process calls for a complex application flow involving several branch points.

Create Username Button (2.1)



To implement the Create Username event, we first created a Chain Properties to start the event, called `Create_createuser`. The Chain Properties calls a Get External Data Interaction, `CheckUserNameExists`, to look up the username in an external database. The `CheckUserNameExists` Interaction uses the `GetCredentialsByUserName` SQL query, the query we created for the `GetCredentials` Interaction, to access the external database and return a result if the username exists in the database. We use the `CheckUserNameExists` Interaction to ensure that there are no duplicate usernames in the database.

Like the `GetCredentials` Interaction, the `CheckUserNameExists` Interaction defines a branch in the application flow based on the outcome of the query.

- ❖ If no results are returned by the `CheckEmployeeId` query, the `CheckEmployeeId` Interaction runs. The `CheckEmployeeId` Interaction is another Get External Data Interaction that uses the `CheckEmployeeId` query to get the employee ID from the database. The `CheckEmployeeId` Interaction defines another branch in the application flow.
 - ◆ If no results are returned by the `GetOfficers` query, then the `InvalidEmployeeError` Interaction runs to set, and later display, an error message. This Interaction calls an Eforms Request Interaction, the `Display_createuser` Interaction, to redisplay the Create Username Page with the error message.
 - ◆ If the `CheckEmployeeId` query returns results, then the employee ID exists in the database, and the `InsertNewUser` Interaction runs. This is an Update External Data Interaction that creates a new user record in the database with the information that the user enters. If there are no errors creating the new username, then the `ClearNewUser` Interaction runs. This is the same Interaction we used in the Cancel Button event on the Create Username Page. If there are errors, and the update does not return results, then the default error Interaction, `PageError`, runs.
- ❖ If results are returned by the `GetCredentialsByUsername` query, then the username already exists in the database, and the next Interaction, `UniqueUserError`, runs. This Interaction sets an error message and calls an existing Interaction, `Display_createuser`, to redisplay the Page with the error message. The `UniqueUserError` Interaction is a Custom Interaction with configuration properties that use the `SetMessage` Custom Handler we created earlier.

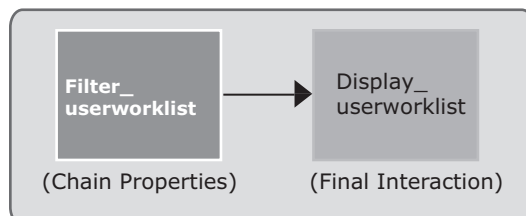
My Work List Page (3.0)

The My Work List Page displays the passport applications in a work queue that are available for the user to process. From this Page, the user can filter the list based on the status of the passport application and can choose to process a passport application.

Filter button on the My Work List Page

The Filter button filters the list of passport applications that display to the user.

Filter Button (3.0)



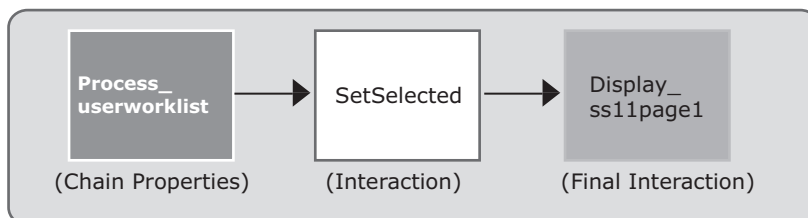
To implement the Filter event we,

Stage	Process
1	Created a Chain Properties called <code>Filter_userworklist</code> to start the event
2	Reused the previously created Eforms Request Interaction, <code>Display_userworklist</code> to display the filtered work list.

Process buttons on the My Work List Page

The Process buttons activate the Form Result for the selected SS11 Form and display the first Page of the selected form.

Process Button (3.0)



To implement the Process event we,

Stage	Process
1	Created a Chain Properties called <code>Process_userworklist</code> to start the event.
2	Created a Set Active Form Result Interaction, <code>SetSelected</code> , to set the selected Form Result as the active Form Result.
3	Created an Eforms Request Interaction, <code>Display_ss11page1</code> to display the first Page of the selected SS11 Form.

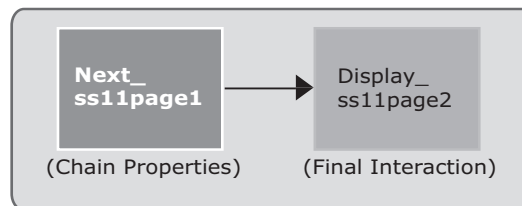
Form SS11 - Pages 1 through 4 (3.1 - 3.4)

An SS11 Form contains the information needed to issue a new passport to a citizen. The user can move linearly between the four Pages of the form using the Next and Back buttons. They also can use the Cancel button to return to the My Work List Page without saving their changes.

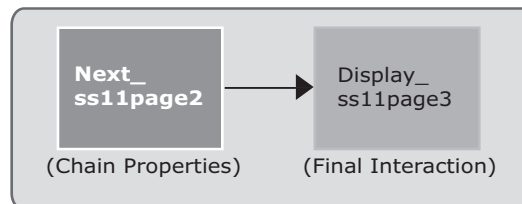
Next buttons, Pages 1-3

In the first three Pages of the SS11 Form, the Next button simply moves the user from one Page to the next.

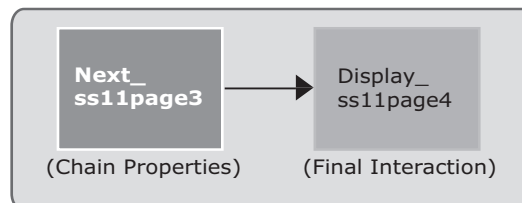
Next Button (3.1)



Next Button (3.2)



Next Button (3.3)



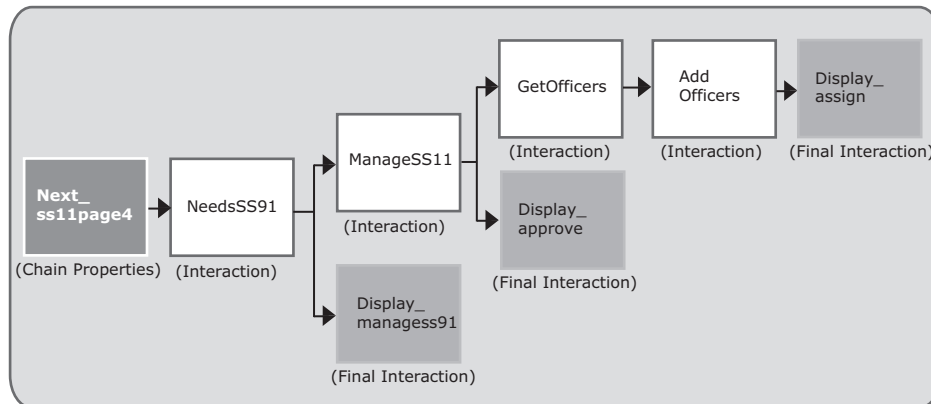
To implement the Next button events for Pages 1-3 we,

Stage	Process
1	Created a Chain Properties called <code>Next_ss11page<#></code> to start the event.
2	Created an Eforms Request Interaction, <code>Display_ss11page<#></code> to display the next appropriate Page.

Next button, Page 4

The Next button on the fourth Page is more complex. It defines a branch point in the application flow based on whether a Form SS91 is required.

Next Button (3.4)



To implement the Next button event for Page 4 we created a Chain Properties called `Next_ss11page4` to start the event. The `Next_ss11page4` Chain Properties calls the Custom Flow Control Interaction, `NeedsSS91`, to determine if a Form SS91 is required.

- ❖ If a Form SS91 is required, the `Need_SS91` Interaction returns a value of `MANAGE_SS91` and the `Display_manage91` Interaction will run. The `Display_manage91` is an Eforms Request Interaction that displays the Manage Form SS91 (`managess91`) Page.
- ❖ If a Form SS91 is not required, the `Need_SS91` Interaction returns a value of `NO_SS91_MANAGEMENT` and the `ManageSS11` Interaction runs. The `ManageSS11` determines whether this user assigns or approves this application. This Interaction uses the `ApproveOrAssign` Rule Script to determine the application flow based on the user's role.
 - ◆ If the `ApproveOrAssign` Rule Script returns a value of `Approve`, which indicates that the user has the role of Passport Officer, we use the Eforms Request Interaction, `Display_approve`, to display the Approve Passport Application Page.
 - ◆ If the `ApproveOrAssign` Rule Script returns a value of `Assign`, which indicates that the user has the role of Passport Agent. The `GetOfficers` Interaction is run, which retrieves available Passport Officers.

To find the available Passport Officers we first created a Get External Data Interaction, `GetOfficers`, to get the list of Passport Officers from an external database. This Interaction uses the `GetOfficer` query to retrieve available Passport Officers from an external database. The `GetOfficers` Interaction adds the query results to the `OfficersResultSet`.

We then created a Custom Interaction, `AddOfficers`, and a Custom Handler, `AddOfficers`, to access the `OfficersResultSet` and register the values as dynamic options. After the options are registered, we created an E-forms Request Interaction, `Display_assign`, to display the Assign Passport Application Page. The Assign Passport Application Page contains the Select Officer dropdown Page Element populated with the registered dynamic values. See [Chapter 7, *Advanced Presentation Techniques*](#), for more information on dynamically populating the Response Choices in the Passport Officers dropdown list.

NOTE Refer to *Creating Business Transaction Manager Applications* for more information on creating Rule Scripts for a Business Rule Interaction.

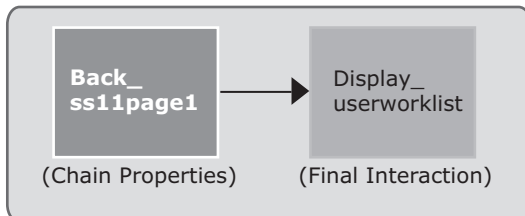
Back buttons on Form SS11

The Back buttons on each Page of Form SS11 return the user to the previous Page in the application flow. The Back button on the first Page takes the user to the My Work List Page. The Back button on the remaining Pages takes the user to the previous Page in Form SS11.

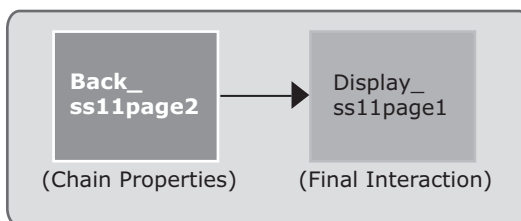
To implement the Back button event we,

Stage	Process
1	Created a Chain Properties called <code>Back_ss11page<#></code> to start the event.
2	Created the following Eforms Request Interactions to display Pages 2-4: <code>Display_ss11page2</code> , <code>Display_ss11page1</code> , and <code>Display_ss11page3</code> .
3	Reused the <code>Display_userworklist</code> Eforms Request Interaction for the first Page.

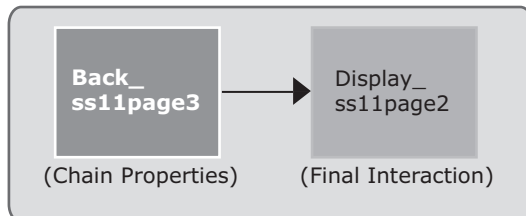
Back Button (3.1)



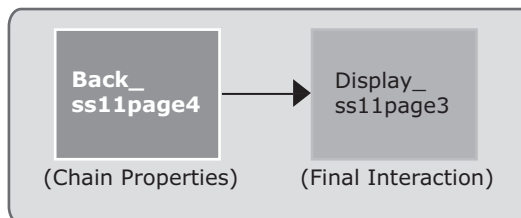
Back Button (3.2)



Back Button (3.3)



Back Button (3.4)



Cancel buttons on a Form SS11

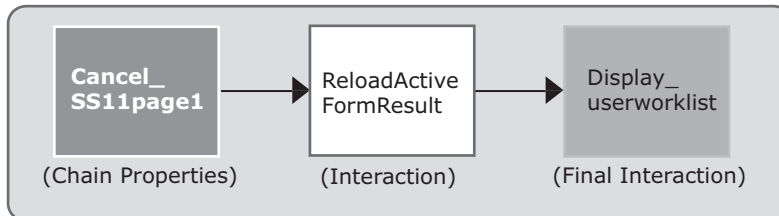
In the Form SS11, the Cancel button always takes the user back to the My Work List Page. See Cancel buttons illustration [on page 60](#).

To implement these Cancel events we,

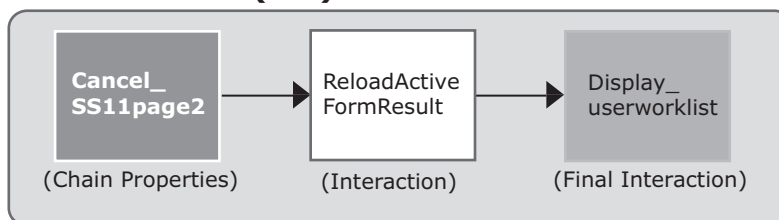
Stage	Process
1	Created a unique Chain Properties called <code>Cancel_ss11page<#></code> for each Cancel button to start the event.
2	Created a custom Interaction, <code>ReloadActiveFormResult</code> , with a custom Handler, <code>ReloadFormResult</code> Handler, which sets the Form Result for the Login Form Schema as the active Form Result.
3	Reused an Eforms Request Interaction, <code>Display_userworklist</code> to display the My Work List Page. We used this Interaction for all four of the Cancel button events.

Illustration of the Cancel button implementation on the SS11 Form Pages.

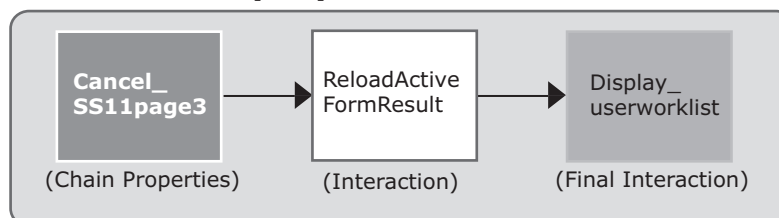
Cancel SS11 page 1



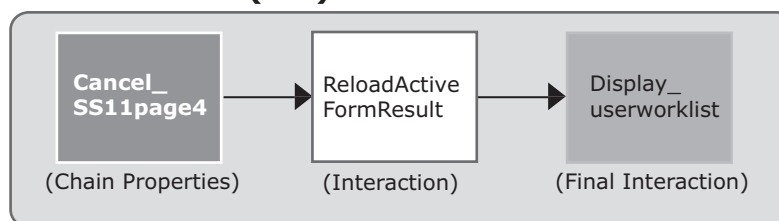
Cancel Button (3.2)



Cancel Button (3.3)



Cancel Button (3.4)



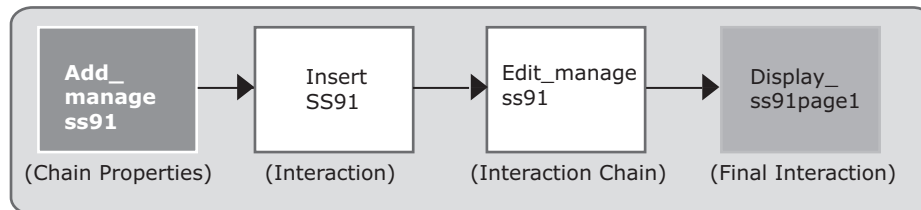
Manage Form SS91 Page (3.5)

The Manage Form SS91 Page allows a user to add, edit, or delete a Form SS91.

Add Form SS91 button

The Add Form SS91 button allows a user to add a Form SS91 to the passport application.

Add Form SS91 Button (3.5)



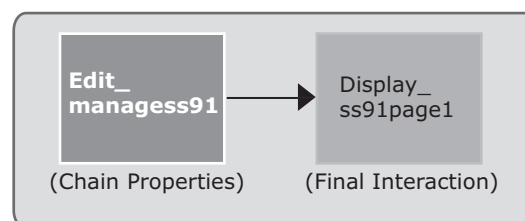
To implement the Add Form SS91 event we,

Stage	Process
1	Created a Chain Properties called <code>Add_managess91</code> to start the event.
2	Used the <code>InsertSS91</code> custom Handler to create a new Form Result for the Form Schema because this button gives the user access to Pages on a new Form Schema. The <code>InsertSS91</code> Interaction associates the new Form SS91 with the Form SS11.
3	Created another Interaction Chain to display the first Page of Form SS91. We accomplished this by calling another Interaction Chain, <code>Edit_managess91</code> , to display the first Page of the SS91 Form.

Edit Form SS91 button

The Edit Form SS91 button allows a user to view and edit a previously created SS91 Form.

Edit Form SS91 Button (3.5)



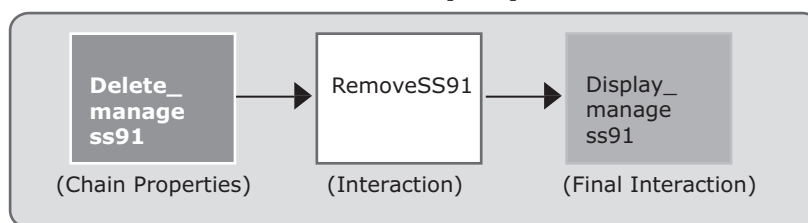
To implement the Edit Form SS91 event we,

Stage	Process
1	Created a Chain Properties called <code>Edit_managess91</code> to start the event.
2	Created an Eforms Request Interaction, <code>Display_ss91page1</code> to display the first Page of the Form SS91.

Delete Form SS91 button

The Delete Form SS91 button allows a user to delete an existing Form SS91.

Delete Form SS91 Button (3.5)



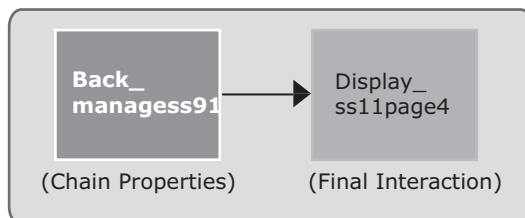
To implement the Delete Form SS91 event we,

Stage	Process
1	Created a Chain Properties called <code>DeletemanageSS91</code> to start the event.
2	Created a Custom Interaction, <code>RemoveSS91</code> , and a Custom Handler, <code>RemoveSS91</code> that removes the Form SS91 Form Result from the database and the container. It also sets the Form Result ID as null on the Form SS11.
3	Reused the Eforms Request Interaction, <code>Display_managess91</code> , to redisplay the SS91 Page.

Back button on the Manage Form SS91

The Back button on the Manage Form SS91 Page allows a user to return to the fourth Page of the SS11 Form.

Back Button (3.5)



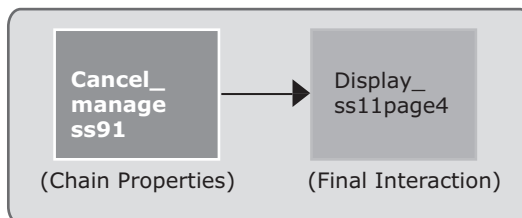
To implement the Back button event on the Manage Form SS91 we,

Stage	Process
1	Created a Chain Properties called <code>Back_managess91</code> to start the event.
2	Reused the Eforms Request Interaction, <code>Display_ss11page4</code> , to display the fourth Page of the SS11 Form.

Cancel Button on the Manage Form SS91 Page

The Cancel button on the Manage Form SS91 Page performs the same functionality as the Back button on the Manage Form SS91 Page.

Cancel Button (3.5)



To implement the Cancel button event on the Manage Form SS91 Page we,

Stage	Process
1	Created a Chain Properties called <code>Cancel_managess91</code> to start the event.
2	Reused the Eforms Request Interaction, <code>Display_ss11page4</code> , to display the fourth Page of the SS11 Form.

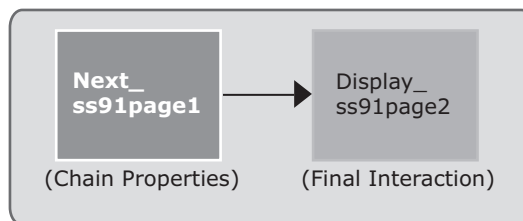
Form SS91 - Pages 1 and 2 (3.6 - 3.7)

The user is required to complete a Form SS91 if the passport applicant lost a previously issued passport. Both Pages contain Back and Next buttons that allow the user to move between Pages and a Cancel button that returns the user to the Manage Form SS91 Page.

Next Buttons on Form SS91

Unlike the Next buttons on Form SS11 that move the user from one Page to the next, the Next buttons on the Form SS91 initiate different navigation paths. The Next button on the first Page performs a simple linear flow to display the second Page in the SS91 Form.

Next Button (3.6)

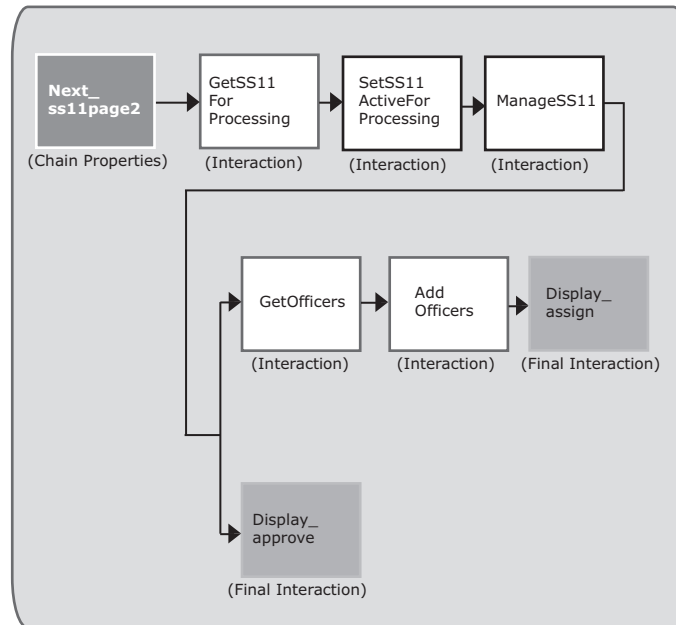


To implement the Next button event on the first Page of Form SS91 we,

Stage	Process
1	Created a Chain Properties called <code>Next_ss91page1</code> to start the event.
2	Created an Eforms Request Interaction, <code>Display_ss91page2</code> , to display the second Page of the SS91 Form.

The Next button on the second Page of the SS91 is more complex because it has to link Form SS91 with Form SS11.

Next Button (3.7)



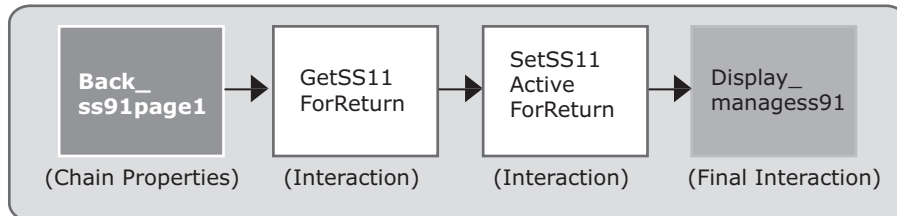
To implement the Next button event on the second Page of Form SS91 we,

Stage	Process
1	Created a Chain Properties called <code>Next_ss91page2</code> to start the event.
2	Linked the Form SS91 to the Form SS11 using a custom Interaction, <code>GetSS11ForProcessing</code> , that implements the custom Handler <code>SetSelectedFormResultAsActive</code> , to get the FormResult ID of the SS11 Form.
3	Created a Set Active Form Result Interaction, <code>SetSS11ActiveForProcessing</code> , that sets the Form Result for the SS11 Form as the active Form Result.
4	Called an existing Interaction, <code>Manage_SS11</code> , which we created for the Next button on Page 4 of the Form SS11.

Back buttons on Form SS91

The Back button on Page 1 of the Form SS91 updates the link between the Form SS11 and the Form SS91 and returns the user to the Manage SS91 Page.

Back Button (3.6)

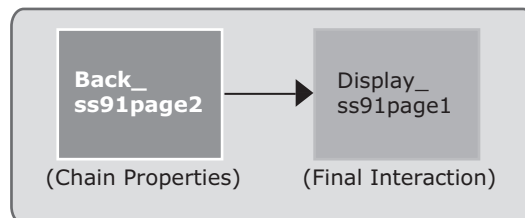


To implement the Back button event on the first Page of the Form SS91 we,

Stage	Process
1	Created a Chain Properties called <code>Back_ss91page1</code> to start the event.
2	Linked the Form SS91 to the Form SS11 by implementing a custom Interaction, <code>GetSS11ForReturn</code> , that reuses the custom Handler, <code>SetSelectedFormResultAsActive</code> , to get the Form Result ID of the SS11 Form.
3	Created the <code>SetSS11ActiveForReturn</code> , a Set Active Form Result Interaction that sets the Form Result for the SS11 Form as the active Form Result.
4	Reused the <code>Display_managess91</code> Interaction to display this Page.

The Back button on the second Page of the Form SS91 just takes the user back to the first Page of the Form SS91.

Back Button (3.7)



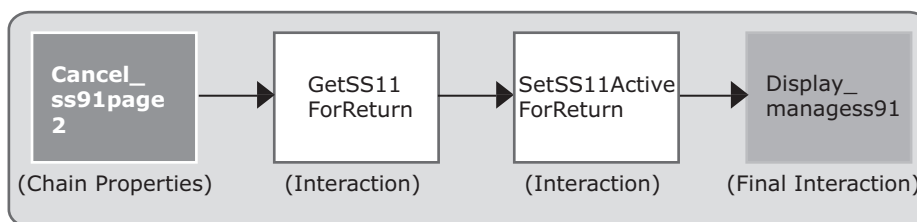
To implement the Back button event on the second Page of the Form SS91 we,

Stage	Process
1	Created a Chain Properties called <code>Back_ss91page2</code> to start the event.
2	Reused the Eforms Request Interaction, <code>Display_ss91page1</code> , to display the first Page of the SS91 Form.

Cancel buttons on Form SS91

The Cancel buttons on Form SS91 perform the same action as the Back button on the first Page of Form SS91.

Cancel Button (3.7)



To implement the Cancel button event on the Form SS91 we,

Stage	Process
1	Created a Chain Properties called <code>Cancel_ss91page<#></code> to start the event.
2	Reused existing Interactions, <code>GetSS11ForReturn</code> , <code>SetSS11ActiveForReturn</code> , and <code>Display_managess91</code> , to display the Manage Form SS91 Page.

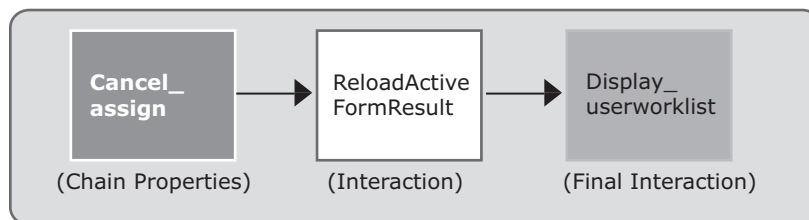
Assign Passport Application Page (3.8)

The Assign Passport Application Page allows a Passport Agent to assign the passport application to a Passport Officer. This Page contains a Close Passport Application button and a Cancel button. The Close Passport Application button saves any changes made to the data and returns the user to the My Work List page.

Cancel Button on the Assign Passport Application Page

The Cancel button returns the user to the My Work List Page.

Cancel Button (3.8)



To implement the Cancel button event on the Assign Passport Application Page we,

Stage	Process
1	Created a Chain Properties called <code>Cancel_assign</code> to start the event.
2	Created a custom Interaction, <code>ReloadActiveFormResults</code> , that reloads the SS11 Form Results prior to displaying the My WorkList Page.
3	Reused an Eforms Request Interaction to display the My Work List Page.

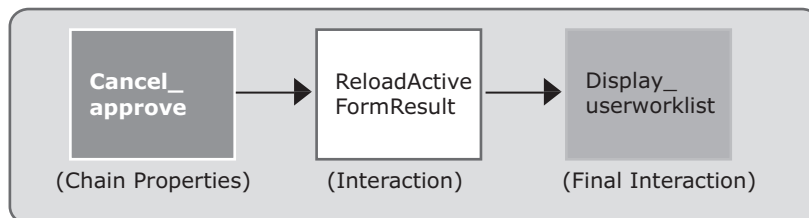
Approve Passport Application Page (3.9)

The Approve Passport Application Page allows a Passport Officer to approve or deny a passport application. This Page contains a Close Passport Application button and a Cancel button. The Close Passport Application button saves any changes made to the data and returns the user to the My Work List page.

Cancel Button on the Approve Passport Application Page

The Cancel button on the Approve Passport Application Page returns the user to the My Work List Page.

Cancel Button (3.9)



We created an externally accessible Chain Properties, `Cancel_approve`, to start the event and to call the same Interaction Chain used for the Assign Passport Application Page.

Calling the Events from Buttons

In Business Transaction Manager applications, the Style applied to the buttons contains information about which Interaction Chain to run when a user selects the button. The Style also contains information about whether to execute Business Rules before running the Interaction Chain, and whether to save the changes made to the Page. See [Chapter 4, *User Interface Considerations*](#), for more information about the Styles used in the Passport Application Approval System.

For example, we created a Custom Page Style (`Login.jsp`) for the Login Page that contains the Log In and the Create Username buttons. When the end-user enters information in the Username (`login_user`) and Password (`login_password`) fields, and chooses the Log In button, the application first runs the Business Rules defined for those two Page Elements.

Once the user enters a username and password that meets all the Business Rules, the application runs the `Login_login` Interaction Chain.

The following code sample in the `login.jsp` Style tells the application to run the Business Rules, and then to run the externally accessible `Login_login` Chain Properties:

```
<eforms:setInputName var="login" save="false" validate="true" complete="false"
  interaction="Login_login"/>
<td colspan="3"><input type="submit" value="Log In >" class="blacktext"
  name="<eforms:expr value="$login"/>"></td>
```

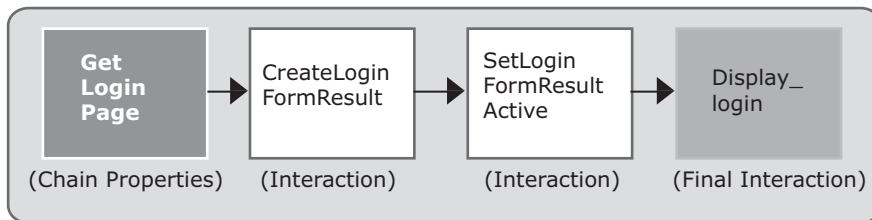
Navigation Properties

You create Navigation Properties for each project you build using Developer Studio. You only have to define these properties once for the entire project. Creating Navigation Properties includes defining a Default Error Interaction, a Default Chain Properties, a Default Authentication Interaction, and an implicit Chain Properties. The Default Error Interaction is required for the project.

Default Chain Properties

The Default Chain Properties are the Chain Properties that run if no Chain Properties are explicitly specified in the incoming request. For the PAAS, we selected the Chain Properties, `GetLoginPage`, which displays the Login Page. We chose this for the default because it returns the user to the first Page of the application, and requires them to enter a valid username and password to re-access the application.

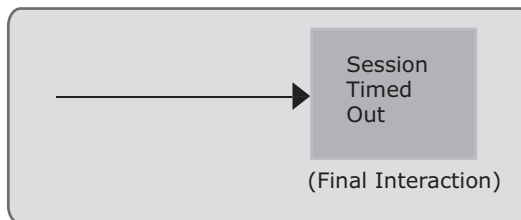
Default Chain Properties



Default Authentication Interaction

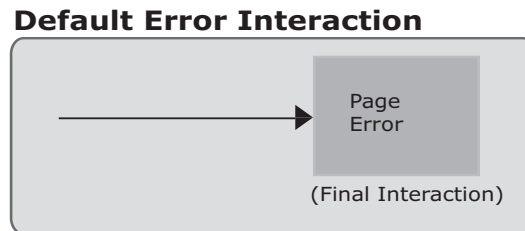
The Default Authentication Interaction is the Interaction the system executes if authentication fails for a Chain Properties for which it is required, and no Authentication Interaction is specified in the Chain Properties. For the Blueprint, we selected the `SessionTimedOut` Interaction as the Default Authentication Interaction. This Interaction displays the Session Time Out Page, and takes the user out of the application gracefully.

Default Authentication Interaction



Default Error Interaction

The Default Error Interaction is the Interaction that executes if an error occurs in the processing of an Interaction and no Error Interaction is specified for that Interaction or in the Chain Properties for the current Interaction Chain. We selected the `PageError` Interaction, which displays the System Busy Page.



Implicit Chain Properties

The implicit Chain Properties is the Chain Properties that executes at the beginning of every event before the Chain Properties specified for the event run. It contains a series of Interactions that initiate contact with the Interaction Manager (called Front Controller Interactions). For the PAAS, we used the implicit Chain Properties delivered with Developer Studio, `ImplicitChain`.

We implemented the `ValidateRequest` Interaction with a Custom Handler to check for a valid session. If no valid session exists, the `sessionExpired.jsp` is displayed.

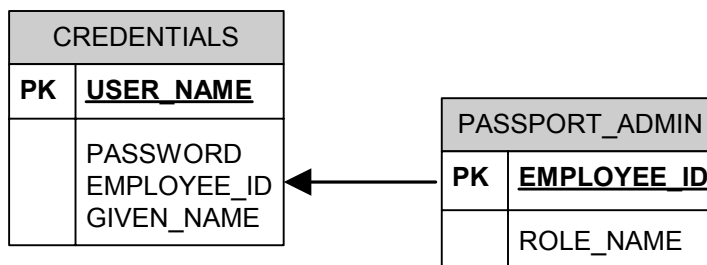
External Data Integration

Business Transaction Manager enables external data integration by providing a set of external data Interactions that allow you to:

- ❖ retrieve data from relational databases
- ❖ map the data into a format that Business Transaction Manager applications can use, and
- ❖ update data stored in an external relational database.

Data Integration in the PAAS

The Passport Application Approval System uses external data Interactions to integrate with an external relational database to manage user credentials. The database contains a username, password, employee ID, and given name for each user with access to the system. The database includes a `Credentials` table with the username and password information for each user, and a `Passport_Admin` table that associates a user role with each particular user by joining the user's Employee ID between the two tables. The following diagram depicts the `Credentials` and `Passport_Admin` tables in the database.



For the Blueprint application, the tables are implemented in an embedded database that is included with the Developer Studio installation.

External Data Interactions in the PAAS

The Passport Application Approval System uses three types of external data Interactions to interface with the user credentials database. The following table describes these Interactions.

Interaction	Description
Get External Data Interaction	Retrieves data stored in an external relational database. A Get External Data Interaction can execute a relational database query. Specifically, this Interaction can execute SELECT SQL.
Update External Data Interaction	Updates data stored in an external relational database with the information in a populated Form Result. Specifically, this Interaction can execute either INSERT or UPDATE SQL queries.
Map External Data	Maps the data in the first row of a ResultSetView into a Form Result that can be used by an application.

NOTE Refer to *Creating Business Transaction Manager Applications* for more information on the external data Interactions.

Get External Data Interaction

The Passport Application Approval System implements a Get External Data Interaction, called `GetCredentials`, to validate the user login information (username and password) against the information in the database. The `GetCredentials` Interaction compares the username that the user enters with all valid usernames in the `Credentials` table of the database.

GetCredentials Interaction

The `GetCredentials` Interaction uses the `GetCredentialsByUserName` query to retrieve the credential information for the username if it exists and if the user's Employee ID (the user's Employee ID is associated with their username) matches an Employee ID in the `Passport_Admin` table.

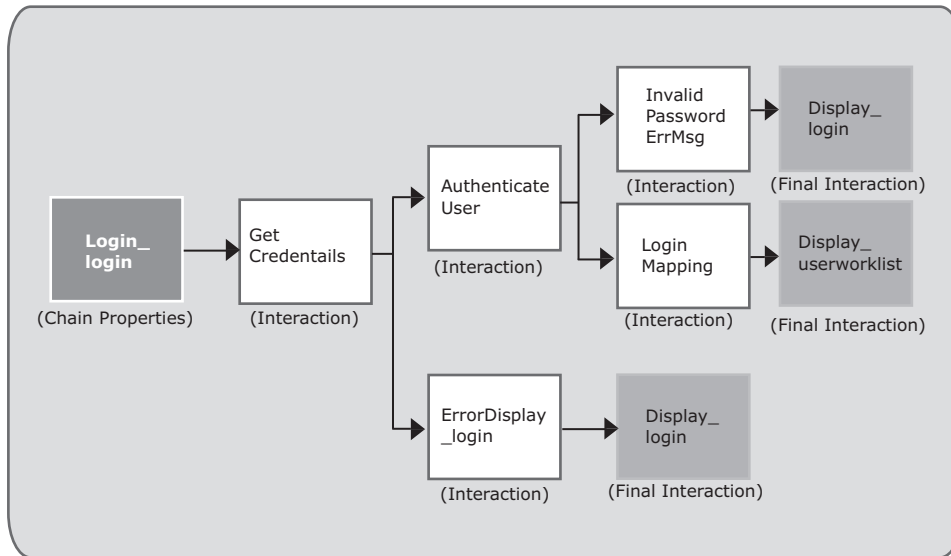
If the username does not exist in the database or if the Employee ID from the `Credentials` table does not have a match in the `Passport_Admin` table, then no records are returned by the query. Following is the `GetCredentialsByUsername` query.

```
select c.*, p.* from credentials c, passport_admin p
where user_name = ? and c.employee_id = p.employee_id
```

Implementing the GetCredentials Interaction

To implement the `GetCredentials` Interaction, we first defined a Query Argument called `name` in the Query Definition for the `GetCredentialsByUserName` query. Every question mark in a query represents a Query Argument. So for the `GetCredentialsByUserName` query, we had to define one Query Argument. We then provide a runtime value for this Query Argument by creating a Parameter Mapping that associates the Query Argument with a Form Element. For the `GetCredentials` Interaction, we created a Parameter Mapping called `name` to associate the `name` Query Argument with the `login_user` Form Element in the `login` Form Schema. When the PAAS executes the `GetCredentials` Interaction at runtime, Business Transaction Manager replaces the question mark in the `GetCredentialsByUserName` query with the value in the `login_user` Form Element.

We associated the `login_user` Form Element with the `username` Page Element on the `login` Page, so user input into that Page Element is sent as the Query Argument at runtime. The following diagram illustrates the flow of the `GetCredentials` Interaction.



The `GetCredentials` Interaction uses the results of the query to determine the next Interaction to run in the application flow. If no records are returned by the query, meaning that the PAAS did not find the username in the database, the `ErrorDisplay_login` Interaction runs. If the query does return a record, meaning that the username is found in the database, the next Interaction in the chain, the `AuthenticateUser` Interaction, runs. `AuthenticateUser` is a Custom Interaction that checks whether the password is valid for the username.

If it is valid, the next Interaction to run is a Map External Data Interaction, `LoginMapping`, which stores the query results in the application user session so that the information can be referenced later in the application. The last Interaction in the chain, `display_userworklist`, renders the My Work List Page.

Map External Data Interaction

The PAAS implements a Map External Data Interaction to map values from the user credentials database into a Form Result. This makes the user credential information available for making application flow decisions during the user's session.

LoginMapping Interaction

If the username and password are valid, as determined by the `AuthenticateUser` Interaction, the PAAS runs the `LoginMapping` Interaction to put the values from the `EMPLOYEE_ID` and the `GIVEN_NAME` database fields into the session. The PAAS later uses this information to look up a user's role when determining which branch to take in the application flow.

Implementing the LoginMapping Interaction in the PAAS

To implement the `LoginMapping` Interaction, we first created a Mapping Definition, `LoginResultsMapping`, that maps fields from the database to Form Elements in the application. The `LoginResultsMapping` Mapping Definition maps the values from the `EMPLOYEE_ID` and the `GIVEN_NAME` database columns to the `employee_id` and `my_name` Form Elements of the `Login` Form Schema. The values are then available in the user session. The results from the `GetCredentialsByUserName` query provide the values from these database fields that the `LoginMapping` Interaction uses.

Update External Data Interaction

If the user chooses to create a new username, the PAAS uses the an Update External Data Interaction called `InsertNewUser` to insert the new username information into the external database.

InsertNewUser Interaction

The `InsertNewUser` Interaction uses the `InsertUser` query to insert a new row into the `Credentials` table using the information the user entered on the Create Username Page. Following is the `InsertUser` query.

```
insert into credentials (employee_id, password, user_name,  
given_name) values (?, ?, ?, ?)
```

Implementing the InsertNewUser Interaction

First, we defined four Query Arguments in the Query Definition for the `InsertUser` query—`employeeId`, `password`, `userName`, and `givenName`. Then we added a Parameter Mapping to the `InsertNewUser` Interaction for each of the Query Arguments. The Parameter Mappings specify that the values from the `employee_id`, `password`, `user_name` and `my_name` Form Elements on the `login` Form Schema are passed to the Query Argument at runtime.

NOTE The Parameter Mappings are not required to have the same name as the Query Arguments, because arguments are associated with Parameter Mappings based on the order they appear in Developer Studio, rather than the name. For example, the `InsertNewUser` Interaction, the value for the `employee_id` Form Element is used for the `employeeId` Query Argument.

Advanced Presentation Techniques

This chapter describes how we implemented the dynamic Response Values and repeating form data functionality in the Passport Application Approval System. This is advanced presentation functionality that requires some custom code, and is not defined through Developer Studio Editors.

Advanced presentation techniques in the PAAS

For the PAAS, we reviewed the project artifacts and determined that there are two Pages that require advanced presentation techniques: the My Worklist (`userworklist`) Page and the Assign Passport Application (`assign`) Page.

Page	Advanced Presentation Technique
My Work List Page	contains dynamic Response Values and a table with repeating form data.
Assign Passport Application Page	contains dynamic Response Values.

Dynamic Response Values

Dynamic Response Values are options for a multiple-choice Form Element (drop-down list or radio button) that are dynamically determined at runtime. In the PAAS, the My Work List (`userworklist`) and Assign Passport Application (`assign`) Pages contain dynamic Response Values; therefore, we created custom code to determine the options at runtime. The options for both Form Elements are registered at the Session-level.

NOTE Refer to *Creating Business Transaction Manager Applications*, Chapter 4, *Defining Form Schemas* for more information on dynamic Response Values.

We took the following approach to create the dynamic Response Values for the My Work List and Assign Passport Application Pages:

Stage	Process
1	Defined a Form Element and a Page Element in Developer Studio without defining Response Values or Response Choices.
2	Created custom code to define the options at runtime.
3	Created custom code to register the options with the application at runtime.

Application Status dynamic Response Values list

The Application Status dropdown Page Element on the My Work List Page is a dropdown list that provides the user with a list of valid application statuses (Submitted, In Progress, Denied, and Approved). The valid statuses are based on the user's role (Passport Agent or Passport Officer), which the system determines when the user logs into the system.

- ❖ Passport Agent: The Passport Agent can view and process applications for passports that have a status of Submitted or In Progress. After reviewing an application, the Passport Agent can assign it to a Passport Officer.
- ❖ Passport Officer: The Passport Officer can view and process all applications, regardless of status. After reviewing an application, the Passport Officer can approve or deny it.

For the dynamic options, we created the following in Developer Studio:

- ❖ an `app_status` Form Element in the Login Form Schema.
- ❖ an `app_status` dropdown Page Element for the My Work List Page.

We did not define static Response Values or Response Choices for the Form Element or Page Element in Developer Studio.

Defining dynamic Response Values for the Application Status dropdown

The dynamic Response Values for the `app_status` Page Element are registered at the Session-level because the options presented are specific to a particular user's session in the application. We included code in the `BlueprintApp.java` file that loads all options for both the Passport Agent and the Passport Officer when the application is initialized. The `BlueprintApp.java` file is the application initialize class, which is located in the `src` directory in Developer Studio. Once the PAAS is initialized, the options are loaded and stored to memory.

We included the following code in the `BlueprintApp.java` file to access the dynamic Response Values defined in an external database.

```
private void buildDynamicAnswerLists()
{
    DynamicAnswer all = new DynamicAnswer("All", "All", null, null);
    DynamicAnswer submitted = new DynamicAnswer("Submitted", "Submitted", null,
null);
    DynamicAnswer inprogress = new DynamicAnswer("In Progress", "In Progress", null,
null);
    DynamicAnswer assigned = new DynamicAnswer("Assigned", "Assigned", null, null);
    DynamicAnswer approved = new DynamicAnswer("Approved", "Approved", null, null);
    DynamicAnswer denied = new DynamicAnswer("Denied", "Denied", null, null);
}
```

Next, we included code to make the options lists specific to the user's role:

`MenuForOfficers` and `MenuForAgents`.

```
applicationStatusMenuForOfficers = new Vector();
applicationStatusMenuForAgents = new Vector();
```

Finally, we included code to define the valid Response Values for each list. The following code adds the appropriate options to each list:

❖ `MenuForOfficers`: all, submitted, in progress, assigned, approved, and denied.

```
applicationStatusMenuForOfficers.add(all);
applicationStatusMenuForOfficers.add(submitted);
applicationStatusMenuForOfficers.add(inprogress);
applicationStatusMenuForOfficers.add(assigned);
applicationStatusMenuForOfficers.add(approved);
applicationStatusMenuForOfficers.add(denied);
```

❖ `MenuForAgents`: all, submitted, and in progress.

```
applicationStatusMenuForAgents.add(all);
applicationStatusMenuForAgents.add(submitted);
applicationStatusMenuForAgents.add(inprogress);
```

Registering Response Values in the application

We implemented code to retrieve the options from memory based on the user's role and to register these options in the application. We included this code in the `ValidLogin` custom Handler.

We included the following code in the `ValidLogin` Handler to determine if the user's role is set in the session as Passport Agent. If the role is "Agent", the `MenuForAgent` option list is registered and displayed as the Response Choices for the `app_status` Page Element.

```
if ("Agent".equals(role))
{
```

```
DynamicAnswersManager.registerDynamicAnswersForSession(  
    new EformsHttpSession(session),  
    BlueprintConstants.BLUEPRINT_FORMS_OWNER,  
    BlueprintConstants.BLUEPRINT_LOGIN_FORM_TITLE,  
    "app_status",  
    BlueprintApp.getInstance().getApplicationStatusMenuForAgents());
```

If the user's role is set to "Officer", the `MenuForOfficer` option list is registered and displayed as the Response Choices for the `app_status` Page Element.

```
}  
else  
{  
  
    DynamicAnswersManager.registerDynamicAnswersForSession(  
        new EformsHttpSession(session),  
        BlueprintConstants.BLUEPRINT_FORMS_OWNER,  
        BlueprintConstants.BLUEPRINT_LOGIN_FORM_TITLE,  
        "app_status",  
        BlueprintApp.getInstance().getApplicationStatusMenuForOfficers());  
}
```

The `AuthenticateUser` Interaction invoked on the Login (`login`) Page uses the `ValidLogin` Handler. See [Chapter 5, *Application Flow*](#), for more information on the `AuthenticateUser` Interaction.

Select Officer Dynamic Response Values List

The Select Officer dropdown Page Element on the Assign Application Page provides a Passport Agent with a list of valid Passport Officers to which they can assign a passport. The list is based on the Passport Officers registered with the PAAS. Once a passport application is assigned to a Passport Officer, the PAAS updates the application status to "Assigned" and the application is ready for the assigned Passport Officer to approve.

For the dynamic options, we created the following in Developer Studio:

- ❖ the `selectOfficer` Form Element in the Login Form Schema.
- ❖ the `selectOfficer` dropdown Page Element for the Assign Passport Application Page.

We did not define static Response Values or Response Choices for the Form Element or Page Element in Developer Studio.

Defining dynamic Response Values for the Select Officer dropdown

The Response Values for the `selectOfficer` Page Element are registered at the Session-level. Although the options do not change based on a particular user's session, registering the options at the Session-level allows the PAAS to access the most current list of Passport Officers. For example, if a Passport Officer registers with the system while a Passport Agent is in the PAAS, the PAAS adds this Passport Officer as an option to the `selectOfficer` list when the Passport Agent accesses the `assign_app` Page.

In order to access the list of valid Passport Officers, we created the `GetOfficers` Interaction that queries an external database for registered Passport Officers and adds the results to the `OfficersResultSet`. The following code shows the SQL query defined for the `GetOfficers` Interaction:

```
select * from credentials where employee_id = ?
```

Registering Response Values in the application

We created the `AddOfficer` Interaction to access the `OfficersResultSet` and register the options with the application. These options are displayed as the Response Choices for the `selectOfficer` Page Element on the `assign_app` Page. We included the following code in the `AddOfficer` Handler, to access the result set created in the `GetOfficer` Interaction and register the options.

```
ResultSetView rsv = (ResultSetView) request.getAttribute("OfficersResultSet");
Enumeration enum = rsv.getEnumeration();

while (enum.hasMoreElements())
{
    ResultSetRow rsr = (ResultSetRow) enum.nextElement();
    String name = (String) rsr.get("GIVEN_NAME");
    DynamicAnswer officer = new DynamicAnswer(name, name, null, null);
    officers.add(officer);
}
DynamicAnswersManager.registerDynamicAnswersForSession(
    new EformsHttpSession(session),
    "Blueprint",
    "form_ssl1",
    "selectOfficer",
    officers);
```

See [Chapter 5, Application Flow](#), for more information on the `GetOfficers` Interaction.

Repeating form data

Business Transaction Manager provides repeating form data functionality that allows you to access non-active Form Results created in multiple Form Schemas and display the data in a single Page. Business Transaction Manager stores a user's response to each Page Element in a Form Result for the associated Form Element. Non-active Form Results are responses captured in a Form Schema other than the one in which the response is being displayed.

This section describes how we created a Page Style for the PAAS that accesses non-active Form Results and displays them on a single Page.

Repeating form data in the PAAS

The My Work List (`userworklist`) Page includes a table that displays data from Form Results captured in other Form Schemas. In order to implement this functionality, we created the `RepeatingGroups.jsp` Page Style and applied it to the `userworklist` Page.

To create the `userworklist` Page that contains the repeating form data we used the following approach:

Stage	Process
1	Created a Form Element and a Page Element in Developer Studio for the <code>userworklist</code> Page.
2	Created a custom Page Style containing the Business Transaction Manager tags to access and to display the Form Results.

We did not create additional Form Elements or Page Elements for the remaining data on the `userworklist` Page.

Creating custom Page Style

We created the `RepeatingGroups.jsp` custom Page Style that contains the Business Transaction Manager tags that access and display the responses from non-active Form Results on the `userworklist` Page.

We included the following lines of HTML code in the `RepeatingGroups.jsp` Page Style to create and populate the first table row. This table row contains the headings for each column of data.

```
<table cellpadding="5" cellspacing="0" border="0" bgcolor="#666666">
  <tr class="strongtext">
    <td>Tracking Number</td>
    <td>Status</td>
```

```

        <td>ID Number</td>
        <td>Last Name</td>
        <td><a href="help/form_ss91.html" class="text" target="help"
onclick="window.open('', 'help', 'resizable=yes, toolbar=no, directories=no, status=yes,
menubar=0, scrollbars=yes, location=no, width=500, height=500') ">Form SS91
Required?</a></td>
        <td>Assigned To</td>
        <td>&nbsp;</td>
</tr>

```

We then included the following tag to access all Form Results created for the SS11 Form Schema.

```

<eforms:forEachFormResult formowner="Blueprint" formtitle="form_ss11"
formresultvar="process">

```

Next, we included tags to access user responses in the Form Result for the following Form Elements: trackingNumber, status, citizen_id_number, last_name, and form_ss91_required.

```

<eforms:set var="ss11_status" value="$process.getResponse(\"status\").getResponse()"/
>
<eforms:set var="officer"
value="$process.getResponse(\"selectOfficer\").getResponse()" default=""/>
<eforms:if test="$filterBy == \"All\" || filterBy.equals(ss11_status)">
<tr class="text">
<td><eforms:responseValue formresultvar="process"
questionalias="trackingNumber"/></td>

<td><eforms:responseValue formresultvar="process"
questionalias="status"/></td>

<td><eforms:responseValue formresultvar="process"
questionalias="citizen_id_number"/></td>

<td><eforms:responseValue formresultvar="process"
questionalias="last_name"/></td>

<td><eforms:responseValue formresultvar="process"
questionalias="form_ss91_required"/></td>

```

Last, we applied the RepeatingGroups.jsp custom Style to the user_workload Page in Developer Studio.

Logging into the PAAS

The Passport Application Approval System is available to you as a sample project in Developer Studio. You can deploy the PAAS, either by executing it from within Developer Studio or by launching it in Developer Studio and deploying it to another environment. After deploying the PAAS, you can use the user credential information provided below to log into the application.

For information on creating the PAAS as a sample project in Developer Studio, launching the PAAS in Developer Studio, or packaging the PAAS in Developer Studio, see *Creating Business Transaction Manager Applications*.

Existing User Credentials for the PAAS

To log into the Passport Application Approval System, you can use any one of the following username and password combinations. Every user with access to the PAAS has a role of either Passport Agent or Passport Officer, and an Employee ID. See [Chapter 1, Project Overview](#) for more information on the application functionality available to each role.

Name	Username	Password	Role	Employee ID
John Smith	jsmith123	jsmith678	Passport Officer	666777
Henry Walker	hwalker123	hwalker678	Passport Officer	111222
Katherine Neil	kneil123	koneil678	Passport Officer	777888
Hanna Gibbens	hgibbens123	hgibbens678	Passport Agent	444555
Charles Albert	calbert123	calbert678	Passport Agent	222333

Creating New User Credentials for the PAAS

To create a new username in the Passport Application Approval System, you must enter a valid Employee ID. You can use any of the following Employee IDs to create a new username and password for the application.

Valid Employee IDs in the PAAS

The following table contains valid employee IDs in the PAAS

Valid Employee ID	Role
1000	Passport Agent
2000	Passport Officer
3000	Passport Agent
4000	Passport Officer
5000	Passport Agent

Use Cases for the PAAS

All requirements used to create the PAAS Blueprint application are captured in project artifacts. This Appendix discusses the Use Case artifacts, which documents the interaction of a user with the system. Each use case describes a sequence of interactions between the system and the user that yields an observable result that has value to the user. It includes the most common sequence of actions, the Basic Flow, and other alternative flows stemming from the basic flow. Two use cases were created for the PAAS.

This Appendix provides the following Use Cases:

- ❖ Login
- ❖ Approve Passport Application

Project Artifact: Login Use Case

Use Case Title: Login

1. Login

1.1 Brief Description

This use case describes the process for a registered user to log in to the Passport Application Approval System (henceforth the System), and the process for a new user to register with the system.

The users interacting with the system in this use case are Passport Agent and Passport Officer. Each user who logs into the Passport Application Approval System has one of these two roles.

2. Flow of Events

2.1 Basic Flow

2.1.1 Registered User Logs Into System

1.	User accesses the Passport Application Approval System.
2.	System prompts User to enter a username and password or to register with the system.
3.	User enters his/her username and password.
4.	System validates the username and password, and verifies that the username and password are registered. (See Section 3.1)
5.	System prompts User to select a language to view the system in.
6.	User chooses to login.
7.	System allows User to access the system.
	End of Flow

2.2 Alternative Flows

2.2.1 User Registers with the System

1.	Basic Flow: Steps 1-2
2.	User chooses to register with the system.
3.	System prompts user to enter their Employee ID, name, a username, and a password.
4.	User enters their Employee ID, name, a username, and a password.
5.	System authenticates the Employee ID, verifies that the username and password are valid, and verifies that the username is not already registered. (See Section 3.1)
6.	User chooses to create the new username and password.
7.	System registers the username and password, and then prompts User to login with the new username and password.
8.	Continue Basic Flow: Step 3

2.2.1.1 User Enters a User Name that is Already Registered

1.	Alt Flow 2.2.1: Steps 1-4
2.	System finds that the User entered a username that is already registered with the system.
3.	System presents error message to the User and prompts User to enter a new username.
4.	Continue Alt Flow 2.2.1: Step 3

2.2.1.2 System Unable to Authenticate Employee ID (for the first time)

1.	Alt Flow 2.2.1: Steps 1-4
2.	System is unable to authenticate the User's Employee ID (for the first time).
3.	System presents error message to the User and prompts User to reenter their Employee ID.
4.	Continue Alt Flow 2.2.1: Step 3

2.2.1.3 System Unable to Authenticate Employee ID (for the second time)

1.	Alt Flow 2.2.1: Steps 1-4
2.	System is unable to authenticate the User's Employee ID (for the second time).
3.	System presents error message to the User and prompts User to reenter their Employee ID.
4.	Continue Alt Flow 2.2.1: Step 3

2.2.1.4 System Unable to Authenticate Employee ID (for the third time)

1.	Alt Flow 2.2.1: Steps 1-4
2.	System is unable to authenticate the User's Employee ID (for the third time).
3.	System presents an error message stating that the User should contact their system administrator for assistance in registering with the system.
4.	End Flow

2.2.2 User Enters Invalid Username and/or Password

1.	Basic Flow: Steps 1-3
2.	System finds that the User has entered an invalid username and/or password.
3.	System presents error message to the User.
4.	Continue Basic Flow: Step 2

2.2.3 User Enters Unregistered Username

1.	Basic Flow: Steps 1-3
2.	System finds that the User has entered a username that is not registered.
3.	System presents error message to the User.
4.	Continue Basic Flow: Step 2

2.2.4 User Enters Valid Username with Incorrect Password (for the first time)

1.	Basic Flow: Steps 1-3
2.	System finds that the User has entered a registered username, but that the password is not correct (for the first time).
3.	System presents error message to the User.
4.	Continue Basic Flow: Step 2

2.2.5 User Enters Valid Username with Incorrect Password (for the second time)

1.	Basic Flow: Steps 1-3
2.	System finds that the User has entered a registered username, but that the password is not correct (for the second time).
3.	System presents error message to the User.
4.	Continue Basic Flow: Step 3

2.2.6 User Enters Valid Username with Incorrect Password (for the third time)

1.	Basic Flow: Steps 1-3
2.	System finds that the User has entered a registered username, but that the password is not correct (for the third time).
3.	System presents an error message stating that the User should contact their system administrator for assistance in logging into the system.
4.	Continue Basic Flow: Step 2

3. Special Requirements

3.1 User Name and Password

The username and password must:

- Be between 8 and 10 alphanumeric characters
- Contain at least one number (0-9)
- Contain at least one letter (a-z)

- Username and password cannot be the same
- Password must be case sensitive

Project Artifact: Approve Passport Application Use Case

Use Case Title: Approve Passport Application

1. Approve Passport Application

1.1 Brief Description

This use case describes how users review and approve an application for a new passport in the Passport Application Approval System (henceforth the System). It describes the process for a Passport Agent to review a Form SS11, an application for a new passport submitted by a citizen. If the passport application is complete and accurate, the Passport Agent can assign the application to a Passport Officer for approval. The Passport Agent can also add a Form SS91 to the passport application if the citizen's previously issued passport was lost or stolen.

It is a precondition for this use case that the user has successfully logged into the system. (See Login Use Case)

The users interacting with the system in this use case are Passport Agent and Passport Officer. Each user who logs into the Passport Application Approval System has one of these two roles.

2. Flow of Events

2.1 Basic Flows

2.1.1 Assign Passport Application

1.	System displays passport applications available for processing by the User (See Section 3.2 and 3.3)
2.	User chooses a passport application to process (Form SS11).
3.	System displays passport application.
4.	User reviews and edits passport application.
5.	System finds that the passport application contains no errors.
6.	System finds that the passport application does not require a Form SS91 (See section 3.5)
7.	System displays summary of the passport application.
8.	System checks role associated with the User's login information. (See Section 3.1)
9.	System determines that the User has the role of Passport Agent.
10.	System prompts User to assign the passport application to a Passport Officer.
11.	User selects a Passport Officer to assign the passport application to.
12.	System assigns the passport to the selected Passport Officer.
	End Flow

2.2 Alternative Flows

2.2.1 Approve Passport Application

1.	Basic Flow: Steps 1-8
2.	System determines that the User has the role of Passport Officer.
3.	System prompts User to approve or deny the passport application.
4.	User chooses to approve the passport application.
	End Flow

2.2.1.1 Deny Passport Application

1.	Alt Flow 2.2.1: Steps 1-3
2.	User chooses to deny the passport application.
3.	System prompts User to enter a reason for denying the passport application.
4.	User enters reason for denying the passport application.
	End Flow

2.2.2 Passport Application Requires Form SS91

1.	Basic Flow: Steps 1-5
2.	System finds that the passport application requires a Form SS91. (See Section 3.5)
3.	System prompts User to add a Form SS91 to the passport application.
4.	User chooses to add a Form SS91 to the passport application.
5.	System adds a Form SS91, and prompts User to complete Form SS91.
6.	User completes Form SS91.
7.	System finds that Form SS91 contains no errors.
8.	Continue Basic Flow: Step 7

2.2.2.1 Delete Form SS91

1.	Alt Flow 2.2.2: Steps 1-2
2.	System finds that a Form SS91 has already been added.
3.	System prompts User to edit or delete the form.
4.	User chooses to delete the form.
5.	System prompts User to confirm the delete.
6.	User confirms the delete.
7.	System deletes Form SS91.
8.	Continue Alt Flow 2.2.2: Step 3

2.2.2.2 Edit Form SS91

1.	Alt Flow 2.2.2: Steps 1-2
2.	System finds that a Form SS91 has already been added.
3.	System prompts User to edit or delete the form.
4.	User chooses to edit the form.
5.	Continue Alt Flow 2.2.2: Step 6

2.2.2.3 System Finds that Form SS91 Contains Errors

1.	Alt Flow 2.2.2: Steps 1-6
2.	System finds that Form SS91 contains errors.
3.	System provides User with an error message.
4.	User resolves error.
5.	Continue Alt Flow 2.2.2: Step 6

2.2.3 System Finds that Passport Application (Form SS11) Contains Errors

1.	Basic Flow: Steps 1-4
2.	System finds that Form SS11 contains errors.
3.	System provides User with an error message.
4.	User resolves error.
5.	Continue Basic Flow: Step 5

3. Special Requirements

3.1 Roles

All users that log into the Passport Application Approval System have one of two roles: Passport Agent or Passport Officer.

3.2 Passport Agent Role

A user with the role of Passport Agent can process passport applications with the following statuses:

- Submitted
- In Process

3.3 Passport Officer Role

A user with the role of Passport Officer can process passport applications with the following statuses:

- Submitted
- In Process
- Assigned
- Approved
- Denied

3.4 Form SS91

System must require the user to add a Form SS91 if the passport applicant is unable to return a previously issued passport to the Passport Agency (passport was stolen or is lost).

3.5 Application Status

Passport applications must have one of the following 5 statuses:

- **Submitted** – application has been submitted to the Passport Agency, but has not been processed.
- **In Progress** – application has been processed, but has not yet been assigned to a Passport Officer.
- **Assigned** – application has been assigned to a Passport Officer, but has not yet been approved or denied.
- **Approved** – application has been approved by a Passport Officer.
- **Denied** – application has been denied by a Passport Officer.

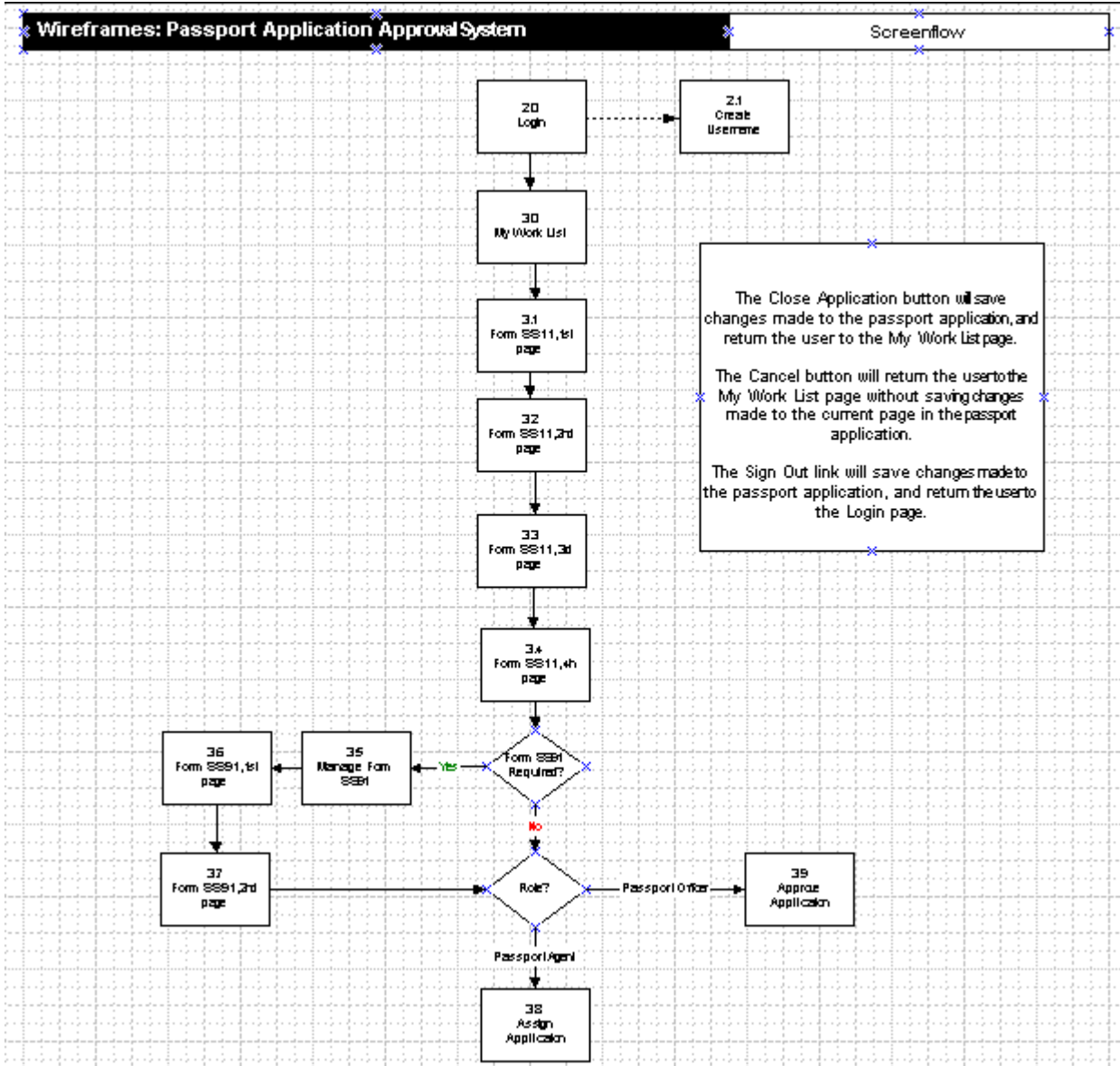
3.6 Form SS91 No Longer Required

If the user adds Form SS91 to the passport application, then modifies Form SS11 so that Form SS91 is no longer required, the system should not delete Form SS91. Form SS91 should remain available for editing and deletion although it is not required.

Screenflow for the PAAS

All requirements used to create the PAAS Blueprint application are captured in project artifacts. The Screenflow artifact documents the Page flow throughout the application. The Screenflow appears on the following page.

Screenflow



Wireframes for the PAAS


All requirements used to create the PAAS Blueprint application are captured in project artifacts. This Appendix discusses the Wireframe artifacts, which provide a visual representation for each Page in the application. This includes the text on the Page, Page Elements on the Page, and the navigational elements that are available to the user (for example, buttons).

This Appendix provides the following Wireframes:

- ❖ 2.0 Login
- ❖ 2.1 Create Username
- ❖ 3.0 My Work List
- ❖ 3.1 Form SS11, 1st Page
- ❖ 3.2 Form SS11, 2nd Page
- ❖ 3.3 Form SS11, 3rd Page
- ❖ 3.4 Form SS11, 4th Page
- ❖ 3.5 Manage Form SS91
- ❖ 3.6 Form SS91, 1st Page
- ❖ 3.7 Form SS91, 2nd Page
- ❖ 3.8 Assign Application
- ❖ 3.9 Approve Application

2.0 Login

Wireframes : Passport Application Approval System	2.0 - Login
---	-------------



Application Logo

Passport Application Approval System

A Blueprint for Business Transaction Manager Applications

Welcome to the Passport Application Approval System

This system can be used to review, edit, and approve applications for a new passport.

A [Link to Context Sensitive Help.](#)

Registered Users

If you are already registered with the Passport Application Approval System, please enter your username and password below, then select Log In to proceed.

Username:

Password:
(case sensitive)

Select Language: ▾

Log In >

Note: if you have forgotten your username or password, please contact your system administrator.

Unregistered Users


If you are not registered with the Passport Application Approval System, you will need to create a username and password.

Note: you will need to enter your A [Employee ID](#) to create a new username and password.

Create Username >

2.1 Create Username

Wireframes : Passport Application Approval System	2.1 - Create Username
---	-----------------------



Application Logo

Passport Application Approval System

A Blueprint for Business Transaction Manager Applications

Create Passport Application Approval System ID

To use this system you need to create a username and password.

Your username and password must:

- be between 8 and 10 characters (letters and numbers)
- contain at least one number(0-9)
- contain at least one letter(a-z)
- your username and password cannot be the same

Please note that your password is case sensitive. You will need to remember the username and password you enter.

A Employee ID:

Name:

Username:

Password:

Retype password:

After you enter your username and password, select Create Username. Your username and password will be created, and you will be returned to the Login screen. You can then use your new username and password to access the system.

Cancel
Create Username >
B

A


The Employee ID will be validated against an external database. Based on the Employee ID, the user will have a role of either Passport Agent or Passport Officer. Each Employee ID is associated with a role.

B

The Cancel button returns user to the Login page. The Create User Name button also returns user to the Login page.

3.0 My Work List

Wireframes : Passport Application Approval System
3.0 My Work List



Passport Application Approval System
A Blueprint for Business Transaction Manager Applications

Welcome
[Sign Out](#)

My Work List

Welcome to the Passport Application Approval System. This system allows the Passport Agency to review, edit and approve applications for new passports. The application process includes two forms, Form SS11 and Form SS91. Form SS11 is submitted by the passport applicant and provides all of the information needed to issue a new passport. Form SS91 is required in addition to Form SS11 if the applicant is unable to return a previously issued passport to the Passport Agency. Form SS91 is completed by the Passport Agent.

Search for passport applications.

Application status: All Filter

Select Process to review and update a passport application.

Tracking Number	Status	ID Number	Last Name	Form SS91 Required?	Assigned To	
12576985	Submitted	32165498734	Smoe, Joe	Yes	Not Assigned	<input type="button" value="Process"/>
12576989	Assigned	32165498734	Doe, Jane	No	Abet, Paul	<input type="button" value="Process"/>
12576995	In Progress	32165498734	Smith, John	Yes	Not Assigned	<input type="button" value="Process"/>
12577895	Approved	32165498734	Smith, Ann	No	Russell, Kathy	<input type="button" value="Process"/>
12576790	Denied	32165498734	Doe, John	No	Roberts, Kim	<input type="button" value="Process"/>

A The list of statuses that can be displayed in this dropdown includes:

- Submitted
- In Progress
- Assigned
- Approved
- Denied

This dropdown is dynamic based on the user's role. If the user has a Passport Officer role, all statuses will be available. If the user has a Passport Agent role, they can only view passports with a status of Submitted and In Progress. A Passport Officer can approve or deny an application while it is in any status. The application does not have to be assigned prior to approval.


When user first enters this page, the Application Status will be defaulted to "All" and all applications appropriate to the user's role will appear in the list.

All results returned by the query are shown in the application's list (there is no limit to the number of applications that appear on the list).

Passport Officers can choose to process any application regardless of who it is assigned to.

3.1 Form SS11, 1st page

Wireframes : Passport Application Approval System
3.1 Form SS11, 1st page



Passport Application Approval System
A Blueprint for Business Transaction ManagerApplications

Welcome
[Sign Out](#)

Form SS11 - Page 1 of 4

Tracking Number: ##### Close Passport Application

Following is the information submitted on [Form SS11](#) by the passport applicant:

1. Name:

First	Middle	Last
2. Mail passport to:

Address line 1:
Address line 2:
Address line 3:
Address line 4:
Postal Code: Country:
3. Gender:

<input type="checkbox"/> Yes	<input type="checkbox"/> No
------------------------------	-----------------------------
4. Place of birth:

(City and State or City and County)
5. Date of birth:

DD	MM	YYYY
----	----	------
6. Citizen Identification Number:

--
7. Height:

--
8. Hair color:

--
9. Eye color:


--

Cancel
< Back
Next >

All required fields on page should be populated with the information submitted by the passport applicant. All fields should be writable so that the Passport Agent or Officer can make updates.

3.2 Form SS11, 2nd page

Wireframes : Passport Application Approval System
3.2 Form SS11, 2ndpage



Passport Application Approval System

A Blueprint for Business Transaction Manager Applications

Welcome
[Sign Out](#)

Form SS11 - Page 2 of 4

Close Passport Application

Tracking Number: #####

10. Home telephone:

11. Business telephone:

12. Occupation:

13. Permanent Address:

Address line 1:

Address line 2:

Address line 3:

Address line 4:

Postal Code: Country:

14. Father's Information:

First Middle Last

Father's full name:

Father's birthplace:
(City and State or City and County)

Father's birth date: / /
DD MM YYYY

Citizen: Yes No

15. Mother's Information:

First Middle Last

Mother's full name:

Mother's birthplace:
(City and State or City and County)


Mother's birth date: / /
DD MM YYYY

Citizen: Yes No

All required fields on page should be populated with the information submitted by the passport applicant. All fields should be writable so that the Passport Agent or Officer can make updates.

3.3 Form SS11, 3rd page

Wireframes : Passport Application Approval System
3.3 Form SS11, 3rd page



Passport Application Approval System

A Blueprint for Business Transaction Manager Applications

Welcome
[Sign Out](#)

Form SS11 - Page 3 of 4

Tracking Number: ##### [Close Passport Application](#)

16. Have you ever been married? Yes No

17. Spouse or Former Spouse's Information:

Spouse's full name:
First Middle Last

Spouse's birthplace:
(City and State or City and County)

Spouse's birth date:
DD MM YYYY

Citizen: Yes No

18. Date of most recent marriage:
DD MM YYYY

19. Widowed or divorced? Yes No
 If yes, please specify date
DD MM YYYY

20. Other names you have used:

(1)
First Middle Last

(2)
First Middle Last

(3)
First Middle Last

21. Previously Issued Passport Information:

Have you ever been issued a passport? Yes No If yes, mail passport if available.

Most recent passport number:


Approximate issue date:
DD MM YYYY

Status of passport: Submitted Lost

All required fields on page should be populated with the information submitted by the passport applicant. All fields should be writable so that the Passport Agent or Officer can make updates.

3.4 Form SS11, 4th page

Wireframes : Passport Application Approval System
3.4 Form SS11, 4th page



Passport Application Approval System

A Blueprint for Business Transaction Manager Applications

Welcome
[Sign Out](#)

Form SS11 - Page 4 of 4

Tracking Number: ##### Close Passport Application

FOR PASSPORT AGENT USE Only:

1. If Applicant is Over Age 14, verify that you have received and reviewed the following :

- Proof of citizenship
(must be different than the identification provided for the prookidenship)
- Proof of identity
(must be different than the identification provided for the prookidenship)
- Two Photographs

2. If Applicant is Under Age 14, verify that you have received and reviewed the following :

- Evidence of child's citizenship
- Evidence of child's relationship to parent
- Parental Identification

2a. Parental Identifying Document:

- Driver's License
- Passport

Issue date:

Number:

Expiration date:

Name on document:


Cancel Back Next

No fields should be populated with information. All fields should be visible so that the Passport Agent or Officer can enter information.

Appendix D
Wireframes for the PAAS

3.5 Manage Form SS91

Wireframes : Passport Application Approval System
3.5 Manage Form SS91



Application
Logo

Passport Application Approval System

A Blueprint for Business Transaction ManagerApplications

Welcome
[Sign Out](#)

Manage Form SS91

Tracking Number: ##### Close Passport Application

Form SS91 is required for passport applications where a previously issued passport is lost or was stolen.

A citizen may bear only one valid or potentially valid passport at a time, except as otherwise authorized by the government. Therefore it is necessary to submit Form SS91 with an application for a new passport when a previous valid or potentially valid passport cannot be presented. Form SS91 must be used to set forth in detail why the previous passport cannot be presented.

This application for a new passport requires Form SS91.

Add Form SS91 A

Edit Form SS91 B

Delete Form SS91 C

Cancel
Back


A If Form SS91 has already been added, the Add Form SS91 button will not be available.

B The Edit Form SS91 button is only available after a Form SS91 has been added.

C The Delete Form SS91 button is only available after a Form SS91 has been added.

3.6 Form SS91 1st page

Wireframes : Passport Application Approval System
3.6 Form SS91, 1st page



Application
Logo

Passport Application Approval System

A Blueprint for Business Transaction Manager Applications

Welcome
[Sign Out](#)

Form SS91 - Page 1 of 2

Tracking Number: ##### Close Passport Application

Complete [Form SS91](#) for inclusion with passport applications where a previously issued passport is lost or was stolen.

1. Citizen's name:

First
Middle
Last
2. Gender: Yes No
3. Passport number:
4. Issue date:

DD
MM
YYYY
5. Place of issue:
(City and State or City and County)
6. Place of birth:
(City and State or City and County)
7. Date of birth:


DD
MM
YYYY
8. Citizen's Address:
 - Address line 1:
 - Address line 2:
 - Address line 3:
 - Address line 4:
 - Postal Code: Country:

No fields should be populated with information. All fields should be untable so that the Passport Agent or Officer can enter information.

The system should verify that the name and passport number entered on this form matches the information entered in the Name field and the Most Recent Passport Number field on Form SS11.

3.7 Form SS91, 2nd page

Wireframes : Passport Application Approval System 3.7 Form SS91, 2nd page



Passport Application Approval System

A Blueprint for Business Transaction Manager Applications

Welcome
[Sign Out](#)

Form SS91 - Page 2 of 2

Tracking Number: ##### Close Passport Application

9. How was passport lost or stolen?

10. If stolen, were police authorities notified?

11. What efforts did the citizen make to recover the passport?

I certify that the information furnished herein was obtained from the person to whom the passport was issued, and is correct and complete to the best of my knowledge and belief.


Cancel Back Next

No fields should be populated with information. All fields should be uneditable so that the Passport Agent or Officer can enter information.

3.8 Assign Application

Wireframes : Passport Application Approval System

3.8 Assign Application



Passport Application Approval System
A Blueprint for Business Transaction ManagerApplications

Welcome
[Sign Out](#)

Assign Passport Application

Tracking Number: #####
Close Passport Application

Applicant Last Name:

Citizen Identification Number:

Select the officer to assign the application to:


Optional comments:

A Responses are read-only

B Dynamic dropdown. Populated with the name of all users that have created a username for the login process, and have the role of Passport Officer. The name shown in the dropdown list is the name the user entered when they created their username and password.

3.9 Approve Application

Wireframes : Passport Application Approval System 3.9 Approve Application



Application Logo

Passport Application Approval System

A Blueprint for Business Transaction Manager Applications

Welcome
[Sign Out](#)

Approve Passport Application

[Close Passport Application](#)

Tracking Number: #####

Applicant Last Name:

Citizen Identification Number: A

Application assigned to:

Select status for application: B

If the application is denied, provide detailed comments:

A Responses are read-only

B Dropdown options:
Approved
Denied

Business Rules for PAAS

All requirements used to create the PAAS Blueprint application are captured in project artifacts. The Business Rules artifact documents the Business Rules for the PAAS and the associated error and warning messages. For example, it specifies the number of characters a user can enter for a Page Element and the valid characters a user can enter for a Page Element.

This Appendix contains Business Rules for the following Pages:

- ❖ 2.0 Login Business Rules
- ❖ 2.1 Create Username Business Rules
- ❖ 3.0 My Work List Business Rules
- ❖ 3.1 Form SS11, 1st Page Business Rules
- ❖ 3.2 Form SS11, 2nd Page Business Rules
- ❖ 3.3 Form SS11, 3rd Page Business Rules
- ❖ 3.4 Form SS11, 4th Page Business Rules
- ❖ 3.6 Form SS91, 1st Page Business Rules
- ❖ 3.7 Form SS91, 2nd Page Business Rules
- ❖ 3.8 Assign Application Business Rules
- ❖ 3.9 Approve Application Business Rules

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
2.0 - Login	Username	TextField	10	<ul style="list-style-type: none"> · Be between 8 and 10 alphanumeric characters · Contain at least one number (0-9) · Contain at least one letter (a-z) · Username and password can not be the same 	YES	Error	<p>(1) Please enter your username.</p> <p>(2) You entered a username that is not registered with the system. Please try again or choose Create New Username to register with the system.</p>	login_user	login
	Password	TextField	10	<ul style="list-style-type: none"> · Be between 8 and 10 alphanumeric characters · Contain at least one number (0-9) · Contain at least one letter (a-z) · Username and password can not be the same · Password must be case sensitive 	YES	Error	<p>(1) Please enter your password.</p> <p>(2) You entered a password that is not valid for your username. Please try again or contact your system administrator for assistance.</p>	login_password	login
				validate that password matches user name - first and second time		Error	(1) You entered a password that is not valid for your username. Please try again or contact your system administrator for assistance.		login
				validate that password matches user name - third time		Error	(1) You exceeded the maximum number of attempts to login. Please contact your system administrator for assistance.		login

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
2.1 - Create User Name	Employee ID	TextField	10	Numeric only	YES	Error	(1) Please enter your Employee ID. (2) You entered an invalid Employee ID. Please verify that you only entered numeric characters.	employee_id	createuser
				validate that Employee ID exists - first and second time		Error	(1) You entered an Employee ID that does not exist. Please re-enter your Employee ID.		createuser
				validate that Employee ID exists - third time		Error	(1) You exceeded the maximum number of attempts to register. Please contact your system administrator for assistance.		createuser
	Name	TextField	50	Alphanumeric and hyphen characters only	YES	Error	(1) Please enter your name. (2) The name you entered is invalid. The name can only contain alphanumeric characters and the following special character: hyphen (-).	my_name	createuser
	Username	TextField	10	<ul style="list-style-type: none"> · Be between 8 and 10 alphanumeric characters · Contain at least one number (0-9) · Contain at least one letter (a-z) · Username and password can not be the same 	YES	Error	(1) Please enter a username. (2) You entered an invalid username. Your username must be between eight and ten characters in length, contain only alphanumeric characters, and contain at least one letter and one number. Your username cannot be the same as your password.	user_name	createuser

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
2.1 Create User Name				validate that user name does not already exist		Error	(1) The username you entered is already registered. Please enter a new user name.		createuser
	Password	TextField	10	<ul style="list-style-type: none"> · Be between 8 and 10 alphanumeric characters · Contain at least one number (0-9) · Contain at least one letter (a-z) · Username and password can not be the same · Password must be case sensitive 	YES	Error	<p>(1) Please enter a password.</p> <p>(2) You entered an invalid password. Your password must be between eight and ten characters in length, contain only alphanumeric characters, and contain at least one letter and one number. Your password cannot be the same as your username.</p>	password	createuser
	Retype password	TextField	10	Must be the same value entered for password	YES	Error	<p>(1) Please retype your password.</p> <p>(2) Your password entries do not match. Please re-enter your password.</p>	re_password	createuser

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.1 Form SS11, 1st page	First Name	TextField	50	Alphanumeric and hyphen characters only	YES	Error	(1) Please enter the passport applicant's first name. (2) The first name you entered is invalid. The first name can only contain alphanumeric characters and the following special character: hyphen (-).	first_name	ss11page1
	Middle Name	TextField	50	Alphanumeric and hyphen characters only	NO	Error	(1) The middle name you entered is invalid. The middle name can only contain alphanumeric characters and the following special character: hyphen (-).	middle_name	ss11page1
	Last Name	TextField	50	Alphanumeric and hyphen characters only	YES	Error	(1) Please enter the passport applicant's last name. (2) The last name you entered is invalid. The last name can only contain alphanumeric characters and the following special character: hyphen (-).	last_name	ss11page1
	Address line 1	TextField	30	Alphanumeric, comma, and hyphen characters only	YES	Error	(1) Please enter the passport applicant's mailing address. (2) The first line of the address you entered is invalid. The address can only contain alphanumeric characters and the following special characters: hyphen (-) and comma (.).	address1	ss11page1

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.1 Form SS11, 1st page	Address line 2	TextField	30	Alphanumeric, comma, and hyphen characters only	NO	Error	(1) The second line of the address you entered is invalid. The address can only contain alphanumeric characters and the following special characters: hyphen (-) and comma (.).	address2	ss11page1
	Address line 3	TextField	30	Alphanumeric, comma, and hyphen characters only	NO	Error	(1) The third line of the address you entered is invalid. The address can only contain alphanumeric characters and the following special characters: hyphen (-) and comma (.).	address3	ss11page1
	Address line 4	TextField	30	Alphanumeric, comma, and hyphen characters only	NO	Error	(1) The fourth line of the address you entered is invalid. The address can only contain alphanumeric characters and the following special characters: hyphen (-) and comma (.).	address4	ss11page1
	Postal Code	TextField	10	numbers, letters and hyphen only	YES	Error	(1) Please enter a postal code for the mailing address. (2) The postal code you entered is invalid. The postal code can only contain alphanumeric characters and the following special character: hyphen (-).	postalCode	ss11page1

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.1 Form SS11 1st Page	Country	TextField	20	Alpha characters only	NO	Error	(1) The country you entered is invalid. The country can only contain letters and spaces.	country	ss11page1
	Male	RadioButton	1	Male or Female required, M stored	YES			gender	ss11page1
	Female	RadioButton	1	Male or Female required, F stored	YES				ss11page1
	Place of birth	TextField	40	Alphanumeric characters only, comma, hyphen and apostrophe allowed	YES	Error	(1) Please enter the passport applicant's place of birth. (2) The place of birth you entered is invalid. The place of birth can only contain alphanumeric characters and the following special characters: hyphen (-), apostrophe ('), and comma (,).	place_of_birth	ss11page1
	Birth Day	TextField	2	Two digit number, 01 to 31, validate on month	YES	Error	(1) Please enter the passport applicant's day of birth. (2) The day of birth you entered is invalid. The day of birth can only contain numeric characters and must be between 01 and 31.	birth_day	ss11page1

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.1 Form SS11 1st Page	Birth Month	TextField	2	Two digit number, 01 to 12	YES	Error	(1) Please enter the passport applicant's month of birth. (2) The month of birth you entered is invalid. The month of birth can only contain numeric characters and must be between 01 and 12.	birth_month	ss11page1
	Birth Year	TextField	4	Four digit number 1900 to today	YES	Error	(1) Please enter the passport applicant's year of birth. (2) The year of birth you entered in invalid. The year of birth can only contain numeric characters and must be after 1900.	birth_year	ss11page1
	BirthDate			Combination of above fields		Error	The date of birth you entered is invalid. The date can not be in the future.		ss11page1
	Citizen Identification Number	TextField	11	Number and hyphens only	YES	Error	(1) Please enter the passport applicant's Citizen Identification Number. (2) The Citizen Identification Number you entered is invalid. The Citizen Identification Number can only contain numbers and hyphens.	citizen_id_number	ss11page1

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.1 Form SS11 1st Page	Height	TextField	30	Alphanumeric characters, comma, hyphen and apostrophe only	YES	Error	(1) Please enter the height of the passport applicant. (2) The height you entered is invalid. The height can only contain alphanumeric characters and the following special characters: hyphen (-), apostrophe ('), and comma (,).	height	ss11page1
	Hair color	TextField	8	Alpha characters only	YES	Error	(1) Please enter the passport applicant's hair color. (2) The hair color you entered is invalid. The hair color can only contain letters.	hair_color	ss11page1
	Eye color	TextField	8	Alpha characters only	YES	Error	(1) Please enter the passport applicant's eye color. (2) The eye color you entered is invalid. The eye color can only contain letters.	eye_color	ss11page1

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.2 Form SS11, 2nd page	Home Telephone	TextField	20	Numbers, hyphens, plus, and parentheses only	YES	Error	(1) Please enter the passport applicant's home telephone number. (2) The home telephone number you entered is invalid. The home telephone number can only contain numbers and the following special characters: hyphen (-), plus (+), and parentheses ().	home_phone	ss11page2
	Business Telephone	TextField	20	Numbers, hyphens, plus, and parentheses only	NO	Error	(1) The business telephone number you entered is invalid. The business telephone number can only contain numbers and the following special characters: hyphen (-), plus (+), and parentheses ().	work_phone	ss11page2
	Occupation	TextField	30	Alpha characters only	NO	Error	(1) The occupation you entered is invalid. The occupation can only contain letters.	occupation	ss11page2
	Address line 1	TextField	30	Alphanumeric, comma, and hyphen characters only	YES	Error	(1) Please enter the passport applicant's permanent address. (2) The first line of the address you entered is invalid. The address can only contain alphanumeric characters and the following special characters: hyphen (-) and comma (,).	permanent_address1	ss11page2

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.2 Form SS11, 2nd page	Home Telephone	TextField	20	Numbers, hyphens, plus, and parentheses only	YES	Error	(1) Please enter the passport applicant's home telephone number. (2) The home telephone number you entered is invalid. The home telephone number can only contain numbers and the following special characters: hyphen (-), plus (+), and parentheses ().	home_phone	ss11page2
	Business Telephone	TextField	20	Numbers, hyphens, plus, and parentheses only	NO	Error	(1) The business telephone number you entered is invalid. The business telephone number can only contain numbers and the following special characters: hyphen (-), plus (+), and parentheses ().	work_phone	ss11page2
	Occupation	TextField	30	Alpha characters only	NO	Error	(1) The occupation you entered is invalid. The occupation can only contain letters.	occupation	ss11page2
	Address line 1	TextField	30	Alphanumeric, comma, and hyphen characters only	YES	Error	(1) Please enter the passport applicant's permanent address. (2) The first line of the address you entered is invalid. The address can only contain alphanumeric characters and the following special characters: hyphen (-) and comma (.).	permanent_address1	ss11page2

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.2 Form SS11, 2nd page	Country	TextField	20	Alpha characters only	NO	Error	(1) The country you entered is invalid. The country can only contain letters and spaces.	permanent_country	ss11page2
	Father's First Name	TextField	50	Alphanumeric and hyphen characters only	YES	Error	(1) Please enter the father's first name. (2) The father's first name you entered is invalid. The first name can only contain alphanumeric and the following special character: hyphen (-).	fathers_first_name	ss11page2
	Father's Middle Name	TextField	50	Alphanumeric and hyphen characters only	NO	Error	(1) The father's middle name you entered is invalid. The middle name can only contain alphanumeric characters and the following special character: hyphen (-).	fathers_middle_name	ss11page2
	Father's Last Name	TextField	50	Alphanumeric and hyphen characters only	YES	Error	(1) Please enter the father's last name. (2) The father's last name you entered is invalid. The last name can only contain alphanumeric characters and the following special character: hyphen (-).	fathers_last_name	ss11page2

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.2 Form SS11, 2nd page	Father's Birthplace	TextField	40	Alphanumeric characters only, comma and apostrophe allowed	YES	Error	(1) Please enter the father's place of birth. (2) The father's place of birth you entered is invalid. The place of birth can only contain alphanumeric characters and the following special characters: hyphen (-), apostrophe ('), and comma (,).	fathers_birthplace	ss11page2
	Father's Day	TextField	2	Two digit number, 01 to 31, validate on month	YES	Error	(1) Please enter the father's day of birth. (2) The father's day of birth you entered is invalid. The day of birth can only contain numeric characters, and must be between 01 and 31.	fathers_birth_day	ss11page2
	Father's Month	TextField	2	Two digit number, 01 to 12	YES	Error	(1) Please enter the father's month of birth. (2) The father's month of birth you entered is invalid. The month of birth can only contain numeric characters, and must be between 01 and 12.	fathers_birth_month	ss11page2

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.2 Form SS11, 2nd page	Father's Year	TextField	2	Four digit number 1900 to today	YES	Error	(1) Please enter the father's year of birth. (2) The father's year of birth you entered is invalid. The year of birth can only contain numeric characters, and must be after 1900.	fathers_birth_year	ss11page2
				Combination of above fields		Error	The father's birth date you entered is invalid. The date can not be in the future.		ss11page2
	Father's Yes	RadioButton	1	Yes or No required, Y stored	YES			father_citizen	ss11page2
	Father's No	RadioButton	1	Yes or No required, N stored	YES				ss11page2
	Mother's First Name	TextField	50	Alphanumeric and hyphen characters only	YES	Error	(1) Please enter the mother's first name. (2) The mother's first name you entered is invalid. The first name can only contain alphanumeric characters and the following special character: hyphen (-).	mothers_first_name	ss11page2
	Mother's Middle Name	TextField	50	Alphanumeric and hyphen characters only	NO	Error	(1) The mother's middle name you entered is invalid. The middle name can only contain alphanumeric characters and the following special character: hyphen (-).	mothers_middle_name	ss11page2

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.2 Form SS11, 2nd page	Mother's Last Name	TextField	50	Alphanumeric and hyphen characters only	YES	Error	(1) Please enter the mother's last name. (2) The mother's last name you entered is invalid. The last name can only contain alphanumeric characters and the following special character: hyphen (-).	mothers_last_name	ss11page2
	Mother's Birthplace	TextField	40	Alphanumeric characters only, comma and apostrophe allowed	YES	Error	(1) Please enter the mother's place of birth. (2) The mother's place of birth you entered is invalid. The place of birth can only contain alphanumeric characters and the following special characters: hyphen (-), apostrophe ('), and comma (,).	mothers_birthplace	ss11page2
	Mother's Day	TextField	2	Two digit number, 01 to 31, validate on month	YES	Error	(1) Please enter the mother's day of birth. (2) The mother's day of birth you entered is invalid. The day of birth can only contain numeric characters and must be between 01 and 31.	mothers_birth_day	ss11page2

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.2 Form SS11, 2nd page	Mother's Month	TextField	2	Two digit number, 01 to 12	YES	Error	(1) Please enter the mother's month of birth. (2) The mother's month of birth you entered is invalid. The month of birth can only contain numeric characters and must be between 01 and 12.	mothers_birth_month	ss11page2
	Mother's Year	TextField	2	Four digit number 1900 to today	YES	Error	(1) Please enter the mother's year of birth. (2) The mother's year of birth you entered is invalid. The year of birth can only contain numeric characters and must be after 1900.	mothers_birth_year	ss11page2
				Combination of above fields		Error	The mother's birth date you entered is invalid. The date can not be in the future.		ss11page2
	Mother's Yes	RadioButton	1	Yes or No required, Y stored	YES			mother_citizen	ss11page2
	Mother's No	RadioButton	1	Yes or No required, N stored	YES				ss11page2

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.3 Form SS11, 3rd page	Married Yes	RadioButton	1	Yes or No required, Y stored	If Widowed/ Divorced Y			married	ss11page3
	Married No	RadioButton	1	Yes or No required, N stored	If Widowed/ Divorced Y				ss11page3
	Spouse's First Name	TextField	50	Alphanumeric and hyphen characters only	If Married Y	Error	(1) Please enter the spouse's first name. (2) The spouse's first name you entered is invalid. The first name can only contain alphanumeric characters and the following special character: hyphen (-).	spouses_first_name	ss11page3
	Spouse's Middle Name	TextField	50	Alphanumeric and hyphen characters only	If Married Y	Error	(1) The spouse's middle name is invalid. The middle name can only contain alphanumeric characters and the following special character: hyphen (-).	spouses_middle_name	ss11page3
	Spouse's Last Name	TextField	50	Alphanumeric and hyphen characters only	If Married Y	Error	(1) Please enter the spouse's last name. (2) The spouse's last name you entered is invalid. The last name can only contain alphanumeric characters and the following special character: hyphen (-).	spouses_last_name	ss11page3

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.3 Form SS11, 3rd page	Spouse's Birthplace	TextField	40	Alpha characters only, comma and apostrophe allowed	If Married Y	Error	(1) Please enter the spouse's place of birth. (2) The spouse's place of birth you entered is invalid. The place of birth can only contain alphanumeric characters and the following special characters: hyphen (-), apostrophe ('), and comma (,).	spouses_birthplace	ss11page3
	Spouse's Day	TextField	2	Two digit number, 01 to 31, validate on month	If Married Y	Error	(1) Please enter the spouse's day of birth. (2) The spouse's day of birth you entered is invalid. The day of birth can only contain numeric characters and must be between 01 and 31.	spouses_birth_day	ss11page3
	Spouse's Month	TextField	2	Two digit number, 01 to 12	If Married Y	Error	(1) Please enter the spouse's month of birth. (2) The spouse's month of birth you entered is invalid. The month of birth can only contain numeric characters and must be between 01 and 12.	spouses_birth_month	ss11page3

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.3 Form SS11, 3rd page	Spouse's Year	TextField	2	Four digit number 1900 to today	If Married Y	Error	(1) Please enter the spouse's year of birth. (2) The spouse's year of birth you entered is invalid. The year of birth can only contain numeric characters, and must be after 1900.	spouses_birth_year	ss11page3
				Combination of above fields		Error	The spouse's birth date you entered is invalid. The date can not be in the future.		ss11page3
	Spouse's Yes	RadioButton	1	Yes or No required, Y stored	If Married Y			spouse_citizen	ss11page3
	Spouse's No	RadioButton	1	Yes or No required, N stored	If Married Y				ss11page3
	Marriage Day	TextField	2	Two digit number, 01 to 31, validate on month	If Married Y	Error	(1) Please enter the day the passport applicant was married. (2) The day of marriage you entered is invalid. The day of marriage can only contain numeric characters, and must be between 01 and 31.	marriage_day	ss11page3
	Marriage Month	TextField	2	Two digit number, 01 to 12	If Married Y	Error	(1) Please enter the month the passport applicant was married. (2) The month of marriage you entered is invalid. The month of marriage can only contain numeric characters, and must be between 01 and 12.	marriage_month	ss11page3

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.3 Form SS11, 3rd page	Marriage Year	TextField	2	Four digit number 1900 to today	If Married Y	Error	(1) Please enter the year the passport applicant was married. (2) The year of marriage you entered is invalid. The year of marriage can only contain numeric characters, and must be after 1900.	marriage_year	ss11page3
				Combination of above fields		Error	The date of the most recent marriage you entered is invalid. The date can not be in the future.		ss11page3
	Widow/Divorce Yes	RadioButton	1	Yes or No required, Y stored	If Married Y	Error	(1) Please indicate whether the passport applicant is widowed or divorced.	widowDivorce	ss11page3
	Widow/Divorce No	RadioButton	1	Yes or No required, N stored	If Married Y	Error	(1) Please indicate whether the passport applicant is widowed or divorced.		ss11page3
	Widow/Divorce Day	TextField	2	Two digit number, 01 to 31, validate on month	If Widowed/Divorced Y	Error	(1) Please enter the day the passport applicant was widowed or divorced. (2) The day the passport applicant was widowed or divorced is invalid. The day can only contain numeric characters, and must be between 01 and 31.	widowDivorce_day	ss11page3

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.3 Form SS11, 3rd page	Widow/Divorce Month	TextField	2	Two digit number, 01 to 12	If Widowed/Divorced Y	Error	(1) Please enter the month the passport applicant was widowed or divorced. (2) The month the passport applicant was widowed or divorced is invalid. The month can only contain numeric characters, and must be between 01 and 12.	widowDivorce_month	ss11page3
	Widow/Divorce Year	TextField	2	Four digit number 1900 to today	If Widowed/Divorced Y	Error	(1) Please enter the year the passport applicant was widowed or divorced. (2) The year the passport applicant was widowed or divorced is invalid. The year can only contain numeric characters, and must be after 1900.	widowDivorce_year	ss11page3
				Combination of above fields can not be in the future.		Error	The widowed or divorced date you entered is invalid. The date can not be in the future.		ss11page3
	Other Names You Have Used 1 - First	TextField	20	Alphanumeric and hyphen characters only	NO	Error	(1) The first other name used by the passport applicant is invalid. The name can only contain alphanumeric characters and the following special character: hyphen (-).	other_names_1_first	ss11page3

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.3 Form SS11, 3rd page	Other Names You Have Used 1 - Middle	TextField	20	Alphanumeric and hyphen characters only	NO	Error	(1) The first other name used by the passport applicant is invalid. The name can only contain alphanumeric characters and the following special character: hyphen (-).	other_names_1_middle	ss11page3
	Other Names You Have Used 1 - Last	TextField	20	Alphanumeric and hyphen characters only	NO	Error	(1) The first other name used by the passport applicant is invalid. The name can only contain alphanumeric characters and the following special character: hyphen (-).	other_names_1_last	ss11page3
	Other Names You Have Used 2	TextField	50	Alphanumeric and hyphen characters only	NO	Error	(1) The second other name used by the passport applicant is invalid. The name can only contain alphanumeric characters and the following special character: hyphen (-).	other_names_2_first	ss11page3
	Other Names You Have Used 2 - Middle	TextField	20	Alphanumeric and hyphen characters only	NO	Error	(1) The second other name used by the passport applicant is invalid. The name can only contain alphanumeric characters and the following special character: hyphen (-).	other_names_2_middle	ss11page3
	Other Names You Have Used 2 - Last	TextField	20	Alphanumeric and hyphen characters only	NO	Error	(1) The second other name used by the passport applicant is invalid. The name can only contain alphanumeric characters and the following special character: hyphen (-).	other_names_2_last	ss11page3

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.3 Form SS11, 3rd page	Other Names You Have Used 3 - First	TextField	50	Alphanumeric and hyphen characters only	NO	Error	(1) The third other name used by the passport applicant is invalid. The name can only contain alphanumeric characters and the following special character: hyphen (-).	other_names_3_first	ss11page3
	Other Names You Have Used 3 - Middle	TextField	20	Alphanumeric and hyphen characters only	NO	Error	(1) The third other name used by the passport applicant is invalid. The name can only contain alphanumeric characters and the following special character: hyphen (-).	other_names_3_middle	ss11page3
	Other Names You Have Used 3 - Last	TextField	20	Alphanumeric and hyphen characters only	NO	Error	(1) The third other name used by the passport applicant is invalid. The name can only contain alphanumeric characters and the following special character: hyphen (-).	other_names_3_last	ss11page3
	Passport Yes	RadioButton	1	Yes or No required, Y stored	YES			passport_issued	ss11page3
	Passport No	RadioButton	1	Yes or No required, N stored	YES				ss11page3
	Most Recent Passport Number	TextField	8	Numeric only	If Passport Y	Error	(1) Please enter the most recent passport number. (2) The most recent passport number you entered is invalid. The passport number can only contain numbers.	recent_passport_num	ss11page3

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.3 Form SS11, 3rd page				Validate passport number with external passport number database on entering and exiting page.		Warning	(1) The most recent passport number is not a valid passport number issued by the Passport Approval Agency.		ss11page3
	Issue Day	TextField	2	Two digit number, 01 to 31, validate on month	If Passport Y	Error	(1) Please enter the day the most recent passport was issued. (2) The day the most recent passport was issued is invalid. The day can only contain numeric characters, and must be between 01 and 31.	issue_day	ss11page3
	Issue Month	TextField	2	Two digit number, 01 to 12	If Passport Y	Error	(1) Please enter the month the most recent passport was issued. (2) The month the most recent passport was issued is invalid. The month can only contain numeric characters, and must be between 01 and 12.	issue_month	ss11page3
	Issue Year	TextField	2	Four digit number 1900 to today	If Passport Y	Error	(1) Please enter the year the most recent passport was issued. (2) The year the most recent passport was issued is invalid. The year can only contain numeric characters, and must be after 1900.	issue_year	ss11page3

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.3 Form SS11, 3rd page				Combination of above fields can not be in the future.		Error	The approximate issue date you entered is invalid. The date can not be in the future.		ss11page3
	Submitted	RadioButton	6	Submitted, Lost or Stolen required, SUBMITTED stored	If Passport Y	Error	(1) Please indicate whether the most recent passport was submitted, lost or stolen.	recent_passport_status	ss11page3
	Lost	RadioButton	6	Submitted, Lost or Stolen required, LOST stored	If Passport Y	Error	(1) Please indicate whether the most recent passport was submitted, lost or stolen.		ss11page3

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.4 Form SS11, 4th page	Proof of citizenship	CheckBox	1	Boolean 1 or 0	If Over Age 14	Error	(1) Please indicate whether you received and reviewed the passport applicant's proof of citizenship.	proof_of_citizenship	ss11page4
	Proof of identity	CheckBox	1	Boolean 1 or 0	If Over Age 14	Error	(1) Please indicate whether you received and reviewed the passport applicant's proof of identity.	proof_of_identity	ss11page4
	Evidence of child's citizenship	CheckBox	1	Boolean 1 or 0	If Under Age 14	Error	(1) Please indicate whether you received and reviewed the child's proof of citizenship.	childs_citizenship	ss11page4
	Evidence of child's relationship to parent	CheckBox	1	Boolean 1 or 0	If Under Age 14	Error	(1) Please indicate whether you received and reviewed evidence of the child's relationship to the parent.	child_parent_relationship	ss11page4
	Parental identification	CheckBox	1	Boolean 1 or 0	If Under Age 14	Error	(1) Please indicate whether you received and reviewed the parental identification.	parental_id	ss11page4
	Driver's license	CheckBox	1	Boolean 1 or 0	Driver or Passport or Other	Error	(1) Please indicate the type of parental identification received.	drivers_license	ss11page4
	Passport	CheckBox	1	Boolean 1 or 0	Driver or Passport or Other	Error	(1) Please indicate the type of parental identification received.	passport	ss11page4
	Issue Day	TextField	2	Two digit number, 01 to 31, validate on month	If Driver, Passport, Other checked	Error	(1) Please enter the day the parental identifying document was issued. (2) The day the parental identifying document was issued is invalid. The day can only contain numeric characters, and must be between 01 and 31.	id_issue_day	ss11page4

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.4 Form SS11, 4th page	Issue Month	TextField	2	Two digit number, 01 to 12	If Driver, Passport, Other checked	Error	(1) Please enter the month the parental identifying document was issued. (2) The month parental identifying document was issued is invalid. The month can only contain numeric characters, and must be between 01 and 12.	id_issue_month	ss11page4
	Issue Year	TextField	2	Four digit number 1900 to today	If Driver, Passport, Other checked	Error	(1) Please enter the year the parental identifying document was issued. (2) The year the parental identifying document was issued is invalid. The year can only contain numeric characters, and must be after 1900.	id_issue_year	ss11page4
				Combination of above fields		Error	The parental identifying document issue date you entered is invalid. The date can not be in the future.		ss11page4
	Number	TextField	20	Numeric only	If Driver, Passport, Other checked	Error	(1) Please enter the number of the parental identifying document. (2) The number of the parental identifying document you entered is invalid. The number can only contain numbers.	id_num	ss11page4

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.4 Form SS11, 4th page	Expiration Day	TextField	2	Two digit number, 01 to 31, validate on month	If Driver, Passport, Other checked	Error	(1) Please enter the day the parental identifying document will expire. (2) The day the parental identifying document will expire is invalid. The day can only contain numeric characters, and must be between 01 and 31.	id_expire_day	ss11page4
	Expiration Month	TextField	2	Two digit number, 01 to 12	If Driver, Passport, Other checked	Error	(1) Please enter the month the parental identifying document will expire. (2) The month parental identifying document will expire is invalid. The month can only contain numeric characters, and must be between 01 and 12.	id_expire_month	ss11page4
	Expiration Year	TextField	2	Four digit number after 1900	If Driver, Passport, Other checked	Error	(1) Please enter the year the parental identifying document will expire. (2) The year the parental identifying document will expire is invalid. The year can only contain numeric characters, and must be after 1900.	id_expire_year	ss11page4

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.4 Form SS11, 4th page	Name on Document	TextField	50	Alphanumeric and hyphen characters only	If Driver, Passport, Other checked	Error	(1) Please enter the name shown on the other parental identifying document. (2) The name shown on the parental identifying document is invalid. The name can only contain alphanumeric characters and the following special character: hyphen (-).	id_name_on_doc	ss11page4

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.6 Form SS91, 1st page	Applicant's First Name	TextField	50	Alphanumeric and hyphen characters only	YES	Error	(1) Please enter the passport applicant's first name. (2) The first name you entered is invalid. The first name can only contain alphanumeric characters and the following special character: hyphen (-).	applicants_first_name	ss91page1
	Applicant's Middle Name	TextField	50	Alphanumeric and hyphen characters only	NO	Error	(1) The middle name you entered is invalid. The middle name can only contain alphanumeric characters and the following special character: hyphen (-).	applicants_middle_name	ss91page1
	Applicant's Last Name	TextField	50	Alphanumeric and hyphen characters only	YES	Error	(1) Please enter passport applicant's last name. (2) The last name you entered is invalid. The last name can only contain alphanumeric characters and the following special character: hyphen (-).	applicants_last_name	ss91page1
	Male	RadioButton	1	Male or Female required, M stored	YES			applicant_gender	ss91page1
	Female	RadioButton	1	Male or Female required, F stored	YES				ss91page1

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.6 Form SS91, 1st page	Passport Number	TextField	8	Numeric only	YES	Error	(1) Please enter the passport number. (2) The passport number you entered is invalid. The passport number can only contain numbers.	applicant_passport_num	ss91page1
				Validate name and passport number with Most Recent Passport Number on SS11 form		Error	(1) The passport number you entered does not match the passport number entered on Form SS11.		ss91page1
	Issue Day	TextField	2	Two digit number, 01 to 31, validate on month	YES	Error	(1) Please enter the day the most recent passport was issued. (2) The day the most recent passport was issued is invalid. The day can only contain numeric characters, and must be between 01 and 31.	applicant_passport_issue_day	ss91page1
	Issue Month	TextField	2	Two digit number, 01 to 12	YES	Error	(1) Please enter the month the most recent passport was issued. (2) The month the most recent passport was issued is invalid. The month can only contain numeric characters, and must be between 01 and 12.	applicant_passport_issue_month	ss91page1

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.6 Form SS91, 1st page	Birth Day	TextField	2	Two digit number, 01 to 31, validate on month	YES	Error	(1) Please enter the passport applicant's day of birth. (2) The day of birth you entered is invalid. The day of birth can only contain numeric characters, and must be between 01 and 31.	applicant_birth_day	ss91page1
	Birth Month	TextField	2	Two digit number, 01 to 12	YES	Error	(1) Please enter the passport applicant's month of birth. (2) The month of birth you entered is invalid. The month of birth can only contain numeric characters, and must be between 01 and 12.	applicant_birth_month	ss91page1
	Birth Year	TextField	2	Four digit number 1900 to today	YES	Error	(1) Please enter the passport applicant's year of birth. (2) The year of birth you entered is invalid. The year of birth can only contain numeric characters, and must be after 1900.	applicant_birth_year	ss91page1
				Combination of above fields		Error	The date of birth you entered is invalid. The date can not be in the future.		ss91page1

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.6 Form SS91, 1st page	Address line 1	TextField	30	Alphanumeric, comma, and hyphen characters only	YES	Error	(1) Please enter the passport applicant's address. (2) The first line of the address you entered is invalid. The address can only contain alphanumeric characters and the following special characters: hyphen (-) and comma (.).	applicant_address 1	ss91page1
	Address line 2	TextField	30	Alphanumeric, comma, and hyphen characters only	NO	Error	(1) The second line of the address you entered is invalid. The address can only contain alphanumeric characters and the following special characters: hyphen (-) and comma (.).	applicant_address 2	ss91page1
	Address line 3	TextField	30	Alphanumeric, comma, and hyphen characters only	NO	Error	(1) The third line of the address you entered is invalid. The address can only contain alphanumeric characters and the following special characters: hyphen (-) and comma (.).	applicant_address 3	ss91page1
	Address line 4	TextField	30	Alphanumeric, comma, and hyphen characters only	NO	Error	(1) The fourth line of the address you entered is invalid. The address can only contain alphanumeric characters and the following special characters: hyphen (-) and comma (.).	applicant_address 4	ss91page1

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.6 Form SS91, 1st page	Postal Code	TextField	10	Numbers, letters and hyphen only	YES	Error	(1) Please enter a postal code for the address. (2) The postal code you entered is invalid. The postal code can only contain alphanumeric characters and the following special character: hyphen (-).	applicant_postalCode	ss91page1
	Country	TextField	20	Alpha characters only	NO	Error	(1) The country you entered is invalid. The country can only contain letters and spaces.	applicant_country	ss91page1

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.7 Form SS91, 2nd page	How was passport lost or stolen?	TextField	255	Alphanumeric characters, comma, hyphen, period, and apostrophe only	YES	Error	(1) Please enter an explanation of how the passport was lost or stolen. (2) The explanation you entered of how the passport was lost or stolen is invalid. The explanation can only contain alphanumeric characters and the following special characters: hyphen (-), apostrophe ('), period (.), and comma (,).	passport_lostStolen	ss91page2
	Notified where and when	TextField	255	Alphanumeric characters, comma, hyphen, period, and apostrophe only	If Notified Y	Error	(1) Please enter an explanation of where and when the police authorities were notified of the stolen passport. (2) The explanation you entered of where and when the police authorities were notified is invalid. The explanation can only contain alphanumeric characters and the following special characters: hyphen (-), apostrophe ('), period (.), and comma (,).	stolen_where_when	ss91page2

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.7 Form SS91, 2nd page	What efforts were made to recover the passport?	TextField	255	Alphanumeric characters, comma, hyphen, period, and apostrophe only	YES	Error	(1) Please enter an explanation of the efforts made to recover the passport. (2) The explanation you entered of the efforts made to recover the passport is invalid. The explanation can only contain alphanumeric characters and the following special characters: hyphen (-), apostrophe ('), period (.), and comma (,).	effort_to_recover	ss91page2
	Certification	CheckBox	1	Boolean 1 or 0	YES	Error	(1) Please indicate that you certify that the information furnished herein was obtained from the person to whom the passport was issued, and is correct and complete to the best of your knowledge and belief.	certification	ss91page2

FlexFoundation Blueprint Application

Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.8 Assign Application	Optional comments	TextField	255	Alphanumeric characters, comma, hyphen, period, and apostrophe only	NO	Error	(1) The optional comments you entered are invalid. The comments can only contain alphanumeric characters and the following special characters: hyphen (-), apostrophe ('), period (.), and comma (,).	comments	assign
	Select the officer	DropDown	10		YES	Error	(1) Please select an officer to assign the passport application to.	selectOfficer	assign

FlexFoundation Blueprint Application


Business Rules Artifact

Wireframe Page	Label Text	Type of Page Element	Max Response Length	Business Rule	Required	Message Severity	Message Text	Page Element Name	Page Name
3.9 Approve Application	Optional comments	TextField	255	Alphanumeric characters, comma, hyphen, period, and apostrophe only	If Application Denied	Error	(1) Please enter an explanation for why the passport application was denied. (2) The explanation you entered for why the passport application was denied is invalid. The explanation can only contain alphanumeric characters and the following special characters: hyphen (-), apostrophe ('), period (.), and comma (,).	deniedComments	approve
	Select status	DropDown	10		YES	Error	(1) Please select the status of the passport application.	status	approve

Mockup for the PAAS

All requirements used to create the PAAS Blueprint application are captured in project artifacts. The Mockup artifact provides the design for a Page. This includes the specific colors used on the Page and the specific layout of text and Page Elements on the Page. Only one mockup was created for the PAAS. This mockup then provided the basis for designing all of the other Pages. The Mockup appears on the following page.

Mockup



Passport Application Approval System

A Blueprint for Business Transaction Manager Applications

Welcome to the Passport Application Approval System

This system can be used to review, edit, and approve applications for a new passport.

Registered Users

If you are already registered with the Passport Application Approval System, please enter your username and password below, then select **Log In** to proceed.

Username:

Password:
(case sensitive)

Select Language: ▼

Note: if you have forgotten your user name or password, please contact your system administrator.

Unregistered Users

If you are not registered with the Passport Application Approval System, you will need to create an username and password.

Note: you will need to enter your Agency ID to create a new username and password.

Packaged Software

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).

The Apache Software License, Version 1.1

Copyright © 2000 The Apache Software Foundation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: “This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).” Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.
4. The names “Apache” and “Apache Software Foundation” must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org.
5. Products derived from this software may not be called “Apache”, nor may “Apache” appear in their name, without prior written permission of the Apache Software Foundation.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation. For more information on the Apache Software Foundation, please see <http://www.apache.org>.

Portions of this software are based upon public domain software originally written at the National Center for Supercomputing Applications, University of Illinois, Urbana-Champaign.