temenos

Springboard

JOURNEY MANAGER

VERSION 24.10

TOC

Springboard overview

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10 Retail DAO 4.1This feature was updated in Retail DAO 4.1. SMB DAO 1.4This feature was updated in SMB DAO 1.4. Lending 1.1This feature was updated in Lending 1.1.

Introduction

Springboard solutions enable US banks to deliver an outstanding digital customer acquisition journey rapidly, using a ready-made solution on an extensible platform that grows with the financial institution. Standard Springboard can be implemented as the ready-made solution or can serve as a starting point for custom projects.

Several Springboard solutions are currently available.

- **Retail DAO:** The Retail Springboard solution for US deposit account opening (DAO[1]) enables onboarding[2] of new and existing customers for US direct deposit accounts (DDA[3]).
- **SMB DAO:**The Small Business Springboard solution for US deposit account opening allows users to open checking and savings business accounts.
- **Lending:** The Lending Springboard solution integrated with Temenos LMS for US loan and credit card applications enables lending for new and existing customers for US auto loans, personal loans, and credit cards.

Why choose a Springboard solution?

Springboard solutions provide a starting point for customer projects, offering an accelerated implementation schedule.

The standard Springboard solution provides value to our clients and implementation teams in several ways:

- Offers a ready-made, quick to market, DAO or Lending experience.
- Offers clients a head-start on developing a custom DAO or Lending solution.

---

[1]Deposit Account Opening (DAO). DAO bank forms allow users to open Checking, Savings and Money Market Accounts.
[2]The steps required to get a new customer integrated into a new program. These steps may vary business to business.
[3]Direct Deposit Account (DDA) is simply a checking or savings account which offers the ability to send and receive funds electronically.

- Offers implementation teams a framework for implementing a custom DAO or Lending solution.
- Demonstrates what a best-of-breed DAO or Lending solution should look like.
- Serves as the reference architecture for a Temenos Journey Manager DAO or Lending solution.

Pre-built components

Springboard solutions require far less development than a custom Temenos Journey Manager project. This is achieved by pre-building many of the typical components required to support US solutions on the Journey platform. Examples of pre-built components include:

- Maestro templates and forms
- Transact Functions including third-party integrations
- standard workflows and collaboration jobs to support manual review and decision-making.
- a product selector

To learn more about the components supplied with each Springboard solution, see Springboard components.

Getting started

To start using Springboard and get the most out of it, we recommend you learn about the following.

- **Temenos Journey Manager platform:** A platform for building, managing, and continuously improving onboarding journeys.
- **Journey Manager services:**TJM software modules configured to perform specific business-related functionality.
- **Journey Maestro:**TJM's customer experience development environment for building financial customer journeys.
- **Journey Narrative:** A new way of thinking about TJM application design that forms the basis of a Springboard solution.
- **Maestro SCM:** Integrate your Maestro projects with external source-code management (SCM) systems such as Git or SVN.
- **Git SCM:** A free and open-source distributed version control system.
- **Journey Workspaces:**TJM's authenticated review and approval portal for account opening and customer onboarding.
- **Journey Analytics:** Capture and analyze behavioral and completion data for applications hosted on the TJM platform.

Information is available on these subjects and more in this documentation and on the TJM resources website. We also offer online courses and instructor-led training on developing Temenos Journey Manager platform solutions.

Implementing a Springboard solution

While substantial portions of a Springboard solution are pre-built, client-specific work is still required for every implementation. Typically, the bulk of the client work is focused into three main areas:

- **Configuration/Setup:** Define style/brand, products, and other configurable options. Deploy and test components, and test integrations.
- **Core Banking Integration:** Supports account and customer creation, and other critical onboarding activities.
- **Customization:** Specific modifications to the project scope, added to the project statement of work (SOW) or via change requests (CR).

To learn more, see What clients need to know.

Integrated Development Environment

For Springboard development, you need an Integrated Development Environment (IDE) that supports Maven, Ant scripts, and writing services in Groovy. Eclipse and IntelliJ are two IDEs commonly used when building a Springboard project. Note that IntelliJ has native Groovy support while Eclipse requires a Groovy plugin.

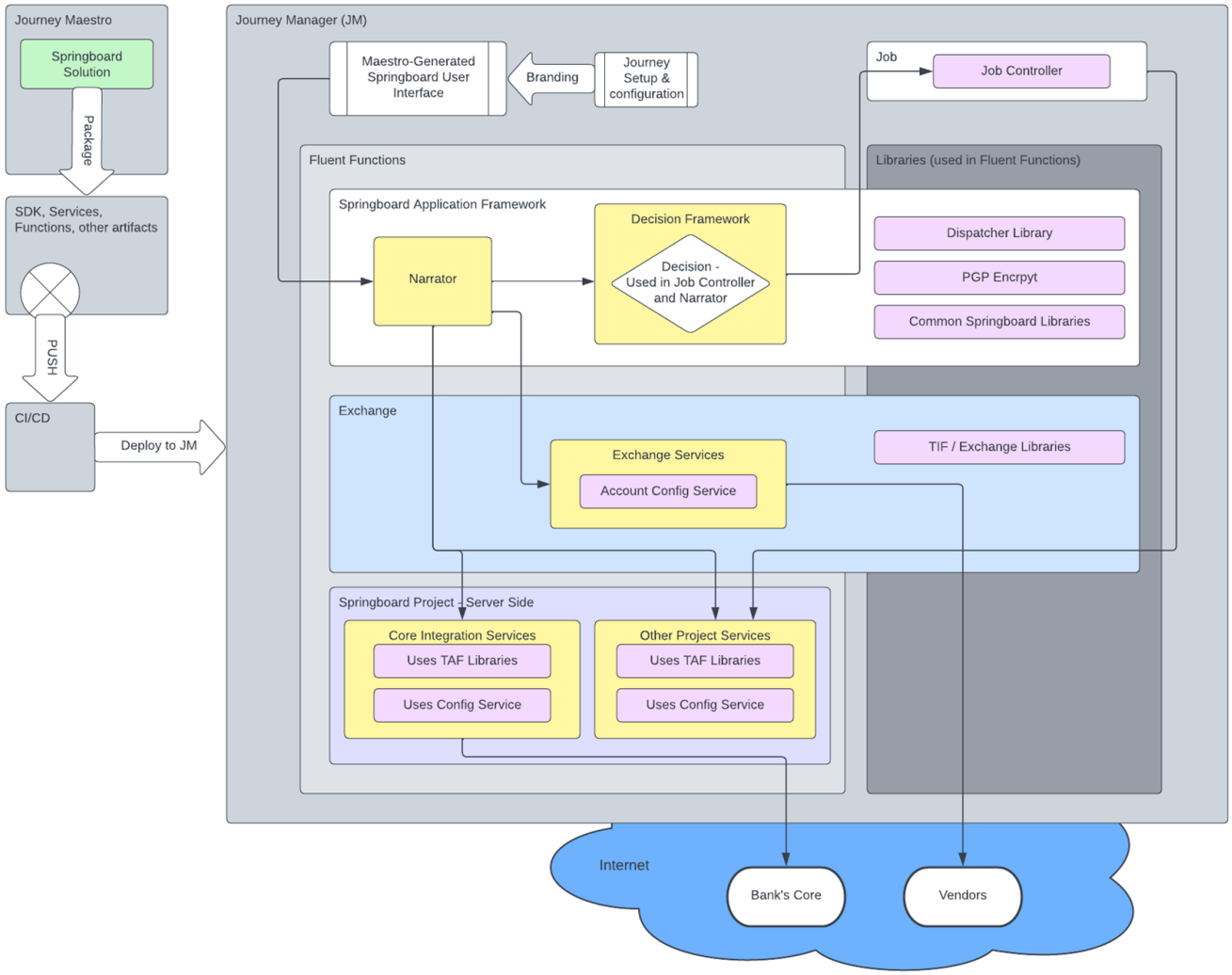
You also need to install Maven and a JDK. The JDK version must match the version installed on Journey Manager. To learn more, see Journey Manager 3rd Party Libraries > Java.

Next, learn about Springboard architecture.

Springboard architecture

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10 Retail DAO 4.1This feature was updated in Retail DAO 4.1. SMB DAO 1.4This feature was updated in SMB DAO 1.4. Lending 1.1This feature was updated in Lending 1.1.

All Springboard solutions are based on the standard Springboard architecture, shown in the following diagram.



Next, learn about Springboard configuration.

Springboard configuration

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10 Retail DAO 4.1This feature was updated in Retail DAO 4.1. SMB DAO 1.4This feature was updated in SMB DAO 1.4. Lending 1.1This feature was updated in Lending 1.1.

Springboard is a highly configurable product. Much of what you need to do to make a Springboard solution meet your needs can be done though configuration of the pre-built components included with a Springboard solution. However, you may have requirements that aren't handle by Springboard out of the box, and in these situations your Springboard solution can be customized to meet your requirements.

For a faster-to-market implementation that's easier to maintain and upgrade, Temenos recommends using a standard Springboard solution rather than a customized solution.

Configuration or customization?

It's important to carefully review the details in your Springboard statement of work (SOW) and supporting product documentation. Springboard projects are extremely proscriptive. Certain changes are permitted, while others are not. These constraints ensure solutions are implemented in a repeatable and cost-effective manner, as well as ensuring ease of support for future updates and maintenance.

We refer to permitted changes, which are included by default in all Springboard SOWs, as *configuration*. Certain other changes are still permitted but they require additional funding and must be explicitly added to the project scope via additions to the SOW or change requests (CR).

Other changes are unsupported, whether added to scope via a SOW change, CR, or any other means. If any unsupported changes are made, the solution is no longer eligible for upgrades and may not be supported by the Springboard Engineering team.

For these reasons, it is important to carefully consider any application changes, understand whether they're in scope, and assess what impact they might have on the support and maintenance of your Springboard solution.

NOTE

To determine whether a change is in scope, check your Springboard project documentation or contact your Springboard implementation team. If you have any doubt about whether a change is in scope, assume it isn't in scope until either you find a reference to the change in your project documentation or you receive confirmation from your Springboard implementation team that the change is in scope.

How features are offered

The major features and common requests are summarized below. At a glance, you can determine how each feature is offered: as a standard Springboard feature, via configuration, or via customization. Any customization will add to implementation cost and timeline.

Feature	Standard	Configuration	Customization
Product selector with shopping cart (Retail DAO and Lending only)		check	
Product catalog		check	
Styling, images, logos		check	
Status message content		check	
KYC questions		check	
Credit Union eligibility		check	
Application prefill with Prove	check		
Application prefill with Mitek	check		
Joint applications (Retail DAO and Lending)	check		
Device fraud check	check		
IDA/IDV	check		
Auto approve new accounts	check		
New deposit account funding via EFT	check		
New deposit account funding via debit or credit card	check		
New deposit account funding via internal transfer (existing customers only)	check		

Status emails		check	
Back-office support in Workspaces (except Lending)	check		
Save and resume	check		
Additional pages			check
Changing order of pages			check
Changing page logic or business rules			check
Alternative vendors for prefill, IDA/IDV, card processing, EFT account verification			check
Altering page content or layout			check

Springboard product limits

A Springboard solution can include several products, defined by the client during project initiation by completing a product specification. This specification is used by the implementation partner when configuring the product catalog. The definition of each product in the product catalog includes the product type, the available options, any funding requirements, and disclosures.

While there is no hard limit to the number of product types or products in a product catalog, consider the following recommendations for the best user experience.

- **Product types:** We recommend not exceeding 10 product types, with 3 to 5 product types considered ideal.
- **Products:** As you might expect, the number of products is related to the number of product types. We recommend not exceeding 25 products.

Next, learn about Springboard components.

Springboard components

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10 Retail DAO 4.1This feature was updated in Retail DAO 4.1. SMB DAO 1.4This feature was updated in SMB DAO 1.4. Lending 1.1This feature was updated in Lending 1.1.

Every Springboard solution consists of multiple components, all of which work together to provide the required functionality. The common components, such as the Narrator and Exchange components, are listed below. For more information, including to learn about components specific to each Springboard solution, check the project documentation for each component.

Implementing a Springboard solution requires familiarity with Journey Manager and its framework for creating services, Journey Maestro, Journey Narrative, Maestro SCM, and Git. Information is available on these subjects and more in this documentation and on the TJM resources website. We also offer online courses and instructor-led training on developing Temenos Journey Manager platform solutions.

Forms

Every Springboard solution is comprised of several forms, each of which represents a different chunk of functionality. Some of these forms are used by all Springboard solutions, being designed and built for each specific Springboard solution, while other forms are specific to a solution. The table below identifies which forms are used by each Springboard solution, and a description of each form follows.

Form	Retail DAO	SMB DAO	Lending
Product Selector Form (optional)	check		check
DAO Form	check		
SMB DAO Form (Primary Applicant)		check	
SMB Owners DAO Form (Owners/Signatories)		check	
Lending Form			check
Deposits Form	check	check	
Review Form	check	check	
Email Unsubscribe Form	check	check	check

Product selector form

The product selector form allows bank applicants to view product details and select one (or many) products that they would like to apply for. The shopping cart functionality makes it clear what the applicant has selected and then transitions them seamlessly to the next form to complete their application. Banks can enforce mandatory or optional bundles, configure the 'Learn More' hyperlinks and restrict the number of products in the shopping cart, if required. All the product information displayed in the product selector comes from the configured product catalog.

DAO Form

This is the Deposit Account Opening form which collects the personal and financial details of the applicant. The application is run through the Alloy Decision Engine and may be approved, denied, or placed in a review queue. This form is the primary entry point into the direct deposit account application and is accessed by bank new account applicants.

SMB DAO Form

This is the Deposit Account Opening form which collects personal details of the primary applicant, and business details. The application is run through the Alloy Decision Engine on submission and may be approved, denied, or placed in a review queue. This form is the primary entry point into the application for the Primary Applicant.

SMB Owners DAO Form

This is the Deposit Account Opening form which collects personal and business details of any Owners or Signatories that were identified by the Primary Applicant. The application is run through the Alloy Decision Engine on submission and may be approved, denied, or placed in a review queue. This form is the primary entry point into the application for any Owners / Signatories.

Deposits Form

error
Making this form optional is a customization.

When an applicant has been approved for a new account, they're typically expected to deposit some initial funds into the account. This form allows the user to fund the new account using Credit, Debit, ACH, Internal Transfer (from an account at the same bank), or Check by Mail.

In the standard US Retail DAO, Credit, Debit and Electronic Funds Transfer (EFT) funds transfers are all processed using the Payroc payment processing platform. In addition to Payroc, an integration with Plaid Auth allows applicants using EFT to retrieve and verify their account details by logging into online banking, rather than typing bank account details. Entering bank account details manually is an option for the applicant.

Internal Transfers are performed by the Core Banking API and are allowed for existing bank customers.

For more information on available funding methods, see Springboard funding options.

Review Form

When applications are sent to the Manual or Fraud Review queues they need to be manually assessed and a decision is made by a staff member of the bank.

The Review Form is used by bank staff to perform these review operations to request more information and make a final Approved/Declined decision for the application.

Email Unsubscribe Form

This form allows the user to unsubscribe from nurture email notifications, such as application reminder emails. It will not unsubscribe them from transactional emails required to provide details about their application or account(s), such as an Approved Confirmation or Declination email.

Journey Narrative

Every Springboard solution is based on the Application Framework and relies on Narratives to control the flow and actions of each of the forms. As a result, the Narration Controller (Narrator) must be installed and active on the Journey Manager servers hosting the solution.

Springboard is preconfigured to include Narratives services in your deployment and comes with a predefined Narrative that includes all of the standard pages and actions required.

> error
> Making changes to the Narrative is a customization.

To learn more about Journey Narrative, see Journey Narrative.

Decision Engine

When a user is applying for a Deposit Account, the Decision Engine can be used to calculate the eligibility of the applicant. The Decision Engine can access the transaction and form data of the application and use this data to determine whether an application should be accepted, rejected, or placed in a review queue. For example, the application can be rejected if the user's credit score is too low.

For more information, see Decision Framework.

Solution services project

US Retail DAO project

This project provides the services required for the Springboard Retail DAO form, as well as the Core Banking API. The Narrative provided is designed to call each of these services as required to support the application.

The Fluent Functions in the US DAO project can do things such as:

- Product data pre-fill
- Customer and Account creation
- Validating the product's minimum and maximum funding configuration
- Fetching customer data to be prefilled
- SFTP delivery of complete applications
- ACH Account verification
- Money movement via Payroc

SMB DAO project

This project provides the services required for the Springboard DAO form, as well as the Core Banking API. The Narrative provided is designed to call each of these services as required to support the application.

The Fluent Functions in the SMB DAO project can do things such as:

- Product data pre-fill
- Applicant IDV/IDA through Alloy integration
- Populating application used by Workspaces
- Fetching customer data to be prefilled

US Lending project

This project provides the services required for the Springboard Lending form, as well as the Lifecycle Management Suite (LMS) API. The Narrative provided is designed to call each of these services as required to support the application.

The Fluent Functions in the US Lending project can do things such as:

- Product data pre-fill
- Application create and update in LMS
- SFTP delivery of complete applications
- Payroc

Workflow (Collaboration Jobs) and Testing Form

Each Springboard solution comes with a predefined Collaboration Job Definition (workflow). This workflow controls what happens to the application once it is submitted, such as forwarding the application to one of the Workspaces queues or sending it on for delivery.

This workflow should not be modified for standard implementations.

Maestro template

The Springboard Maestro template serves as a starting point for configurations and customizations. The template implements our current UX best practices and provides client-side support for the Narrator.

The template can be configured in various ways. Colors, backgrounds, fonts and images can be easily configured without major modifications. The template can also be customized more deeply depending on a customer's specific needs, but these deeper customizations are distinctly custom development and may result in the customer's implementation forking from the base, making it no longer eligible for upgrades.

Exchange services and third-party integrations

Several exchange services and integrations from third-party vendors are used by the Springboard solutions.

error
Use of alternate vendors is a customization.

Exchange service	Retail DAO	SMB DAO	Lending
Alloy: Provides fraud detection using device signatures and metadata. Also provides personal fraud risk.	check	check	check
Amazon SNS: Performs multi-factor authentication using SMS when resuming applications.	check	check	check
Google Places: Provides autocomplete address suggestions to select from as an applicant types an address.	check	check	check
iovation: Provides fraud detection using device signatures and metadata.	check	check	check
Mitek License Pre-fill: Prefills applicant data using driver license. May be used instead of or as an alternative to Prove Pre-fill.	check	check	check
Payroc: Process credit card, debit card, and electronic funds transfer to fund a new account.	check	check	
Plaid Auth: Verifies the applicant's bank account prior to electronic funds transfer.	check	check	
Prove Pre-fill: Prefills applicant data using mobile phone number and last four digits of applicant SSN.	check	check	check

Next, learn about the standard DAO funding options.

Standard DAO funding options

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10 Retail DAO 4.1This feature was updated in Retail DAO 4.1. SMB DAO 1.4This feature was updated in SMB DAO 1.4.

Standard Springboard US Retail DAO and US SMB DAO offer several funding methods. The funding methods offered to the approved applicant are dependent on the funding methods defined for the chosen products. All products must be funded from the same source. The funding method along with minimum and maximum deposit amounts per funding method must be defined as part of the product catalog. For details, see the Product Specification.

The following standard DAO funding methods are available to be configured for US Retail DAO and US SMB DAO solutions.

- **Credit Card / Debit Card:** The processes for funding with credit card or debit card are identical. This method of funding requires an agreement to be established between the financial institution and Payroc.
- **Electronic Funds Transfer:** Springboard DAO uses Plaid Auth to authenticate an applicant's bank account at another financial institution. This method of funding requires an agreement to be established between the financial institution and Plaid.
- **Mail a Check:** This option allows the applicant to send a check to the bank directly. The page will only list one address, which may not be a local branch. If you don't want to offer this option, be sure to mark this option as `false` in the product configuration.
- **Internal Transfer:** Internal transfer is allowed for existing deposit account holders only. The applicant can choose from a list of their existing accounts at the financial institution for transfer of funds to the new accounts.

Next, learn about core integrations.

Integration with core banking systems

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10 Retail DAO 4.1This feature was updated in Retail DAO 4.1. SMB DAO 1.4This feature was updated in SMB DAO 1.4. Lending 1.1This feature was updated in Lending 1.1.

All Springboard solutions integrate with core banking systems.

Retail DAO and SMB DAO

The Springboard Retail DAO and SMB DAO solutions can integrate with any core banking system. However, it must be noted that every implementation of a core banking system is unique and development work is required by the implementation partner. Therefore, it is critical to ensure your core APIs are available. For more details regarding core integrations, see your project documentation.

Lending with LMS

Springboard Lending with Temenos Lifecycle Management Suite (LMS) leverages LMS to integrate with the core as, generally, this connection is already established.

Next, learn about use cases for Springboard solutions.

Use cases for Springboard solutions

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10 Retail DAO 4.1This feature was updated in Retail DAO 4.1. SMB DAO 1.4This feature was updated in SMB DAO 1.4. Lending 1.1This feature was updated in Lending 1.1.

Every Springboard solution is intended to be a ready-made solution that enables onboarding of new and existing customers by a client financial institution (FI). Springboard solutions are not intended to be used without the basic level of customization documented in the client specification documents. The use cases covered below are supported by standard Springboard.

Use case	Retail DAO	SMB DAO	Lending
Select products	check		check
Apply for new retail deposit account	check		
Apply for new retail deposit account in-branch	check		
Apply for new SMB deposit account		check	
Apply for new SMB deposit account in-branch		check	
Apply for new loan			check
Auto-approve	check	check	check
Auto-decline	check	check	check
Manual review	check	check	check
Manual approve	check	check	check
Manual decline	check	check	check
Fund new deposit account	check	check	
Resume application	check	check	check

Use case: Select products

Scenario

Applicant may review available products, learn more, and then select one or more products to apply for before continuing to the DAO application.

Actor

New or existing customer / member

Preconditions

The implementation includes the product selector form. If the product selector form is not included, product selection will occur on the FI website with products passed to the DAO application.

Postconditions

- User has selected one or more products to apply for.
- Transition seamlessly to the DAO form.

Use case: Apply for new retail deposit account

Scenario

Apply for a new personal Direct Deposit account offered by the financial institution using a mobile or desktop device.

Actor

New or existing customer / member

Preconditions

Product has been selected on the FI website or in the product selector form.

Alternate flow: Joint applicant

Collect co-applicant details and submit for IDA/IDV.

Alternate flow: Save

Applicant elects to save the application and resume later. Requires an email and mobile phone number.

Alternate flow: Cancel

Applicant elects to cancel the application.

Exception: Unrecoverable error

An error occurs without a known means of recovery and the application is sent to the Onboarding Error queue in Workspaces. Applicant must start a new application.

Exception: Review

The application is flagged for review based upon applicant information and third-party results. A message is displayed to the applicant. The application is sent to the Review queue in Workspaces. A review email is sent to the primary applicant. See Review Use Case.

Exception: Decline

The application is declined based on applicant information and third-party results.

Postconditions

- Applicant has provided their details.
- Applicant has been submitted for IDA/IDV.
- Applicant has accepted disclosures.
- Application has been decisioned.

Use case: Apply for new SMB deposit account

Scenario

Apply for a new Direct Deposit account for business offered by the financial institution using a mobile or desktop device.

Actor

New business customer.

Preconditions

Product has been selected on the FI website or in the product selector form.

Alternate flow: Beneficial owner

Collect Beneficial Owner details via the SMB Owners form and submit for IDA/IDV.

Alternate flow: Signatory

Collect account Signatory details via the SMB Owners form and submit for IDA/IDV.

Alternate flow: Save

Applicant elects to save the application and resume later. Requires an email and mobile phone number.

Alternate flow: Cancel

Applicant elects to cancel the application.

Exception: Unrecoverable error

An error occurs without a known means of recovery and the application is sent to the Onboarding Error queue in Workspaces. Applicant must start a new application.

Exception: Review

The application is flagged for review based upon applicant information and third-party results. A message is displayed to the applicant. The application is sent to the Review queue in Workspaces. A review email is sent to the primary applicant. See Review Use Case.

Exception: Decline

The application is declined based on applicant information and third-party results.

Postconditions

- Applicant has provided their details.
- Applicant has been submitted for IDA/IDV.
- Applicant has accepted disclosures.
- Application has been decisioned.
- All Beneficial Owner forms are complete.
- All Signatory forms are complete.

Use case: Auto-approve

Scenario

Automatically approve applications based on data collected and information from third party vendors.

Actor

Decision Engine.

Preconditions

- Data collected from the applicants.
- Data from third party vendors indicate approval based upon rules.

Postconditions

- Application approved.
- A confirmation is displayed to the applicant.

Use case: Auto-decline

Scenario

Decline an application based on information gathered in the application and on information from third party vendors.

Actor

Decision Engine.

Preconditions

Decision rules have been established to determine if an applicant is declined.

Postconditions

- A declined message is displayed to the applicant.
- Non-FCRA or FCRA decline email sent to the applicant.

Use case: Approve, deny, or send to manual review (automated)

Scenario

Approve, deny or send application to manual review workflow based on information gathered in the application and based on information from third party vendors. Customized Decision Rules are part of the customization available in a Base Springboard implementation and are defined in the Customization Specification document.

Actor

Financial Institution

Use case: Manual review

Scenario

Applications that receive a Manual Review decision from the automated Decision Rules are sent to a manual review workflow. This workflow uses Workspaces, the details of which are not covered here. After manual review, an application may be approved or

denied. When approved, the application workflow moves forward to the Fund New Accounts use case and workflow.

Actor

Financial Institution Employees

Preconditions

An application receives an automated decision of Manual Review.

Postconditions

- A message is displayed to the applicant.
- The application is sent to the Review queue in Workspaces.
- A review email is sent to the primary applicant.

Use case: Manual approve

Scenario

Manually approve applications in Manual Review based upon data collected and information from third party vendors.

Actor

Financial Institution Employees

Preconditions

- Data is collected from the applicants.
- Data is collected from third party vendors.
- The application is in the Manual Review queue in Workspaces.

Postconditions

- The application is approved.
- An approved email is sent to the applicant.

Use case: Manual decline

Scenario

Manually decline an application based on information gathered in the application and from third party vendors.

Actor

Financial Institution Employees

Preconditions

- Data is collected from the applicants.
- Data is collected from third party vendors.

- The application is in the Manual Review queue in Workspaces.

Postconditions

- A declined message is displayed to the applicant.
- A non-FCRA or FCRA decline email is sent to the applicant.

Use case: Fund new deposit account

Scenario

Allow the applicant to add funds to their new deposit accounts using a mobile or desktop web browser on their personal device. The applicant may add funds using a verified existing bank account, internal existing account, credit card, debit card, or by mailing a check. Funding is required in Standard Springboard DAO solutions.

> error
> Making funding optional is a customization.

Actor

New or existing customer/member.

Preconditions

Applicant has been approved for a new deposit account, either automatically or manually. If auto-approved, the applicant has transitioned seamlessly to the Deposits Form. If manually approved, the applicant has resumed the application at the Deposits Form.

Alternate flow: Credit or Debit Card

Applicant funds the new account with a card.

Alternate flow: Electronic Funds Transfer

Applicant funds the new account with a verified existing account at another institution.

Alternate flow: Internal Transfer

Existing customer/member funds the new account with an existing account at the institution.

Alternate flow: Mail a check

Applicant funds the new account by delivering a check via mail or in person.

Exception: Recoverable error

Problems encountered with the chosen funding method. Applicant allowed to select another method of funding the new account.

Exception: Unrecoverable error

An error occurs without a known means of recovery and the application is sent to the Onboarding Error queue in Workspaces. Applicant must start a new application.

Postconditions

- A new customer record is created (if new to FI) in core banking system.
- New accounts are created in the core banking system.
- Funding initiated.
- Application submitted.

Use case: Resume application

Scenario

Allow the applicant to resume the application after it has been saved, or they have been manually approved.

Actor

Applicant

Preconditions

- Application was saved - manually or automatically.
- Applicant has an email with a link to resume the application.
- Applicant provided a valid mobile phone number in the application.

Postconditions

Applicant returns to the application at the last page saved.

Next, learn about options for the DAO in-branch experience.

Springboard DAO In-branch Experience

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10 Retail DAO 4.1This feature was updated in Retail DAO 4.1. SMB DAO 1.4This feature was updated in SMB DAO 1.4.

The standard Springboard US Retail DAO and SMB DAO solutions support an in-branch experience allowing new customers to open a new account while sitting with a relationship manager at the Financial Institution. The In-branch Experience has the same workflow components as a self-service workflow.

There are two experiences with the in-branch workflow. With the first experience, the relationship manager initiates the application and hands off to the customer to complete on their own device at their convenience. With the second experience, the relationship manager initiates a session with the customer that allows the customer to acknowledge and accept the various disclosures while sitting across the desk from the relationship manager.

ERROR

Collecting wet signatures on disclosures or other documents requires custom development. It is not part of the standard Springboard in-branch workflow.

Feature	Self-service	In-branch	Customization
Product selector with shopping cart (retail DAO and Lending only)	check	check	
Product catalog	check	check	
Status message content	check	check	
KYC questions	check	check	
Credit Union eligibility	check	check	
Application prefill with Prove	check	check	
Application prefill with Mitek	check	Not recommended	
Joint applications (Retail DAO)	check	check	
Device fraud check	check		
IDA/IDV	check	check	

Auto approve new account	check	check	
New deposit account funding via EFT	check		
New deposit account funding via debit or credit card	check		
New deposit account funding via internal transfer (existing customers only)	check		
Mail a check / visit a branch funding	check		
Status emails	check	check	
Back-office support in Workspaces (except Lending)	check		
Save and resume	check		
Hand-off to customer		check	
Share session with customer		check	
Document wet signatures			check
DocuSign			check
Additional pages			check
Changing order of pages			check
Changing page logic or business rules			check
Alternative vendors for prefill, IDA/IDV, card processing, EFT account verification			check
Altering page content or layout			check

Next, learn about SFTP delivery of submitted applications.

SFTP delivery of submitted applications

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10 Retail DAO 4.1This feature was updated in Retail DAO 4.1. SMB DAO 1.4This feature was updated in SMB DAO 1.4. Lending 1.1This feature was updated in Lending 1.1.

Temenos will deliver a zip file package to an SFTP server nominated by the client. This end-point must be accessible from the Springboard production and staging environments.

Springboard allows you to configure the path in which the zip file will reside. The content of the zip file is pre-defined and includes the following artefacts.

- A PDF receipt for the main application form.
- All attachments added to the application process.
- All emails sent to the customer, provided as HTML files.
- The XML representation of the data entered by the user, or used by the application form and persisted.

Next, learn about what clients need to know.

What clients need to know about Springboard

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10 Retail DAO 4.1This feature was updated in Retail DAO 4.1. SMB DAO 1.4This feature was updated in SMB DAO 1.4. Lending 1.1This feature was updated in Lending 1.1.

Overview

This section provides context of the client's role to ensure implementation success. Each team member should be familiar with this section and the available online Springboard documentation.

If you have questions or need assistance, contact your Implementation Manager.

Client project team

The following client roles are key to project success. The most important is a single Product Manager who is empowered to make product decisions, avoiding iterative decision making, facilitating communication, and significantly reducing project risk:

Product Manager (In-scope requirements owner/Business Analyst)

The key owner of the "in-scope" business requirements which applies to the contracted work. The Product Manager ensures consistency through the delivery lifecycle to original scope and, if required, raises the appropriate change requests for impact assessment.

Project Manager

Responsible for managing the client deliverables throughout the duration of the project, including service/API specifications, third party integration account details, and specification documents.

Test Manager

- Responsible for determining and assuring the client's business requirements have been delivered by a Temenos Implementation Partner as required.
- Responsible for integration testing of the Temenos Implementation Partner deliverables into the client's enterprise.
- Works directly with the Temenos Implementation Partner delivery team to ensure any defects are triaged effectively, prioritized, fixed, and delivered back to the client.

Business Analyst

Works with the Product Owner and Temenos Implementation Partner to define, organize, and document requirements as needed. Also works with the Implementation Partner Business Analyst to assist in managing the Partner development workflow in JIRA.

Technical Lead
: A technical contact with responsibility for integration knowledge and ensuring the development readiness of the integrations.

UAT (User Acceptance Testing) team
: The client must have team members available to perform UAT and regression testing as required during the Transition to Live phase, and to organize and manage any additional testing methods required, such as penetration, load, and accessibility testing as needed.

Integrations

The following are standard Springboard integrations. It is critical to have agreements for these integrations in place prior to the start of implementation to ensure the implementation timeline can be met. The client is responsible for signing an agreement with the third-party vendors and obtaining a client key. Delays in having these integrations ready will delay the project delivery timeline.

error
Using alternate third-party vendors is a customization.

- **Alloy:** Provides fraud detection using device signatures and metadata. Also provides personal fraud risk.
- **Amazon SNS:** Performs multi-factor authentication using SMS when resuming applications.
- **Core APIs:** The client is responsible for ensuring their Core APIs are available for integration. At a minimum, the following APIs are required:
  - find person
  - create person
  - find accounts
  - create accounts
- **Google Places:** Provides autocomplete address suggestions to select from as an applicant types an address.
- **iovation:** Provides fraud detection using device signatures and metadata.
- **Mitek License Pre-fill:** Prefills applicant data using driver license. May be used instead of or as an alternative to Prove Pre-fill.
- **Payroc:** Process credit card, debit card, and electronic funds transfer to fund a new account.
- **Plaid Auth:** Verifies the applicant's bank account prior to electronic funds transfer.

- **Prove Pre-fill:** Prefills applicant data using mobile phone number and last four digits of applicant SSN.

Journey Flows

Become familiar with the standard journey flows provided by the Temenos Implementation Partner. These user journeys illustrate the standard user journeys.

error
Modification to the user journeys is a customization.

- Self-service journey
  - New to customer/member
  - Existing customer/member
- Funding journey
  - ACH
  - Credit/Debit card
  - Internal transfer
  - Mail a Check / Visit a Branch
- In-branch, fully assisted journey
  - New customer only

Journey flows are available as PDFs you can download (login required) for the following solution versions.

- Retail DAO 4.1
- Retail DAO 4.0
- Lending 1.0

Specification Documents

Become familiar with the specification documents provided as these will provide your implementation team with your configuration requirements. The client must complete these documents before development begins.

- **Product Specification:** Specify details of each product offered through the Springboard solution. This includes details such as effective dates, paperless statements, minimum funding amounts, supported funding types.
- **Visual Specification:** Specify styling such as colors and logos.

- **Customization Specification:** Specify URLs and provide content for the status messages such as review and decline.
- **Email Specification:** Specify tailored content for each of the status emails as well as timing and frequency of nurture emails.

Specification documents are available to download (login required) for the following solution versions.

- Retail DAO 4.1
- Retail DAO 4.0
- SMB DAO 1.4
- Lending 1.0

Next, learn about the Retail DAO solution.

Springboard solution: Retail DAO

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10 Retail DAO 4.1This feature was updated in Retail DAO 4.1.

The Retail Springboard solution for US Deposit Account Opening (DAO[1]) is a ready-to-use solution based on standard Springboard to enable onboarding[2] of new and existing customers for US Direct Deposit Accounts (DDAs[3]).

Standard Springboard

Springboard solutions enable US banks to deliver an outstanding digital customer acquisition journey rapidly, using a ready-made solution on an extensible platform that grows with the financial institution.

The following information can help you to learn about standard Springboard.

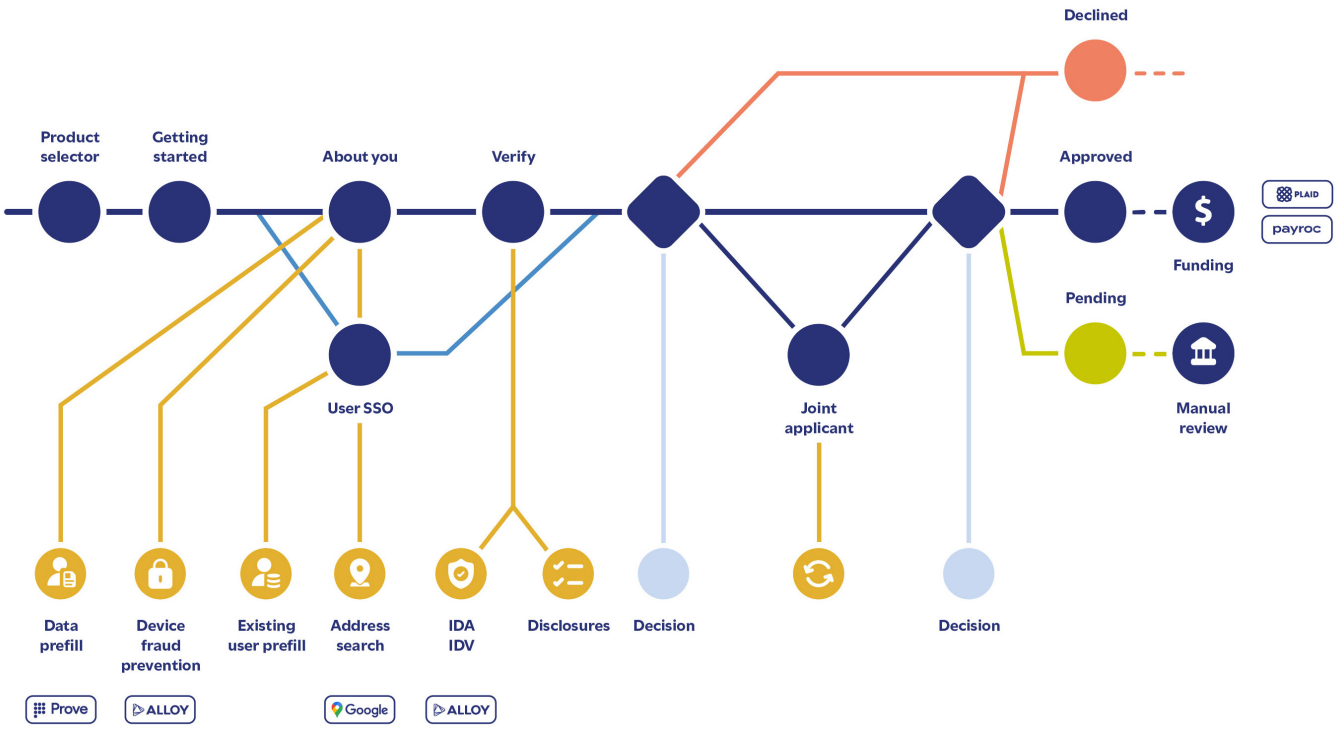
- Why choose a Retail DAO solution?
- Getting started
- Springboard architecture
- Springboard configuration
- Implementing a Springboard solution

Application Flow

A typical application flow for a Retail DAO Springboard solution is shown below.

error
Altering this application flow is a customization.

[1]Deposit Account Opening (DAO). DAO bank forms allow users to open Checking, Savings and Money Market Accounts.
[2]The steps required to get a new customer integrated into a new program. These steps may vary business to business.
[3]Direct Deposit Account (DDA) is simply a checking or savings account which offers the ability to send and receive funds electronically.



Use cases

The following use cases are supported by the Retail DAO solution.

- **Select products:** Review available products, then select one or more to apply for before continuing to the DAO application.
- **Apply for new retail deposit account:** Apply for a new personal Direct Deposit account using a mobile or desktop device.
- **Apply for new retail deposit account in-branch:** Apply for a new personal Direct Deposit account in person in a branch of the financial institution.
- **Auto-approve:** Automatically approve applications based on data collected and information from third party vendors.
- **Auto-decline:** Automatically decline an application based on data collected and information from third party vendors.
- **Manual review:** Automated decision rules may send applications to a manual review workflow where they may be approved or denied.
- **Manual approve:** Manually approve applications based on data collected and information from third party vendors.
- **Manual decline:** Manually decline an application based on data collected and information from third party vendors.
- **Fund new deposit account:** Add funds to a new Direct Deposit account using a mobile or desktop device.

- **Resume application:** Resume an application after it has been saved or manually approved.

Retail DAO components

The Springboard Retail DAO solution comprises multiple components, all of which work together to provide the required functionality. To learn about how these components relate to one another and interact in a Springboard solution, see Springboard solution architecture.

In addition to forms and Exchange services, the components used by the Retail DAO Springboard solution are:

- Journey Narrative
- Decision Engine
- US Retail DAO Project
- Workflow (Collaboration Jobs) and Testing Form
- Maestro template

Forms

Every Springboard solution is comprised of several forms, each of which represents a different chunk of functionality. The following forms are used by the Retail DAO Springboard solution.

- Product Selector Form (optional)
- DAO Form
- Deposits Form
- Review Form
- Email Unsubscribe Form

Exchange services and third-party integrations

The following Exchange services are used by the Retail DAO Springboard solution.

error
Choosing to utilize different third parties other than those listed below is a customization.

- **Alloy:** Provides fraud detection using device signatures and metadata. Also provides personal fraud risk.
- **Amazon SNS:** Performs multi-factor authentication using SMS when resuming applications.

- **Google Places:** Provides autocomplete address suggestions to select from as an applicant types an address.
- **Mitek License Pre-fill:** Prefills applicant data using driver license. May be used instead of or as an alternative to Prove Pre-fill.
- **Payroc:** Process credit card, debit card, and electronic funds transfer to fund a new account.
- **Plaid Auth:** Verifies the applicant's bank account prior to electronic funds transfer.
- **Prove Pre-fill:** Prefills applicant data using mobile phone number and last four digits of applicant SSN.

Implementing a Retail DAO solution

While substantial portions of a Springboard Retail DAO solution are pre-built, client-specific work is still required for every implementation. Typically, the bulk of the client work is focused into three main areas:

- **Configuration/Setup:** Define style/brand, products, and other configurable options. Deploy and test components, and test integrations.
- **Core Banking Integration:** Supports account and customer creation, and other critical onboarding activities.
- **Customization:** Specific modifications to the project scope, added to the project statement of work (SOW) or via change requests (CR).

To learn more, see What clients need to know.

Next, learn about the Retail DAO for credit unions.

Retail DAO for credit unions

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10 Retail DAO 4.1This feature was updated in Retail DAO 4.1.

Credit Unions have some unique requirements for opening Retail DAO accounts for new or existing customers. These requirements have been addressed by adding fully configurable options for eligibility requirements, beneficiary, and employment, as well as new logic to automatically add or remove a membership share to the applicant's shopping cart.

Eligibility

Every Credit Union has eligibility requirements unique to them. With the fully configurable Eligibility page, a Credit Union can specify simple to complex eligibility criteria. This page is automatically included for all Credit Unions and is placed early in the workflow to ensure the applicant meets eligibility before proceeding too far.

error
Moving this page to a different location in the workflow or changing aspects such as the layout or logic is a customization.

Beneficiary

Although created with Credit Unions in mind, adding a beneficiary to a deposit account is an optional feature available to all Springboard retail DAO clients implementing retail DAO v4.1 or later.

error
Moving this page to a different location in the workflow, or changing the layout, content, or logic is a customization.

Employment

While gathering employment information may be more common with lending, this page is now available as an optional feature to all Springboard DAO clients implementing retail DAO v4.1 or later. This page gathers employment status, salary, and occupation.

error
Moving this page to a different location in the workflow, or changing the layout, content, or logic is a customization.

Membership Share

New for Credit Unions with Retail DAO v4.1, membership share accounts will be added to or removed from an applicant's product list automatically as necessary. The membership share needs to be defined in the product catalog. The implementation team needs to know of any previously used membership share to determine if the applicant has an existing membership share. The applicant is advised if a membership share is added to or removed from their product list. The service structure is in place to call core APIs to find an applicant and their account assets. Development work is required by the implementation partner to make the connection and update the logic with the product IDs of all membership share accounts.

error
Moving the location of the series of core API calls, logic, or advisory is a customization.

Next, learn about the Retail DAO for authenticated customers.

Retail DAO for authenticated customers

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10 Retail DAO 4.1This feature was updated in Retail DAO 4.1.

The standard Springboard US Retail DAO solution supports authenticated customers of the financial institution.

Customer prefill enables your existing customers to complete the application quickly without the need for redundant data entry. This is done by retrieving existing data about the customer from the core banking system and prefilling it into the form to speed up the application process.

Standard functionality prefills personal details for an existing customer. The remainder of the workflow is identical to that of a new customer.

error
Changing the workflow or the prefill of more information for authenticated customers is a customization.

This functionality is optional. It is also dependent on your organization hosting a compatible web service to perform data retrieval and supporting customer SSO via online banking.

error
Developing the connection to the core API to enable existing customer prefill is a customization.

Next, learn about the SMB DAO Springboard solution.

Springboard solution: SMB DAO

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10 SMB DAO 1.4This feature was updated in SMB DAO 1.4.

The Small Business (SMB) Springboard solution for US Deposit Account Opening (DAO[1]) is a ready-to-use solution based on standard Springboard to enable onboarding[2] of new small and medium business customers for US Direct Deposit Accounts (DDAs[3]), allowing applicants to open checking and savings accounts.

Standard Springboard

Springboard solutions enable US banks to deliver an outstanding digital customer acquisition journey rapidly, using a ready-made solution on an extensible platform that grows with the financial institution.

The following information can help you to learn about standard Springboard.

- Why choose a SMB DAO solution?
- Getting started
- Springboard architecture
- Springboard configuration
- Implementing a Springboard solution

Use cases

The following use cases are supported by the SMB DAO solution.

- **Apply for new SMB deposit account:** Apply for a new business Direct Deposit account using a mobile or desktop device.
- **Apply for new SMB deposit account in-branch:** Apply for a new business Direct Deposit account in person in a branch of the financial institution.

---

[1]Deposit Account Opening (DAO). DAO bank forms allow users to open Checking, Savings and Money Market Accounts.
[2]The steps required to get a new customer integrated into a new program. These steps may vary business to business.
[3]Direct Deposit Account (DDA) is simply a checking or savings account which offers the ability to send and receive funds electronically.

- **Auto-approve:** Automatically approve applications based on data collected and information from third party vendors.
- **Auto-decline:** Automatically decline an application based on data collected and information from third party vendors.
- **Manual review:** Automated decision rules may send applications to a manual review workflow where they may be approved or denied.
- **Manual approve:** Manually approve applications based on data collected and information from third party vendors.
- **Manual decline:** Manually decline an application based on data collected and information from third party vendors.
- **Fund new deposit account:** Add funds to a new Direct Deposit account using a mobile or desktop device.
- **Resume application:** Resume an application after it has been saved or manually approved.

SMB DAO components

The Springboard SMB DAO solution comprises multiple components, all of which work together to provide the required functionality. To learn about how these components relate to one another and interact in a Springboard solution, see Springboard solution architecture.

In addition to forms and Exchange services, the components used by the SMB DAO Springboard solution are:

- Journey Narrative
- Decision Engine
- US SMB DAO Project
- Workflow (Collaboration Jobs) and Testing Form
- Maestro template

Forms

Every Springboard solution is comprised of several forms, each of which represents a different chunk of functionality. The following forms are used by the SMB DAO Springboard solution.

- SMB DAO Form (Primary Applicant)
- SMB Owners DAO Form (Owners/Signatories)
- Deposits Form
- Review Form
- Email Unsubscribe Form

Exchange services and third-party services

The following Exchange services are used by the SMB DAO Springboard solution.

error
Choosing to utilize different third parties other than those listed below is a customization.

- **Alloy:** Provides fraud detection using device signatures and metadata. Also provides personal fraud risk.
- **Amazon SNS:** Performs multi-factor authentication using SMS when resuming applications.
- **Google Places:** Provides autocomplete address suggestions to select from as an applicant types an address.
- **Payroc:** Process credit card, debit card, and electronic funds transfer to fund a new account.
- **Plaid Auth:** Verifies the applicant's bank account prior to electronic funds transfer.
- **Prove Pre-fill:** Prefills applicant data using mobile phone number and last four digits of applicant SSN.

Implementing a SMB DAO solution

While substantial portions of a Springboard SMB DAO solution are pre-built, client-specific work is still required for every implementation. Typically, the bulk of the client work is focused into three main areas:

- **Configuration/Setup:** Define style/brand, products, and other configurable options. Deploy and test components, and test integrations.
- **Core Banking Integration:** Supports account and customer creation, and other critical onboarding activities.
- **Customization:** Specific modifications to the project scope, added to the project statement of work (SOW) or via change requests (CR).

To learn more, see What clients need to know.

Next, learn about the Lending Springboard solution.

Springboard solution: Lending

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10 Lending 1.1This feature was updated in Lending 1.1.

The Lending Springboard solution for US loan and credit card applications is a ready-to-use solution based on standard Springboard to enable onboarding[1] of new and existing customers for US auto loans, personal loans, and credit cards. The Springboard Lending solution comes pre-integrated with Temenos LMS.

Standard Springboard

Springboard solutions enable US banks to deliver an outstanding digital customer acquisition journey rapidly, using a ready-made solution on an extensible platform that grows with the financial institution.

The following information can help you to learn about standard Springboard.

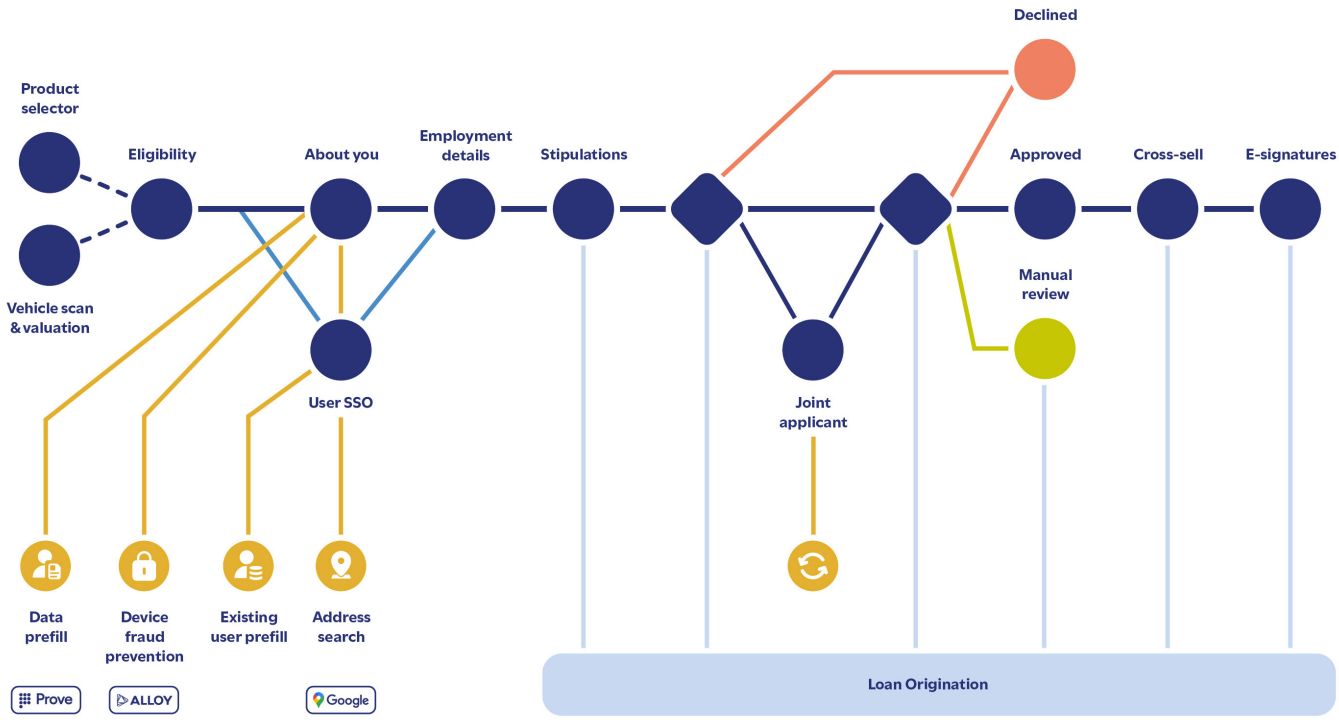
- Why choose a Lending solution?
- Getting started
- Springboard architecture
- Springboard configuration
- Implementing a Springboard solution

Application Flow

A typical application flow for a Lending Springboard solution is shown below.

error
Altering this application flow is a customization.

[1]The steps required to get a new customer integrated into a new program. These steps may vary business to business.



Use cases

The following use cases are supported by the Lending solution.

- **Select products:** Review available products, then select one or more to apply for before continuing to the DAO application.
- **Apply for new loan:** Apply for a new personal loan using a mobile or desktop device.
- **Auto-approve:** Automatically approve applications based on data collected and information from third party vendors.
- **Auto-decline:** Automatically decline an application based on data collected and information from third party vendors.
- **Manual review:** Automated decision rules may send applications to a manual review workflow where they may be approved or denied.
- **Manual approve:** Manually approve applications based on data collected and information from third party vendors.
- **Manual decline:** Manually decline an application based on data collected and information from third party vendors.
- **Resume application:** Resume an application after it has been saved or manually approved.

Lending components

The Springboard Lending solution comprises multiple components, all of which work together to provide the required functionality. To learn about how these components relate to one another and interact in a Springboard solution, see Springboard solution architecture.

In addition to forms and Exchange services, the components used by the Lending Springboard solution are:

- Journey Narrative
- Decision Engine
- US Lending Project
- Workflow (Collaboration Jobs) and Testing Form
- Maestro template

Forms

Every Springboard solution is comprised of several forms, each of which represents a different chunk of functionality. The following forms are used by the Lending Springboard solution.

- Product Selector Form (optional)
- Lending Form
- Email Unsubscribe Form

Exchange services and third-party integrations

The following Exchange services are used by the Lending Springboard solution.

error
Choosing to utilize different third parties other than those listed below is a customization.

- **Alloy:** Provides fraud detection using device signatures and metadata. Also provides personal fraud risk.
- **Amazon SNS:** Performs multi-factor authentication using SMS when resuming applications.
- **Google Places:** Provides autocomplete address suggestions to select from as an applicant types an address.
- **iovation:** Provides fraud detection using device signatures and metadata.
- **Mitek License Pre-fill:** Prefills applicant data using driver license. May be used instead of or as an alternative to Prove Pre-fill.

- **Prove Pre-fill:** Prefills applicant data using mobile phone number and last four digits of applicant SSN.

Implementing a Lending solution

While substantial portions of a Springboard Lending solution are pre-built, client-specific work is still required for every implementation. Typically, the bulk of the client work is focused into three main areas:

- **Configuration/Setup:** Define style/brand, products, and other configurable options. Deploy and test components, and test integrations.
- **Core Banking Integration:** Supports account and customer creation, and other critical onboarding activities.
- **Customization:** Specific modifications to the project scope, added to the project statement of work (SOW) or via change requests (CR).

To learn more, see What clients need to know.

Journey Setup project

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature relates to the 23.10 release

This topic contains instructions and guidelines for Springboard developers relating to the Journey Setup project. It is assumed you have access to the springboard/journey-setup GitLab project (login required).

It is also assumed you are familiar with the requirements for developing solutions on the Temenos Journey Manager platform and dependent software. Information is available on these subjects and more in this documentation and on the TJM resources website. We also offer online courses and instructor-led training on developing Temenos Journey Manager platform solutions.

Contributions

Changes to the Journey Setup project are managed using a GitFlow branch/merge strategy. If you want to contribute any changes, submit them to the `develop` branch as a feature branch, via a Git merge request or pull request.

Getting started

Prerequisites

The minimum JDK and Maven versions required for this project are defined in the Jenkins file (login required).

Maestro form

In Maestro, the Journey Setup form can be found under the Springboard organization, CMS project.

Default brand

The default brand definition files stored under org-properties are a starting point for your brand. After deploying the Journey Setup application in Temenos Journey Manager, follow the instructions below.

Build the Journey Setup project

To build the Journey Setup project, use one of the following commands.

- **Unix:** `build.sh`
- **Windows:** `build.cmd`

> NOTE
>
> Always build locally and resolve any issues before pushing any changes to your Git repo.

Organization properties

Journey Setup generates brand information and stores it in organization properties in Journey Manager. To learn about how to create and maintain organization properties, see Configure Organization Properties.

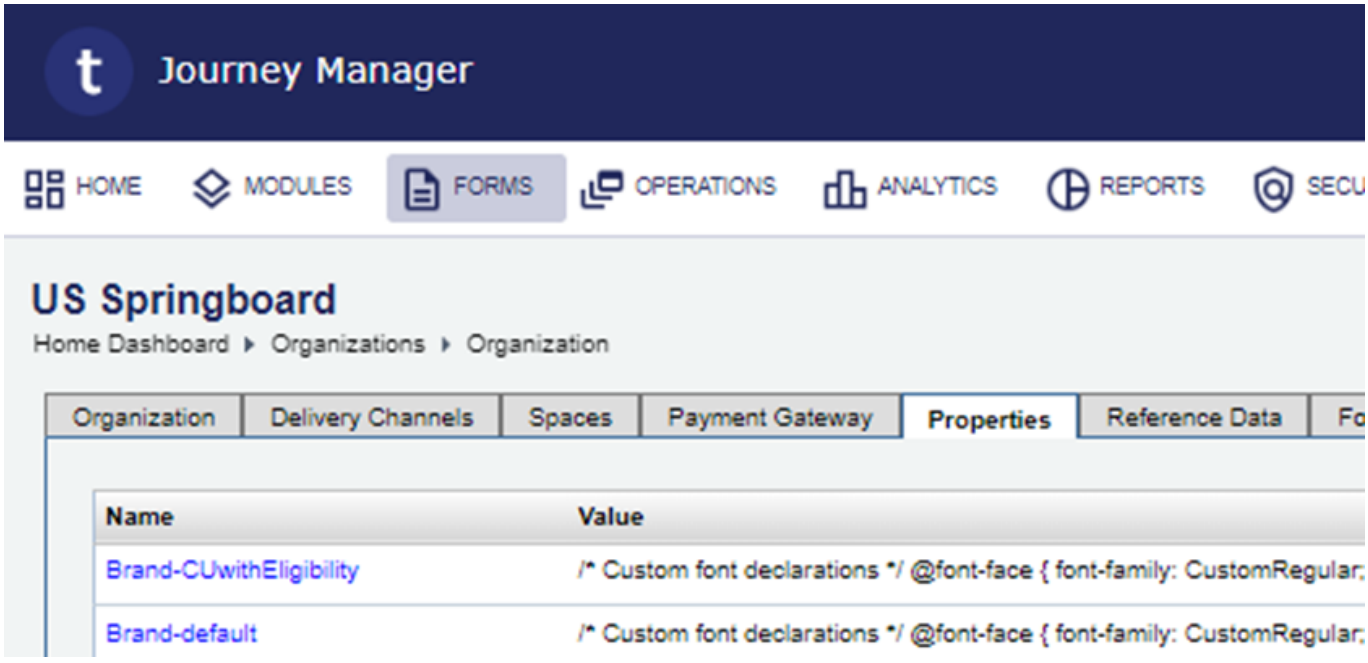
Brand properties

Journey Setup generates two properties for each brand that relate to style. One property stores the brand CSS, and the other stores the saved XML from the Journey Setup tool used to generate the CSS. The names of these properties use the following naming conventions:

- **CSS:** `Brand-`*`BRANDNAME`*
- **XML:** `BrandXML-`*`BRANDNAME`*

Replace *BRANDNAME* with the name of the brand.

The image below shows example CSS organization properties for the default and CUwithEligibility brands.



NOTE

Take care when manually editing or deleting any of these properties to ensure they have valid values.

Brand email template properties

Journey Setup generates an email template file organization property for each brand. The property name uses the naming convention `EmailTemplate-`*`BRANDNAME`*, replacing *`BRANDNAME`* with the name of the brand.

The image below shows example email template properties for the default and CUwithEligibility brands.

Journey Setup data properties

The Journey Setup tool also updates the following organization properties.

- `springboardCountryData`: Includes the following country-specific data: `country`, `iso`, `currency`, `code`, `maestroLang`.
- `springboardCountryLangData`: Includes the following country language-specific data: `country`, `language`, `label`, `maestroLang`.
- `springboardBrandData`: Includes the following brand-related data: `label` (brand's display name), `value`, `bankName`, `bankAddress`, `bankFotterLinks`.

The example below shows a `springboardBrandData` property with content for the default and CUwithEligibility brands.

```
[
    {
        "label": "Default",
        "value": "default",
        "bankName": "Temenos",
        "bankAddress": "www.temenos.com",
        "bankFotterLinks": "[{\"label\":\"About Us\",\"url\":\"ht-
tps://www.temenos.com/\"},{\"label\":\"Terms and con-
ditions\",\"url\":\"https://www.temenos.com/legal-information/website-terms-and-
conditions/\"},{\"label\":\"Privacy Policy\",\"url\":\"https://www.temenos.com/legal-
information/privacy-policy/\"}]"
    },
    {
        "label": " CUwithEligibility",
        "value": " CUwithEligibility",
        "bankName": " CUwithEligibility",
        "bankAddress": "www.temenos.com",
        "bankFotterLinks": "[{\"label\":\"About Us\",\"url\":\"ht-
tps://www.temenos.com/\"},{\"label\":\"Terms and con-
ditions\",\"url\":\"https://www.temenos.com/legal-information/website-terms-and-
conditions/\"},{\"label\":\"Privacy Policy\",\"url\":\"https://www.temenos.com/legal-
information/privacy-policy/\"}]"
    }
]
```


This `springboardBrandData` property results in the following organization properties being defined:

- `Brand-default`
- `Brand-CUwithEligibility`
- `BrandXML-default`
- `BrandXML-CUwithEligibility`

Reference data

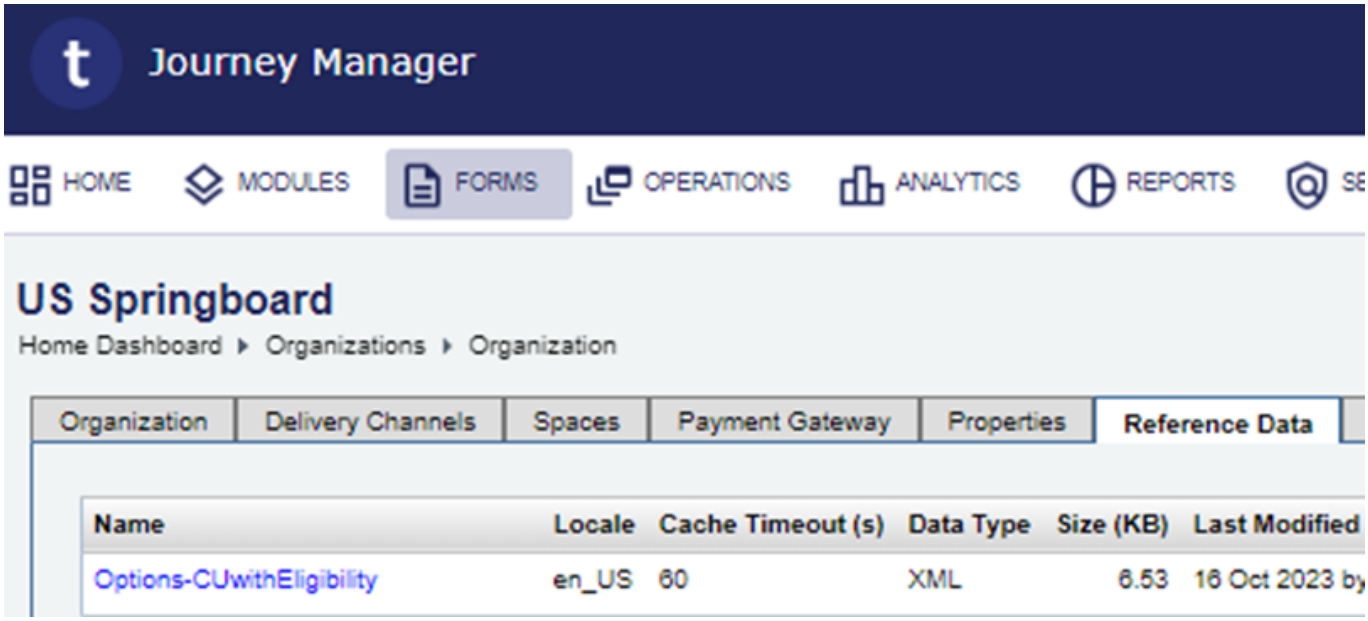
Page content

Credit Union Member Eligibility content, created in the **Content Management Tools > Page Content > Member Eligibility** section of the Journey Setup tool, is created for combinations of Brand+Locale, one at a time, and stored in Journey Manager as reference data. To learn more about reference data, see Configure Organization Reference Data.

You can generate page content for multiple Brand+Locale combinations, each of which is stored as a separate reference data item. The reference data item names use the naming convention `Options-`*`BRANDNAME`*, replacing *`BRANDNAME`* with the name of the brand.

You can have multiple reference data items with the same name, as long as they have different locales. Locale is specified using the format *`LANGUAGE`*`_`*`COUNTRY`* where *`LANGUAGE`* is a language code (often two lowercase letters), and *`COUNTRY`* is a region code (often two uppercase letters) qualifying the language for a specific country, territory or region. For example, the locale `en_US` corresponds to the English language in the United States, while `en_GB` corresponds to the English language in Great Britain.

The image below shows example page content reference data for the CUwithEligibility brand on the Organization > Reference Data tab in Journey Manager.



Deleting a brand

When a brand is no longer required, you can delete it.

To delete a brand:

1. Delete the properties `Brand-`*`BRANDNAME`* and `BrandXML-`*`BRANDNAME`*.

   For example, if your brand is called CUwithEligibility, you need to delete the properties `Brand-CUwithEligibility` and `BrandXML-CUwithEligibility`.

2. Edit the `springboardBrandData` property and delete the JSON section that includes “`value`”: “*`BRANDNAME`*”.

For example, if your brand is called CUwithEligibility, you need to delete the JSON section that includes `“value”: “CUwithEligibility”`.

> INFO
>
> When deleting a section from JSON data, take care to also remove an unnecessary comma on the preceding section.

3. Optional. If Page Content has been defined, delete all reference data items with the name `Options-`*`BRANDNAME`*, noting that there may be more than one. To learn how to delete an organization property, see Configure Organization Reference Data.

   For example, if your brand is called CUwithEligibility, delete the reference data item with the Name `Options-CUwithEligibility`.

Journey Setup for credit unions

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature relates to the 23.10 release

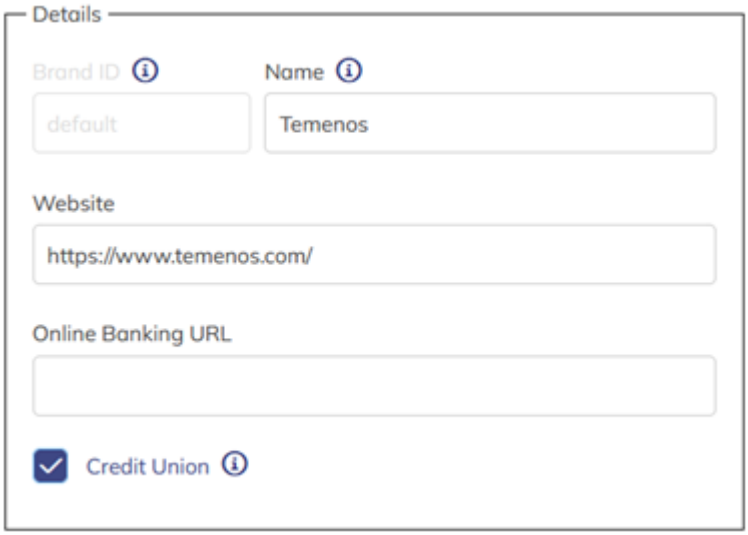
This topic expands on the information provided in Retail DAO for credit unions.

It is assumed you have access to the springboard/journey-setup GitLab project (login required), and that you are familiar with the requirements for developing solutions on the Temenos Journey Manager platform and dependent software. Information is available on these subjects and more in this documentation and on the TJM resources website. We also offer online courses and instructor-led training on developing Temenos Journey Manager platform solutions.

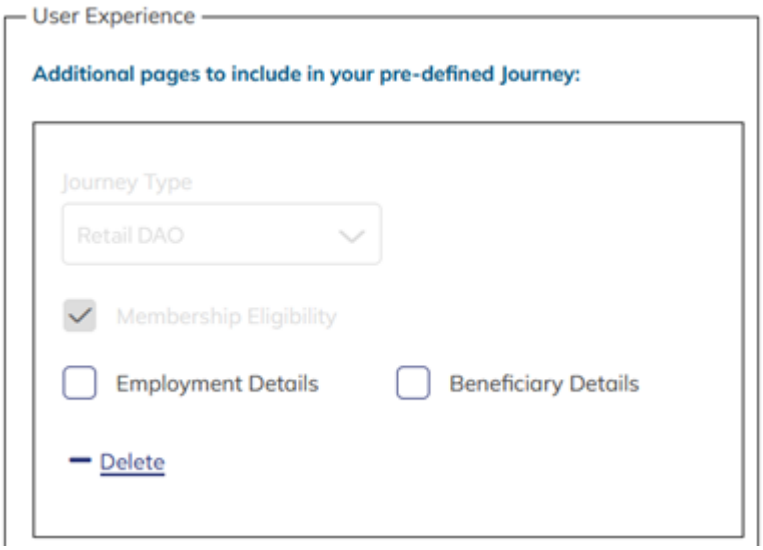
Define the brand

The first step in configuring a credit union is to define the brand as a credit union. This automatically enables the Eligibility page and membership share account requirements.

In Journey Setup, navigate to Brand Tools > Edit Brand. Under the Details section, select Credit Union as shown below.



Once you have done this, Membership Eligibility appears preselected and read only in the User Experience section as shown below.



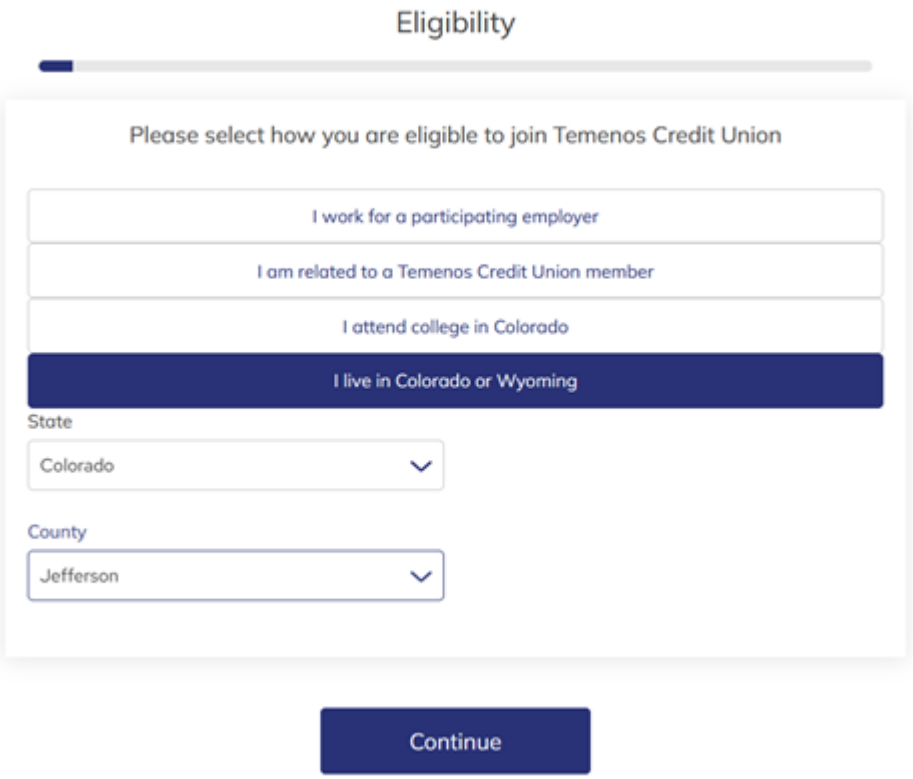
Membership eligibililty

In order to configure the Eligibility page, you must select Is Credit Union as described in the previous section. All credit unions have requirements of their members when joining the credit union, and each credit union has different criteria depending on their charter. To accommodate credit unions' varying requirements, the entire Eligibility page is configurable through Journey Setup. The client is responsible for providing their eligibility criteria in the US Retail DAO Customization Specification. Permissions are required to access this and other specification documents.

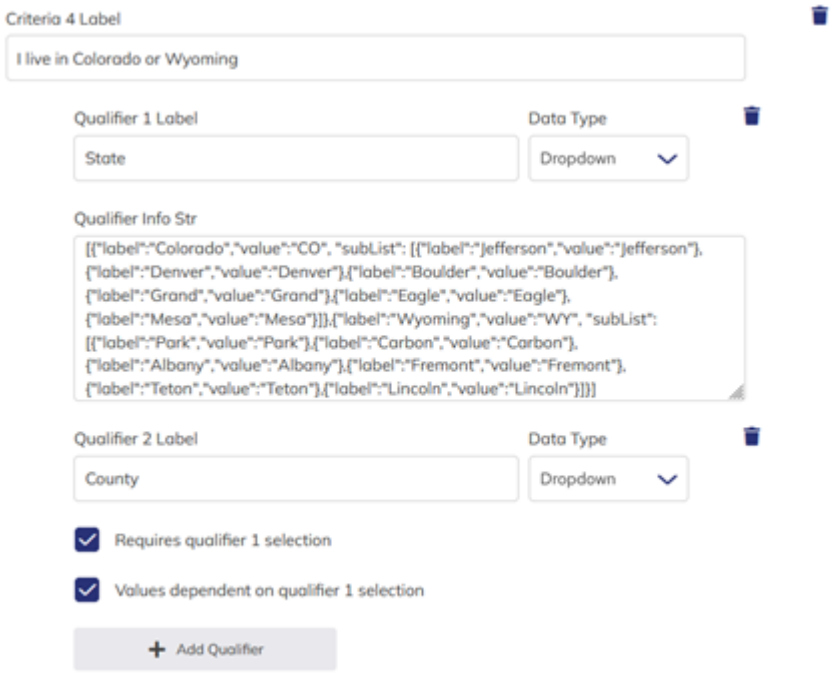
Navigate to Content Management Tools > Page Content > Manage Page Content. Currently, the only option is the Eligibility page. Here, you can define the page content, eligibility criteria, and any qualifiers for each criterion. You can define up to 10 criteria and, optionally, up to 3 qualifiers for each criterion. The platform does not validate whether the applicant really does meet the criteria to become a member.

error
Validation of credit union eligibility requirements is a customization.

The following image shows an example Eligibility page with 4 criteria and the last criteria selected. In this example, the applicant lives in Jefferson County Colorado. The first drop down selection (Colorado) determines the values in the second dropdown (a list of Colorado counties). This is an example of two dependent qualifiers where the first determines the options for the second.



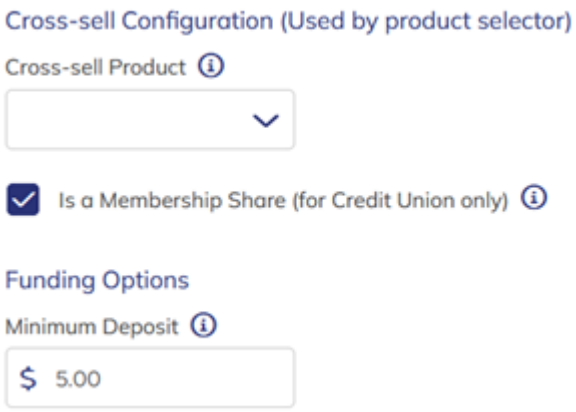
The following image shows the configuration for the selected criteria in the previous example.



Membership share account

One of the products offered for credit unions is a membership share account. This can either be selected by the applicant or automatically added by the platform.

Define the membership share account in the product catalog within Journey Setup. Navigate to Content Management Tools > Product Catalog > Manage Product Catalog then add the membership share as you would any other product. Under the Cross-sell Configuration section, select Is a Membership Share (for Credit Union only) as shown in the following image.



The client is responsible for completing the Springboard Product Configuration (login required) that provides all the data you need to complete the product definition. Ensure the account type for the membership share is Savings.

INFO

Access to specification documents is restricted. To learn more, see What clients need to know about Springboard > Specification documents.

How Springboard uses membership share product

The membership share will appear under Savings in the product selector. The applicant may select this alone or in combination with another product. The applicant is not required to select a membership share. After the applicant has provided their person details, Springboard queries the core banking system to check for a person record. If a person record is found, Springboard queries the core banking system again to retrieve any account assets. The results are examined for any account matching current or legacy membership share product IDs. For this reason, it is critical that the credit union provides any legacy membership share product IDs, and you must add these legacy IDs to the code.

Code logic

Credit Union Membership Share Account processing in the Product Fetch service performs the following actions:

- Retrieve all of the products to scan for Credit Union Membership share Account product.
- If any products are found, assign the share account Id as the value of the transaction property membershipShareAccountId.
- Update the Product Ids list and set flags for handling messaging about adding or removing the share account as needed.
- In the Narrative, the Membership Share Account page condition uses the flag `/Root/AvokaSmartForm/MembershipShare/Notice` set in this process.

Business logic

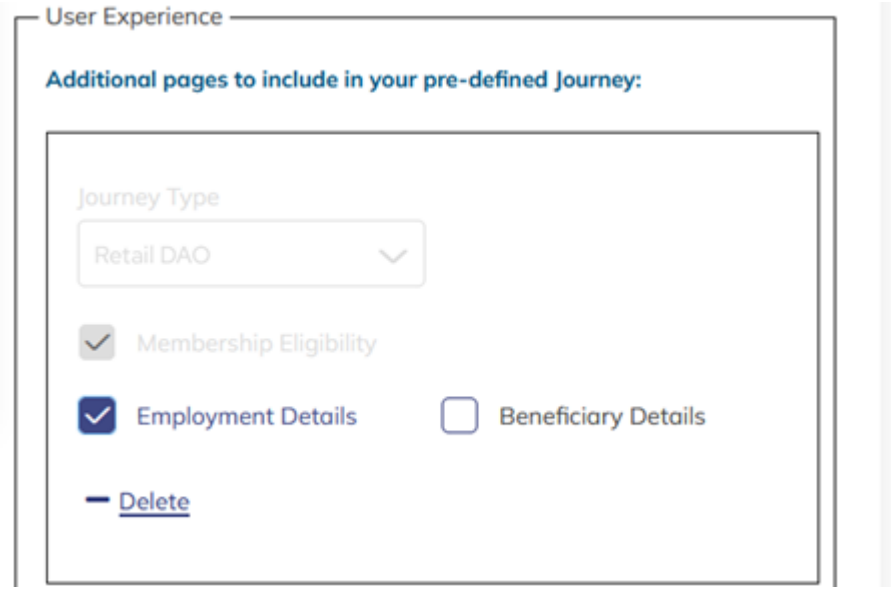
- If a person record is not found and a membership share is not present in the applicant shopping cart, a membership share will be automatically added, and the applicant advised. The applicant will be required to fund the membership share account with the minimum amount as defined in the product catalog.
    - If membership share is in the applicant shopping cart, another is not added.
- If a person record is found but there is no membership share in their current assets, and a membership share is not present in the applicant shopping cart, a membership share will be automatically added, and the applicant advised. The applicant will be required to fund the membership share account with the minimum amount as defined in the product catalog.
    - If membership share is in the applicant shopping cart, another is not added.
- If person record found and membership share in their current assets, and a membership share is present in the applicant shopping cart, the membership share will be automatically removed, and the applicant advised.
    - If membership share is not in the shopping cart, one is not added.
    - If membership share is the only item in the shopping cart, the application ends and the applicant is advised.

For more detail, see the Membership Share subprocess journey flow. Permissions are required to access journey flow diagrams.

Employment Details

The Employment Details page is an optional page available to all clients. When enabled, this page appears in the journey after Residential Address.

In Journey Setup, navigate to Brand Tools > Edit Brand. Under the User Experience section, select Employment Details as shown in the image below.



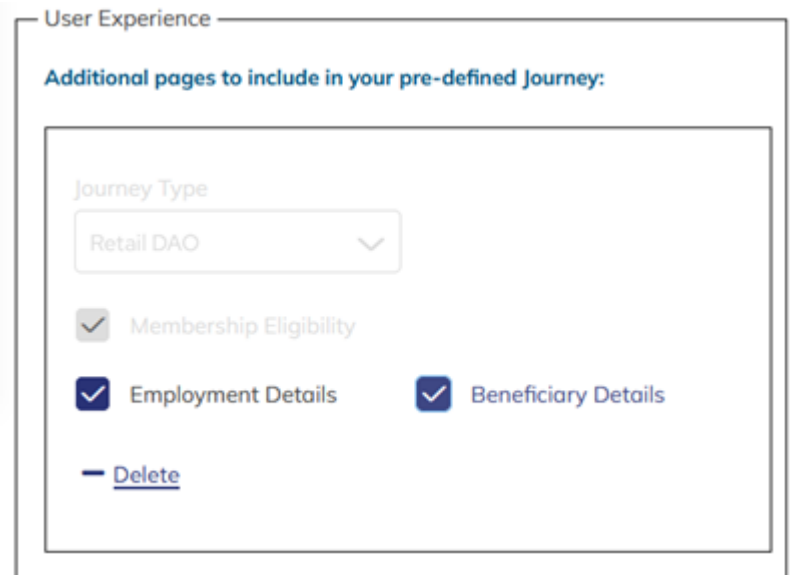
No further action is required. The page appears in the journey flow automatically after Residential Address.

error
Changing the content, page logic, or location in the journey flow is a customization.

Beneficiary

Account Beneficiary is an optional page available to all clients. When enabled, the Account Beneficiary page appears in the journey after Employment Details if that page is enabled, otherwise after Residential Address. The applicant is allowed to add none, one or two account beneficiaries.

In Journey Setup, navigate to Brand Tools > Edit Brand. Under the User Experience section, select Beneficiary Details as shown in the following image.



No further action is required. The page appears in the journey flow automatically after Employment Details or Residential Address.

error
Changing the content, page logic, or location in the journey flow is a customization.

Decision Framework

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10

The Decision Framework is a dynamic decision service driven by configuration rather than code. It allows for easy modification of the rules that leads to application decisions, without the need to write or deploy code.

This service is currently driven by JSON configuration files, with future plans for an editor user interface (UI).

These documents cover the following:

- Configuration drives the decision engine.
- JsonLogic drives the logic of decision paths.
- Data and TXN property map are available to JsonLogic to help make decisions.

Getting started

INFO

To use Decision Framework outside of Springboard, this guide assumes intermediate knowledge of the Temenos Journey Manager (TJM) platform, including:

- Journey SDK
- Fluent API
- SDK Developer Guides
- Fluent Functions
- Journey Maestro
- Journey Manager

If you're unfamiliar with any of these, we recommend you learn the basics of each before using the Decision Framework outside of Springboard.

The Decision Framework is intended for use with Journey Narratives. However, it can be used with any application that has Fluent Functions enabled.

Inputs and outputs

There are two primary inputs to the Decision Framework, the configuration that defines all of the decision rules and outcomes, and the data that is combined with the rules to determine results.

The results of the Decision Framework are called Destinations. Destinations act like flags, indicating which decision rules evaluated to `true` and what should be done with the application as a result.

An example result might look something like this:

```
{
  "destinations" : {
    "approved": {
      "isActive": false,
      "reasons": []
    },
    "manualReview": {
    "isActive": true,
      "reasons": [ "addressMismatch" ]
    },
    "declined": {
      "isActive": false,
      "reasons": []
    }
  },
  "calculatedDate": "2018-01-13T18:25:43.511Z"
}
```

Next, learn about configuring the Decision Framework.

Decision Framework: Configuration

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10

The decision engine is configured via decision logic in a JSON file comprised of two sections: destinations and paths.

Destinations

Destinations are the decisions that your configuration can arrive at. For example, you might have `"approved"`, `"denied"`, and `"pending"` destinations. You can have as many destinations as you need.

You must designate one, and only one, destination as the default destination. To do this, including the property `"isDefault": true` in the destination's configuration. If no paths return `true`, the default destination is chosen. If a path returns `true`, the default flag is ignored.

Decisions are represented in a JSON configuration as shown below. Note the first destination is the default destination in this example.

```
{
  "destinations": [
    {
      "id": "approved",
      "isDefault": true
    },
    {
      "id": "denied"
    }
  ],
  ...
}
```

Paths

Paths are the way you arrive at a destination. They consist of an `id`, a `destination` (which corresponds to a destination's id), and a JsonLogic `condition` property.

If `condition` evaluates to `true`, the `destination` will be selected when processing occurs.

- `id` can be any string you choose.
- `destination` must be the id of a destination which you configured in the `destinations` array.
- `condition` must be a valid JsonLogic configuration.

Paths are represented in a JSON configuration like this:

```
{
  ...
  "paths": [
    {
      "id": "hardFail",
      "destination": "denied",
      "condition": {
        "===": [ { "var": [ "txn.propertyMap.hardFail" ] }, true ]
      }
    }
  ],
  ...
}
```

Putting it together

Key Concepts:

- There may be one or more paths to a single destination.
- Destinations always require a path, unless they are the default destination.
- When a path's condition returns `true`, the destination is chosen.
- If there are multiple decision rules, use `decisionType` to simplify logic.
- Use `txn.` or `data.` to access current values on which to base decisions.

For example, if you want to always approve an application unless the `txn.hardFail` property is `true`, you can set the default destination to be `approved`, and create a path to `denied`. Consequently, the `approved` destination does not require a path to be chosen.

You can also add a `decisionType` parameter to the call and check it via `params.decisionType` which controls when the rule should be applied. In this case, the data variable `data.application.documentSigningURL` is not set until we are at the "esign" step, so it can be ignored until the appropriate decision step is reached.

Using the examples in the previous sections, the configuration file might look like this:

```
{
  "destinations": [
    {
      "id": "approved",
      "isDefault": true
    },
    {
      "id": "denied"
    },
    {
      "id": "waiting"
    }
  ],
  "paths": [
    {
      "id": "hardFail",
      "destination": "denied",
      "condition": {
        "===": [ { "var": [ "txn.propertyMap.hardFail" ] }, true ]
      }
    },
    {
      "id": "Signature Document Generated",
      "destination": "approved",
      "condition": {
        "and" : [
          {
            "!=": [ { "var": [ "data.application.documentSigningURL" ] }, "unavailable" ]
          },
          {
            "==": [ {"var": [ "params.decisionType" ] }, "esign" ]
          }
        ]
      }
    },
    {
      "id": "Signature Document Not Generated",
      "destination": "waiting",
      "condition": {
        "and" : [
          {
```

```
            "==": [ { "var": [ "data.application.documentSigningURL" ] }, "unavailable" ]
          },
          {
            "==": [ { "var": [ "params.decisionType" ] }, "esign" ]
          }
        ]
      }
    }
  ],
  "version": "24.2.0"
}
```

Next, learn about Decision Framework data.

Decision Framework: Data

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10

The Decision Framework relies on an external source to provide it with data for decision making. The external source can be any other Fluent Function that returns a `FormFuncResult`. The data `Map` in the `FormFuncResult` object is passed directly into the Decision Engine at run-time to generate results.

Configuring the Data function

There are two approaches for configuring the Data function:

- Global configuration
- (Recommended) Form-specific configuration

Global / Default configuration

By default the Decision Framework will take the name and version of the Data Fluent Function from its Service parameters. This can be useful if the mechanism for data retrieval is consistent across multiple forms.

Form-specific configuration

If you require a specific Data function for your application or form, you can configure the following two form properties and the Decision Framework will read from these properties at run-time to determine the correct service. These properties take precedence over the Narrator properties listed below.

Property	Description
`decisionDataModelSvcName`	`String` A data model service name. For example, the DataModel service.
`decisionDataModelSvcVersion`	`String` A `decisionDataModelSvcName` service version. For example, `"1.0.1"`.

If you are using the same data model service for both Narrator and the Decision Framework then you can set the following two Form Properties. This is how Springboard configures the data model service.

Property	Description
`dataModelSvcName`	`String` A data model service name. For example, the DataModel service.
`dataModelSvcVersion`	`String` A `decisionDataModelSvcName` service version. For example, `"1.0.1"`.

Accessing data

You can access data from your configured Data Model, TXN Property Map, or via Parameters passed into the service.

To access the data in your rules logic, use the var operator. You'll find the data `Map` returned by your configured data model inside the `dataModel` object.

The syntax for referring to data varies depending on the source of the data:

- **Data model:** `data.<modelPath>`
- **TXN properties:** `txn.propertyMap.<property>`
- **Parameters:** `params.<paramName>`

For more examples, see Decision Framework: Rules logic > Retrieving data and Decision Framework example configurations.

Next, learn about the Decision Framework result.

Decision Framework: Result

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10

The result of running the Decision Framework is an object representing all the possible destinations, with one or more of the destinations being active.

Below is an example result where the default `"approved"` destination was chosen by default because the `"denied"` path's configuration did not return `true`.

```
{
  "destinations" : {
    "approved": {
      "isActive": true,
      "reasons": [ "Default destination" ]
    },
    "denied": {
      "isActive": false,
      "reasons": []
    }
  },
  "calculatedDate": "2018-01-13T18:25:43.511Z"
}
```

Here is another example result, where `"denied"` is active.

```
{
  "destinations" : {
    "approved": {
      "isActive": false,
      "reasons": []
    },
    "denied": {
      "isActive": true,
      "reasons": [ "hardFail" ]
    }
  },
  "calculatedDate": "2018-01-13T18:25:43.511Z"
}
```

Next, learn about Decision Framework rules logic.

Decision Framework: Rules logic

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10

The core logic engine behind the Decision Framework is a Groovy implementation of JsonLogic. All operations supported by JsonLogic are supported by default in the Decision Engine.

There are many examples of using JsonLogic operations in the JsonLogic documentation. The examples below focus on configurations that are likely to be used in the Decision Framework.

Retrieving data

JsonLogic uses the var operator to retrieve data at runtime, using dot notation to access properties. The specific data available to your rules is governed by the data passed into the Decision Framework. To learn more, see Decision Framework: Data.

- Example: Hard Fail Txn Property

```
{
  "var": [ "txn.propertyMap.hardFail" ]
}
```

- Example: Qualifile Account Action Text

```
{
  "var": [ "txn.propertyMap.FisQualiFile\\.PrimaryApplicant\\.accountActionTxt1" ]
}
```

- Example: Qualifile Account Acceptance Text

```
{
  "var": [ "txn.propertyMap.FisQualiFile\\.PrimaryApplicant\\.accountAcceptanceTxt"
]
}
```

- Example: Application data value for LMS Document Signing URL

```
{
  "var": [ "data.application.documentSigningURL" ]
}
```

- Example: Service parameter defined in Narrative to control rule usage

```
{
  "var": [ "params.decisionType" ]
}
```

- Example: Accessing data model and parameters

```
{
  "id": "Signature Document Generated",
  "destination": "approved",
  "condition": {
    "and" : [
      {
        "!=": [ { "var": [ "data.application.documentSigningURL" ] }, "unavailable"
]
      },
      {
        "==": [ { "var": [ "params.decisionType" ] }, "esign" ]
      }
    ]
  }
}
```

Logic operators

JsonLogic uses logic operators to return a result. The logic needs to result in `true` or `false`.

Following are some example statements in plain English, and JsonLogic that implements them.

- **Example:** Check if a value has been set by LMS signature processing

```
{
  "and" : [
    {
      "!=": [
        { "var": ["data.application.documentSigningURL"] }, // Contains LMS URL,
returned when ready
        "unavailable"
      ]
    },
    {
      "==": [
        { "var": ["params.decisionType"] }, // Decision service call includes
"decisionType" parameter
        "esign"
      ]
    }
  ]
}
```

- **Example:** If the user's credit score is less than 800

```
{
  "if" : [
    {
      "<": [
        { "var": "txn.propertyMap.creditScore" }, // user's credit score
        800
      ]
    }
  ]
}
```

- **Example:** If the user's credit score is greater than 700, and less than 800

```
{
  "and": [
    {
      ">": [ { "var": "txn.propertyMap.creditScore" }, 700 ]
    }
    {
      "<": [ { "var": "txn.propertyMap.creditScore" }, 800 ]
    }
  ]
}
```

  This example builds on the previous rule. The `"and"` operator is used to combine the two rules. If both of the logic operations inside `"and"` are true, then `"and"` is true. If either of the logic operations inside `"and"` are false, then `"and"` is false.

Next, let's look at an example that approves applications by default.

Example: Approved by default

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10

This example demonstrates a Decision Framework scenario whereby applications are approved by default.

Configuration

```
{
  "destinations": [
    {
      "id": "approved",
      "isDefault": true
    },
    {
      "id": "denied"
    }
  ],
  "paths": [
    {
      "id": "hardFail",
      "destination": "denied",
      "condition": {
        "===": [ { "var": [ "txn.propertyMap.hardFail" ] }, true ]
      }
    }
  ],
  "version": "0.1"
}
```

Data

```
{
  "txn": {
    "propertyMap": {
      "hardFail": false
    }
  }
}
```

Result


```
{
  "destinations" : {
    "approved": {
      "isActive": true,
      "reasons": [ "Default destination"  ]
    },
    "denied": {
      "isActive": false,
      "reasons": []
    }
  },
  "calculatedDate": "2018-01-13T18:25:43.511Z"
}
```


Next, let's look at an example that uses the "hardfail" property to deny applications.

Example: Denied by "hardFail"

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10

This example demonstrates a Decision Framework scenario whereby applications are denied by the `"hardfail"` property.

Configuration

```
{
  "destinations": [
    {
      "id": "approved",
      "isDefault": true
    },
    {
      "id": "denied"
    }
  ],
  "paths": [
    {
      "id": "hardFail",
      "destination": "denied",
      "condition": {
        "===": [ { "var": [ "txn.propertyMap.hardFail" ] }, true ]
      }
    }
  ],
  "version": "0.1"
}
```

Data

```
{
  "txn": {
    "propertyMap": {
      "hardFail": true
    }
  }
}
```

Result


```
{
  "destinations" : {
    "approved": {
      "isActive": false,
      "reasons": []
    },
    "denied": {
      "isActive": true,
      "reasons": [ "hardFail" ]
    }
  },
  "calculatedDate": "2018-01-13T18:25:43.511Z"
}
```


Next, let's look at an example that uses IDV verify status to decline applications.

Example: Declined by IDV verify status

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10

This example demonstrates a Decision Framework scenario whereby applications are declined by the IDA or IDV verify status.

Configuration

```
{
  "destinations": [
    {
      "id": "approved",
      "isDefault": true
    },
    {
      "id": "declined"
    }
  ],
  "paths": [
    {
      "id": "declinedByIdaVerifyStatus",
      "destination": "declined",
      "condition": {
        "==": [ { "var": [ "txn.propertyMap.FisIda\\.PrimaryApplicant\\.verifyStatus" ]
}, "FAILED" ]
      }
    },
    {
      "id": "declinedByIdvVerifyStatus",
      "destination": "declined",
      "condition": {
        "==": [ { "var": [ "txn.propertyMap.FisIdv\\.PrimaryApplicant\\.idvVerifyStatus"
] }, "FAILED" ]
      }
    }
  ],
  "version": "1.0.0"
}
```

Data

```
{
  "txn": {
    "propertyMap": {
      "FisIdv\\.PrimaryApplicant\\.idvVerifyStatus": "FAILED",
      "FisIda\\.PrimaryApplicant\\.verifyStatus": "PASSED"
    }
  }
}
```

Result

```
{
  "destinations" : {
    "approved": {
      "isActive": false,
      "reasons": []
    },
    "declined": {
      "isActive": true,
      "reasons": [
        "declinedByIdvVerifyStatus"
      ]
    }
  },
  "calculatedDate": "2018-01-13T18:25:43.511Z"
}
```

Next, let's look at an example that declines applications based on Qualifile account acceptance.

Example: Declined by Qualifile account acceptance

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10

This example demonstrates a Decision Framework scenario whereby applications are declined by Qualifile account acceptance.

Configuration

```
{
  "destinations": [
    {
      "id": "approved",
      "isDefault": true
    },
    {
      "id": "declinedFCRA"
    },
    {
      "id": "declined"
    }
  ],
  "paths": [
    {
      "id": "declinedByQualifileAccountAcceptance",
      "destination": "declinedFCRA",
      "condition": {
        "and": [
          {
            "==": [ { "var": ["txn.-
propertyMap.FisQualiFile\\.PrimaryApplicant\\.accountAcceptanceTxt"] }, "DECLINE" ]
          },
          {
            "and": [
              {
                "!=": [
                  true,
                  {
                    "in": [
                      { "var": ["txn.-
propertyMap.FisQualiFile\\.PrimaryApplicant\\.accountActionTxt1"] },
                      ["D0100001", "D0100002", "D0100002", "D0100004"]
                    ]
                  }
                ]
              }
            ]
          }
        ]
      }
```

```
    },
    {
      "id": "declinedByQualifileCreditQuality",
      "destination": "declined",
      "condition": {
        "and": [
          {
            "==": [ { "var": ["txn.-
propertyMap.FisQualiFile\\.PrimaryApplicant\\.accountAcceptanceTxt"] }, "DECLINE" ]
          },
          {
            "in": [
                { "var": ["txn.-
propertyMap.FisQualiFile\\.PrimaryApplicant\\.accountActionTxt1"] },
                ["D0100001", "D0100002", "D0100002", "D0100004"]
            ]
          }
        ]
      }
    }
  ],
  "version": "1.0.0"
}
```

Data

```
{
  "txn": {
    "propertyMap": {
      "FisQualiFile\\.PrimaryApplicant\\.accountActionTxt1": "UNKNOWN",
      "FisQualiFile\\.PrimaryApplicant\\.accountAcceptanceTxt": "DECLINE"
    }
  }
}
```

Result


```
{
  "destinations" : {
    "approved": {
      "isActive": false,
      "reasons": []
    },
    "declined": {
      "isActive": false,
      "reasons": []
    },
    "declinedFCRA": {
      "isActive": true,
      "reasons": ["declinedByQualifileAccountAcceptance"]
    }
  },
  "calculatedDate": "2018-01-13T18:25:43.511Z"
}
```


Next, let's look at an example that controls when applications go to review.

Example: Control when applications go to review

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10

This example demonstrates a scenario where the Decision Framework decides what causes an application to go for review. Multiple rules can be included.

```
{
  "id": "Primary Applicant Review",
  "destination": "manualReview",
  "condition": {
    "or": [
      {
        "==": [ { "var": ["txn.propertyMap.alloy\\.device\\.outcome"] }, "Manual Review"
]
      },
      {
        "==": [ { "var": ["txn.-
propertyMap.Iovation\\.CheckTxnDetails\\.Result\\.PrimaryApplicant\\.result"] }, "R" ]
      }
    ]
  }
},
{
  "id": "Joint Applicant Review",
  "destination": "manualReview",
  "condition": {
    "==": [ { "var": ["txn.propertyMap.alloy\\.joint\\.person\\.outcome"] }, "Manual
Review" ]
  }
},
{
  "id": "Manual Review Loan",
  "destination": "manualReview",
  "condition": {
    "or" : [
      {
        "==": [ { "var": ["txn.propertyMap.Application_Decision"] }, "ERROR" ]
      },
      {
        "==": [ { "var": ["txn.propertyMap.Application_Decision"] }, "NONE" ]
      },
      {
        "==": [ { "var": ["txn.propertyMap.Application_Decision"] }, "REVIEW" ]
      },
      {
        "==": [ { "var": ["txn.propertyMap.Application_Decision"] }, "STIPULATION_ERROR"
]
```

```
      }
    ]
  }
}
```

The Narrative used in conjunction with this set of rules could have the following terminal page definition.

```
{
  "name": "Review Page",
  "condition": "${ jsonSlurper.parseText( txn.propertyMap.get('decision') )['des-
tinations']['manualReview']['isActive']}",
  "allowSubmit": true,
  "preActions": [],
  "postActions": [],
  "nextPages": []
}
```

Next, let's look at an example that controls when applications are declined.

Example: Control when applications are declined

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10

This example demonstrates a scenario where the Decision Framework decides what causes an application to be declined. Multiple rules can be included.

```
{
  "id": "Device Check Suspect",
  "destination": "declined",
  "condition": {
    "and": [
      {
        "!=": [ { "var": [ "txn.propertyMap.alloy\\.device\\.outcome" ] }, "Approved" ]
      },
      {
        "!=": [ { "var": [ "txn.-
propertyMap.Iovation\\.CheckTxnDetails\\.Result\\.PrimaryApplicant\\.result" ] }, "A" ]
      }
    ]
  }
},
{
  "id": "Primary Applicant Declined",
  "destination": "declined",
  "condition": {
    "or": [
      {
        "==": [ { "var": [ "txn.propertyMap.alloy\\.device\\.outcome" ] }, "Denied" ]
      },
      {
        "==": [ { "var": [ "txn.-
propertyMap.Iovation\\.CheckTxnDetails\\.Result\\.PrimaryApplicant\\.result" ] }, "D" ]
      }
    ]
  }
},
{
  "id": "Joint Applicant Declined",
  "destination": "declined",
  "condition": {
    "==": [ { "var": [ "txn.propertyMap.alloy\\.joint\\.person\\.outcome" ] }, "Denied" ]
  }
},
{
  "id": "Decline Loan",
  "destination": "declined",
  "condition": {
```

```
    "and" : [
      {
        "==":[ { "var": ["txn.propertyMap.Application_Decision"] }, "AUTO_REJECTED" ]
      },
      {
        "==": [ { "var": ["params.decisionType"] }, "lms" ]
      }
    ]
  }
}
```

The Narrative used in conjunction with this set of rules could have the following terminal page definition.

```
{
  "name": "Declined Page",
  "condition": "${ jsonSlurper.parseText( txn.propertyMap.get('decision') )['des-
tinations']['declined']['isActive']}",
  "allowSubmit": true,
  "preActions": [],
  "postActions": [],
  "nextPages": []
}
```

Next, let's look at an example showing how to access data in rules.

Example: Accessing data in rules

SpringboardThis topic is related to Springboard. | Form Builder | 23.10This feature was updated in 23.10

This Decision Framework example demonstrates how to access txn, data model, and parameters for use in rules.

```
{
  "destinations": [
    {
      "id": "approved",
      "isDefault": true
    },
    {
      "id": "waiting"
    },
    {
      "id": "declined"
    }
  ],
  "paths": [
    {
      "id": "Joint Applicant Declined",
      "destination": "declined",
      "condition": {
        "==": [ { "var": ["txn.propertyMap.alloy\\.joint\\.person\\.outcome"] }, "Denied"
]
      }
    },
    {
      "id": "Signature Document Generated",
      "destination": "approved",
      "condition": {
        "and" : [
          {
            "!=": [ { "var": [ "data.application.documentSigningURL" ] }, "unavailable" ]
          },
          {
            "==": [ { "var": [ "params.decisionType" ] }, "esign" ]
          }
        ]
      }
    },
    {
      "id": "Signature Document Not Generated",
      "destination": "waiting",
      "condition": {
        "and" : [
```

```
          {
            "==": [ { "var": [ "data.application.documentSigningURL" ] }, "unavailable" ]
          },
          {
            "==": [ { "var": [ "params.decisionType" ] }, "esign" ]
          }
        ]
      }
    }
  ]
}
```

The Narrative syntax to pass params to the **Decide** service is as follows:

```
{
  "name": "Decide",
  "version": "${decision-framework-version}",
  "condition": "true",
  "params": {
    "decisionType": "esign"
  },
  "alwaysProcess": true
}
```