A green buoy in the water

Description automatically generatedA black and grey logo

Description automatically generated

Solution Architecture Design (SAD)

**Project Star**

SAD00002

EY Systems

**CONFIDENTIAL**

**Contents**

[1. Solution Objective 4](#_Toc189557418)

[Request Change Tracking 4](#_Toc189557419)

[2. Architecture Overview 4](#_Toc189557420)

[Solution Diagram 4](#_Toc189557421)

[Narrative 5](#_Toc189557422)

[Security 5](#_Toc189557423)

[Observability and Supportability 5](#_Toc189557424)

[3. Principles and Requirements Compliance 6](#_Toc189557425)

[Principles 6](#_Toc189557426)

[Exceptions 7](#_Toc189557427)

[4. Technology 7](#_Toc189557428)

[Appian 7](#_Toc189557429)

[Diagram 7](#_Toc189557431)

[Activity Diagrams 7](#_Toc189557432)

[Sequence Diagrams 7](#_Toc189557433)

[Data / Persistence 7](#_Toc189557434)

[Security 7](#_Toc189557435)

[Integration 7](#_Toc189557436)

[Performance 7](#_Toc189557437)

[Observability and Supportability 7](#_Toc189557438)

[Resilience & Recovery 7](#_Toc189557439)

[Test Cases 7](#_Toc189557440)

[OKTA 7](#_Toc189557441)

[Diagram 7](#_Toc189557442)

[Security 7](#_Toc189557443)

[Integrations 7](#_Toc189557444)

[Performance 7](#_Toc189557445)

[Observability and Supportability 7](#_Toc189557446)

[Test Cases 7](#_Toc189557447)

[Workato 7](#_Toc189557448)

[Diagram 7](#_Toc189557449)

[Activity Diagrams 7](#_Toc189557450)

[Sequence Diagrams 7](#_Toc189557451)

[Data / Persistence 7](#_Toc189557452)

[Security 7](#_Toc189557453)

[Integrations 7](#_Toc189557454)

[Performance 7](#_Toc189557455)

[Observability and Supportability 7](#_Toc189557456)

[Resilience and Recovery 7](#_Toc189557457)

[Test Cases 7](#_Toc189557458)

[5. Security 7](#_Toc189557459)

[4.2 Security policies 7](#_Toc189557460)

[4.3 Authentication methods 7](#_Toc189557461)

[4.4 Encryption standards 7](#_Toc189557462)

[4.5 Access controls 7](#_Toc189557463)

[6. Deployment Approach. 7](#_Toc189557464)

[Appian 7](#_Toc189557465)

[Deployment steps 7](#_Toc189557466)

[Rollback plans 7](#_Toc189557467)

[Operational Procedures. 7](#_Toc189557468)

[7. Threat & Misuse Cases 8](#_Toc189557469)

[8. Risk, Assumptions and Exclusions 8](#_Toc189557470)

[Risks 8](#_Toc189557471)

[Assumptions 8](#_Toc189557472)

[Exclusions 8](#_Toc189557473)

[9. Glossary of Terms 8](#_Toc189557474)

[10. Associated Documents 8](#_Toc189557475)

[11. Review 8](#_Toc189557476)

[Outcome 8](#_Toc189557477)

1. Solution Objective

This document is intended to be a Reference Architecture for the Customer Onboarding and Due-Diligence applications built on the Appian at Waystone. The Reference Architecture document serves as a blueprint providing succinct design prescriptions where this is possible, and design guidelines where the specifics depend on the application requirements. The guidelines and standards set forth in this document are based on Appian’s best practices and support the three phases of a project – Discover, Build and Run.

Request Change Tracking

*[Any change request referenced as part of the delivery]*

|  |  |  |
| --- | --- | --- |
| Change Reference | Date | Information |
| *[Business change reference]* | *[Date of request]* | *[This is how the change has made it into a delivery, CR, Project or Other initiative]* |
|  |  |  |
|  |  |  |
|  |  |  |

1. Architecture Overview

Solution Diagram

A screenshot of a computer

Description automatically generated

Narrative

Due diligence & fund onboarding application provide the Waystone user a platform to complete their activity which they are doing as of now manually. We have integrated the other third-party system with our new application to avoid any human error and make the work faster as multiple process would happen automatically based on pre-configure rules. Users would be able to track the due diligence or fund onboarding from any device (Waystone authorized) and be up to date about the work & task.

Security

In both application due diligence & onboarding, security has been implemented in multiple places. At the time of user login into the apian portal, they would be authorized via OKTA to inherit Waystone security policy. Once user would authenticate via OKTA then user would be added into respective groups defined by OKTA & Appian, based on groups user will be able to see the data/task etc. During onboarding & due diligence process, Appian need to integrate

Observability and Supportability

As Appian is working here as workflow manager, means managing the flow of both applications (due diligence & onboarding). Communicating with Workato to pull/push data into multiple systems, hence workflow activities would be easily monitored from Appian out of box monitoring feature. Appian instance monitoring would be done by Appian support team (Appian developers) with admin access in the systems. They have to keep eyes on monitoring dashboard for any kind of errors etc. Appian logs multiple activities out of box by itself, for any kind of details root cause analysis, those logs can be drill down. Logs access would be only Appian admin will have.

*Health Check Report*

Health Check provides insights into application design patterns and performance risks in your environment. The insights are summarized in a report that includes links to suggestions for mitigating any identified risks.

Health Check is available to system administrators on any environment and is located in the [Admin Console](https://docs.appian.com/suite/help/24.4/Appian_Administration_Console.html).

 Health Check identifies design and configuration problems early on, when they are easiest and cheapest to fix. When testing your application, Health Check can pinpoint potential functional and performance risks. And when your application is in production, Health Check helps to monitor environment performance and infrastructure capacity trends over time.

A close-up of a data review

Description automatically generated

Monitor View

The Monitor view helps Appian admin keep an eye on health and activity indicators for your applications. It has multiple elements i.e. health dashboard, Process Activity etc.

1. Health Dashboard

The Health Dashboard tab provides an at-a-glance overview of the health of your environment or application. It includes both runtime and design-time information and surfaces related metrics and key performance indicators. Below image is for reference only.

A screenshot of a computer

Description automatically generated

1. Process Activity

Provides a summary of the contents visible on the [Process Activity](https://docs.appian.com/suite/help/24.4/monitoring_view.html#process-activity) tab. This card includes a breakdown of all processes by status. It also displays the number of processes with [unresolved errors](https://docs.appian.com/suite/help/24.4/Process_Errors.html#resolved-and-unresolved-errors) and the number of processes that have completed within the last 24 hours. Clicking this card navigates you to the Process Activity tab.

1. Principles and Requirements Compliance

Principles

*Modularity*

We have tried to build both application in modular way, by splitting down functionalities into smaller, self-contained modules which can be independently developed, tested & maintained as well. Since modules are independent, they can often be reused in different parts of the system or even in other systems. This can significantly reduce development time and effort.

Re-usability

While building the application, we also try to recognize the feature which can be used in multiple please to avoid development effort for same kind of functionality. In both applications due diligence & fund onboarding, notification need to trigger. To send the notification/email, we are building a common component so it can be utilized in multiple places now as well as in the future as well, similarly, we are putting all type of integrations into a common component to use further into multiple places etc.

Single View of Customers

Due diligence & fund onboarding will provide single view of customer to the analyst, project manager & other required users according to their defined roles in the system. User would be able to track every activity of a particular due diligence or fund onboarding from a single view, i.e. number of task, their status, due diligence status, associated documentation etc.

Security

Security is being implemented in every place as per Waystone defined architecture. From the user prospect, user would have accessed the only required set of data as per their role defined. In integration as per Waystone, every call would be route through ESB portal Workato.

Exceptions

As of now, we are following all the architecture guidelines from Waystone to implement two solution/applications.

1. Technology

Appian

Appian’s low-code technology enables applications to be visualized and built up to 10x faster, empowering both professional and citizen developers. Applications need only to be built once and can be deployed across any device or web browser.



Diagram

Onboarding

A screenshot of a diagram

Description automatically generated

Due Diligence

A diagram of a project

Description automatically generated

Activity Diagrams

Attaching the Visio diagram for both onboarding & due diligence process.

Initial due diligence



Ongoing Due diligence



Fund onboarding



Sequence Diagrams

Fund Onboarding

 Fund onboarding handle the fund onboarding process with the help predefined fund template in Appian. for every fund, a project manager would be assigned by global/count head. Project manager will complete every task require to fund onboard either by himself/herself or the help of other team members. During this process Appian will communicate with Workato to get/update details from/to salesforce, sending notifications & working on files.

A diagram of a flowchart

Description automatically generated

Due diligence

Due diligence initiated by Analyst in Appian portal, which would send to counter party delegates to provide questionnaire response. Once Analyst receive all required information from CP delegates, then memo gets generated & memo reviewed by reviewer & approvers. There are two types of due diligence would be created on initial due diligence & ongoing due diligence. During this process, Appian will communicate with workato to send notification to required users.

A diagram of a flowchart

Description automatically generated

Data / Persistence

* + - 1. Data Models

**Due Diligence**

DDQM\_Request is parent table for holding ddq related data. This table hold the DDQ metadata like name, description, analyst, created datetime, expected completion date etc. Other DDQ table hold the details like section, questions etc. DDQM\_Template, Section, Question etc are table to hold reference data like template details, section, question which would be referred while create new ddq request.



**Fund Onboarding**

FO\_Project is parent table for holding project related data. This table hold the project metadata like project name, jurisdiction, project manager, due date etc. Other child table hold the task details for each project, task questions, options, documents etc. FO\_Template, FO\_Tollgate etc hold 6 pre-defined template metadata, tollgate metadata etc.



* + - 1. Database Structures

While designing database, we have used best practices. First tables are split down between two categories references tables & transactional tables. Normalization has been implemented while creating the tables to keep data consistency, avoid data redundancy etc. Primary keys are defined in each table and also every primary key is marked as index. Foreign keys have been defined in relative/child tables to make search operations faster. Tables are defined keeping scalability in future

* + - 1. Data Flow Diagrams

Due diligence

A diagram of a person's process

Description automatically generated

Onboarding

* A diagram of a project

  Description automatically generated
  + - 1. Data Management Strategies

Apart from reference data, transactional data are being captured/input by multiple users in the response of their task/activities. As users are providing the input in the user interface, so we have provided different types of data fields like integer, text etc. along with validation like maximum characters allowed, input is marked necessary in the case of required, null check etc to keep data accurate according to requirements. While integrating, only required fields are transferred to third parties. To define the data consistency, we have used references etc. During the processing, only required fields/data are being saved into required tables and while deleting based on data field, either soft delete or hard delete is being implemented.

* + - 1. Data Categorisation

Data has been categorised into two categories, reference data & transactional data. Reference data holds the information required for process initiation like in due diligence questions, questionnaire, onboarding templates, their tasks, team etc while transactional data are being captured by users in the response of their tasks/activities like delegates responses, task completion data, completed by, documents etc. Apart from this other data also getting saved like audit

* + - 1. Data Retention

1. There would be scheduled process for both applications DDQ & Fund onboarding, which will run on weekly basis on weekends, to delete the data from database & documents from SharePoint.
2. The process will fetch the completed DDQ & Fund Onboarding request for a fixed time frame ( i.e. 5 years) and delete details from child table except parent table. Child table include details like questions, response, audit etc.

Note : About time frame, still discussions are going on with business, if business will agree then only this process would be build and timeframe would be configured else nothing will happen.

* + - 1. Backup and Recovery

The standard backup and retention practices followed by Appian have been designed to meet the following objectives:

* Minimize interruptions to the normal operations of Appian Cloud
* Limit the extent of disruption and damage for Appian customers
* Minimize the overall impact of an unexpected service interruption
* Establish alternative means of operation
* Serve as training material for personnel with emergency responsibilities
* Provide personnel with a reference to aid in restoration of service

More details can be found on below document



Security

*User Authentication*

User would be authenticated via the OKTA while logging into Appian portal. There would be two types of users internal Waystone user & external Waystone user and both types of users would be authenticated from OKTA only. In Appian, similar groups would be created and user's roles/groups would be mapped from OKTA only.

User would be mapped with different Appian groups which are follow:

Fund onboarding:

|  |  |  |
| --- | --- | --- |
| Group Name | Internal/External | Number of Group |
| FO Global Head | Internal | 1 |
| FO Country Head | Internal | 1 |
| FO Project Manager | Internal | 1 |
| FO Support | Internal | 1 |
| FO Delegate | External | 1 |

Due Diligence:

|  |  |  |  |
| --- | --- | --- | --- |
| Group Name |  | Internal/External | Number of Group |
| DDQ Global Head | Global | Internal | 1 |
| DDQ Analyst | each for individual jurisdiction Lux, UK, IRE | Internal | 3 |
| DDQ Analyst Manager | each for individual jurisdiction Lux, UK, IRE | Internal | 3 |
| DDQ Reviewer | each for individual jurisdiction Lux, UK, IRE | Internal | 3 |
| DDQ Approver | each for individual jurisdiction Lux, UK, IRE | Internal | 3 |
| DDQ Country Head | each for individual jurisdiction Lux, UK, IRE | Internal | 3 |
| DDQ Admin | each for individual jurisdiction Lux, UK, IRE | Internal | 3 |
| DDQ Business Manager | each for individual jurisdiction Lux, UK, IRE | Internal | 3 |
| DDQ Delegate |  | External | 1 |

*OKTA Security*

While doing SAML configuration (via OKTA), need to service provider signing certificate in the form of pem file. The below link provides steps how to generate this certificate. While creating this certificate provide 10 years as expiration date (3650 days).



*Encryption*

By default, Appian use AES 256 algorithm to encrypt the data.

*Certificates*

Appian is SaaS based application all kind of certificates related to infrastructure are maintained by

Appian itself like Self-Service Certificate etc. Below document explain how manage Self-Service certificate.



Integration

To align with Waystone defined architecture, Appian (both application due diligence & client onboarding) will have communicate/integrate with all third-party API via Workato except OKTA (single sign on). While integrating with Workato, OAuth 2.0 Client Grant Credential would be used as authentication type. Below image provides the integration architecture as a solution for both the applications.

A white rectangular object with black text

Description automatically generated

* + - 1. End Point Name

**Authentication**: OAuth 2.0 client code grant

Below is the API (tentative)

|  |  |  |
| --- | --- | --- |
| Endpoint | Method | Description |
| /project-details | GET | Retrieves detailed information about projects stored in Salesforce. |
| /update-engagement-status | PUT | Allows updating the engagement status of projects or accounts in Salesforce to reflect the latest developments. |
| /upload-documents | POST | Enables uploading documents directly to Salesforce, associating them with specific project. |
| /create-folder | POST | Creates a new folder in a specified SharePoint library or directory to organize documents. |
| /upload-document | POST | Uploads document to a designated folder in SharePoint for easy access and storage. |
| /delete-document | DELETE | Removes a specific document from a designated SharePoint folder. |
| /grant-access | POST | Assigns access permissions to a specific user for a selected file in SharePoint. |
| /revoke-access | POST | Removes previously granted access permissions for a specific user on a selected file. |
| /retrieve-document | GET | Retrieve document from a specified SharePoint folder. |
| /send-email | POST | Allows sending emails through the Exchange Server by specifying recipients, subject, and email body. |
| api/v1/apps/{app-id}/users | GET | Retrieve users list from given application id |
| api/v1/users/{user-id} | GET | Retrieve users details i.e. firstname, lastname,email etc. |
| api/v1/users/{user-id}/groups | GET | Retrieve user associated groups |

Performance

Appian being a SaaS based application, there is no explicit performance testing required except application testing like QA & UAT,SIT.

Observability and Supportability

Please refer the Section 2.4 – Observability and Supportability

Resilience & Recovery

With High Availability, Appian Cloud is provided via three Availability Zones without a single point of failure. To achieve this, Appian uses servers and storage that are kept up to date with customer data and applications. For this High Availability offering, Appian offers RPO of 1 minute, as well as an RTO of 21 minutes for Advanced customers, or 4 minutes for Enterprise customers.

Test Cases

All testing is happening here manual only.

1. Fund Onboarding:

While doing testing, below functionalities are tested.

* Manual project creation by global head & country head only.
* Project manager assignment by global/country head.
* template configurations
* Task assignment/Update Due date/Cancelling/Complete by project manager.
* Task access/completion by internal project team members.
* External user portal creation & task access.
* Uploading documents & collaboration by multiple users.
* Notifying users in multiple stage via email.
* Multiple report generation.

1. Due diligence:

While doing testing, below functionalities are tested.

* Template management (adding questions, sections/sub-section & questionnaire)
* New initial DDQ initiation by analyst.
* Counter party delegate task assignment.
* CP delegate task completion.
* DDQ memo generation.
* DDQ recommendations generation.
* DDQ reviewed by reviewer, approver 1 & 2.
* Generating upcoming ddq based on rules.
* Ongoing DDQ questions prefilled from last questionnaire.
* Multiple report generation.

OKTA

Diagram

As per Waystone’s architecture, both the applications - Due Diligence & Client Onboarding Single Sign On would be configured for every user to login. OKTA would be used to authorize users as part of SSO configuration. There would be two types of users - Waystone internal users & Waystone external users (mainly counter party delegates). To accommodate this OKTA will have two types of configurations based on the user type and the same needs to be configured within Appian as well. The below diagram represents how this configuration/communication will happen.

A screenshot of a computer

Description automatically generated

SAML Setup in Appian

**Single sign-on**

For a typical SP-initiated login, when a user attempts to connect to Appian, Appian redirects the user's browser to the IdP. The IdP makes an authentication decision and returns that decision to the user's browser, which then sends that decision to Appian. Appian acts on that decision, either permitting or denying the user access to the requested resource without the user having to manually sign in. IdP provider here is OKTA. The sequence diagram below offers more specificity to an SP-initiated login process:

A diagram of a service provider

Description automatically generated

**SAML configuration in Appian**

To configure SAML for single Sign-On, Appian require above information (in the above picture) from OKTA. This configuration/setup will authenticate the user at the time of login but not add the users in any Appian roles/groups defined in OKTA.

A screenshot of a computer

Description automatically generated

To configure SAML for single Sign-On, Appian require above information (in the above picture) from OKTA.

This configuration/setup will authenticate the user at the time of login but not add the users in any Appian roles/groups defined in OKTA. The below details need to be configured in Appian console for SAML setup.

Service provider signing certificate: need to be provided by client IT team for each Appian instance.

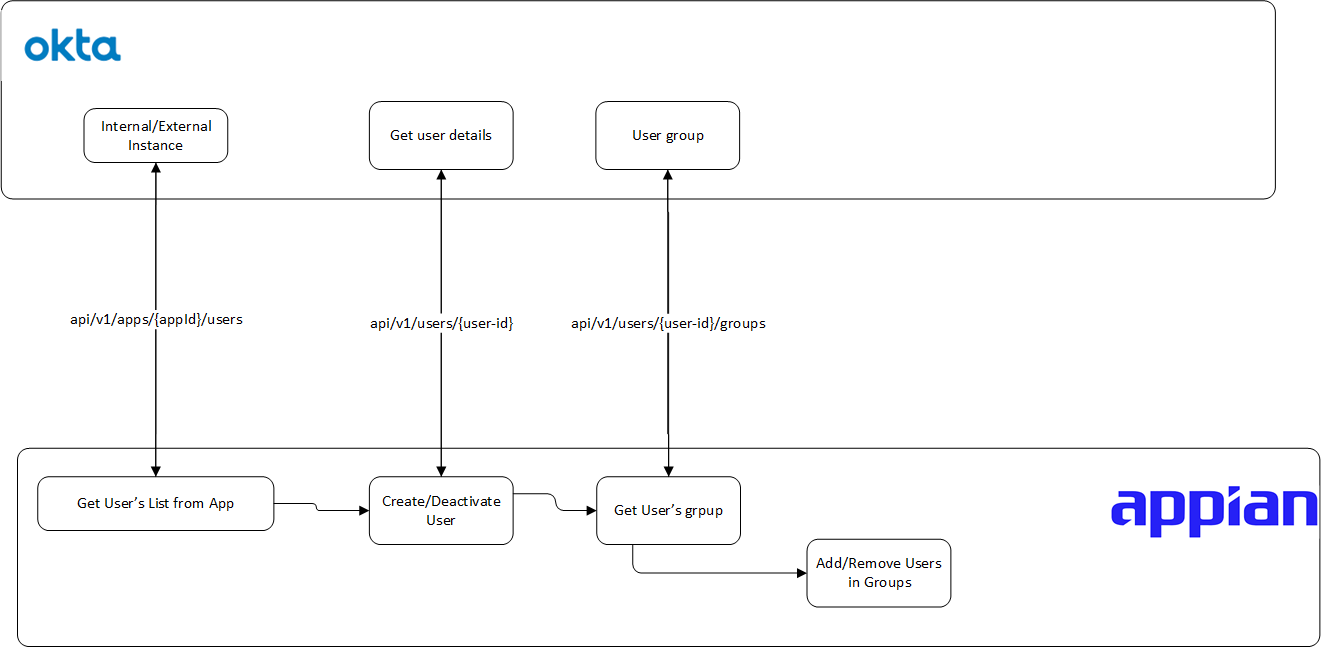
Service Provider Entity Id: The Appian host instance need to be entered here.

Identity Provider Metadata: need to be provided by OKTA team.

Authentication method: we will set as None here.

**User Creation/Deactivation**

Users would be created within Appian with the help of OKTA API. A batch process would run every day at schedule time to consume OKTA API and with the help of API response users would be created within Appian in every day. New users would be created, and users would be marked as deactivated in Appian in the case of any user required to be removed from Appian user’s list. Users would be added into respective groups as well in Appian. Here API-Key security would be used for authentication.



**User’s role/group synchronization**

Appian provide out of box feature to synchronize the users group from SAML to respective Appian groups. Groups would be synchronized with OKTA groups at time of user login.

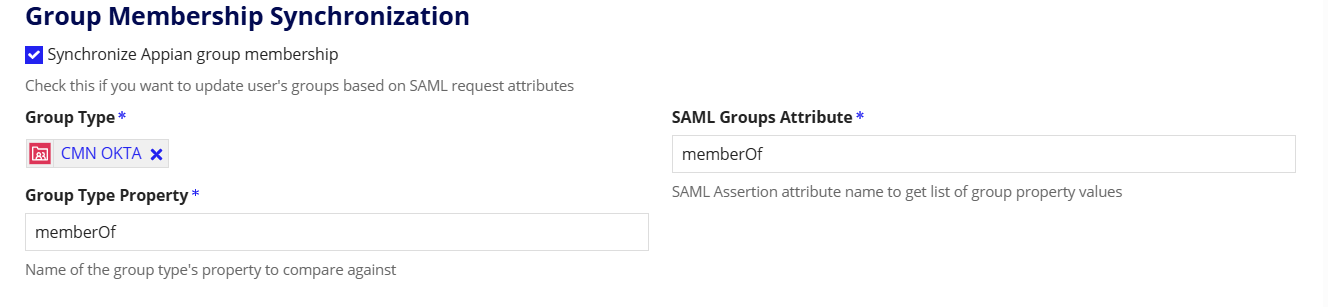
We will follow two ways to user’s role/group.

1. OKTA API

During scheduled process once user would create then with help of OKTA API, user’s associated groups would be fetched from OKTA and would added/removed from respective Appian groups.

1. Group Membership synchronization:

This is out of box feature provided by Appian. This only add/remove users at time of every login.



Above place need to configure to auto synchronization.

**Group Type**: Need to create a group type in Appian application and same need to configure here. This group type needs to select in all required groups created for the Appian application, where user need to be added as per OKTA group configuration.

**Group Type Property**: This need to be configured in Group Type in Appian and add group attribute on which basis the users are going to added into respective groups.

**SAML Groups Attribute**: This will come from SAML assertion attribute. Name to get list of group property values.

***In the case of any new group would be created in OKTA then same group need to be created with Appian application by Appian support team/developer team and set the filter then only newly created group will start adding users.***

**Deployment in higher environment**

Need to follow the above-mentioned steps in all environments manually, where we need to enable SAML.

The below document provides details setup SAML in Appian:

Security

MFA settings required for accessing this application, such as mandatory MFA during login or for certain user roles. Specify any adaptive MFA settings (e.g., based on user location, device, or risk). Define the access policies.

Integrations

*Okta Integration with Appian*

Log into Okta Admin Console: External & Internal instance

Access Okta Admin Console using your administrative credentials.

Add Appian as a New Application:

In the Admin Console, navigate to **Applications > Applications.**

Click **Add Application**.

Search for Appian in the app catalogue or select **Create New App** if not available.

Choose SAML 2.0 as the Sign-On Method:

Select SAML 2.0 from the available options for sign-on method.

Configure the SAML Settings:

**Single sign-on URL:** Enter the URL provided by Appian for SSO (e.g., <https://yourappianurl.com/sso/saml>).

**Audience URI (SP Entity ID**): Enter the Entity ID provided by Appian.

**Application username**: Map this to the correct user attribute (e.g., email or username).

**Authentication Policy:** Make two factor authentication.

**Attribute Mapping:**

Map the Okta attributes (e.g., email, first name, last name) to corresponding SAML attributes expected by Appian.

**Group Attributes Mapping:** Map Okta group for DDQ & FO.

**API Token**- Create an API token using super admin access and share with appian team to develop create user in Appian tenant. Okta team need to share three API to get the scheduler work.

**Download the Okta Metadata:**

After configuration, download the metadata file from Okta. This will include the IdP certificate, URLs, and other configuration details needed for Appian and share it.

**Save and Assign User/Groups:**

Assign the appropriate users and groups to this Appian application in Okta.

*Okta Integration with Sharepoint*

Log into Okta Admin Console: External & Internal instance

Internal users - Belong to a specific Active Directory group and have a Microsoft 365 license

External users - manually setup in Okta, no Microsoft O365 license

Here's how we need it to flow:

User goes to Internet facing URL from Appian

User hits Okta authentication page

Internal user authenticates via SSO with our maintained session Microsoft 365 tenant

External user enters access via our manually setup Okta user

User is passed on to our site based on group they are assigned too.

Access Okta Admin Console using your administrative credentials.

Add O365 as a New Application:

In the Admin Console, navigate to **Applications > Applications.**

Click **Browse App Catalog**.

Search for Office 365 in the app catalog.

Configure SSO

Fill the required details.

Configure SSO with WS-Fed – automatic method:

**Microsoft Tenant Name:** Enter the URL provided by Sharepoint for SSO (e.g- example.microsoft.com).

**Your office 365 company domain:** Enter the domain provided by Sharepoint.

**Display following links:** You have to checkbox on sharepoint online

**Application username**: Map this to the correct user attribute according to domain name (e.g., email ).You can cutomise username for external user where String.substringBefore(user.email, "@") + "[@yourfederated.domain](mailto:@yourfederated.domain)"

Go to **Sign On**>Setting >**Edit**

Ws-Fed>Automatic

Enter you office 365 Admin Username & Password

Click **Fetch and Select**. This displays a list of all Office 365 domains available for federation.

Select domain you want to federate

**Provision user to Office 365**

You can create, update, deprovision, and sync users in Office 365 from your Okta org using SCIM.

Enable API integration

Select **Office 365 Provisioning Type**.

Enable or disable other provisioning settings.

Application provisioning is required in Okta to push users and attributes to service providers such as O365.

**Download the Okta Metadata:**

After configuration, download the metadata file from Okta. This will include the IdP certificate, URLs, and other configuration details needed for Appian and share it.

**Set Up Access & Permissions for Internal Users**

Internal Users are managed from AD so we must log a ticket where appropriate AD group needs to be created with authorised users. Once groups created we will assign to Appian in okta.

**Set Up Access & Permissions for External Users**

Ensure the external users have appropriate permissions on SharePoint resources. This could be done by assigning them to SharePoint groups or providing access to specific sites based on appian application. Once you assign O365 to the user you will see a list of O365 Apps available to the user. From there you will be able to check off the SharePoint Online box and save.

This will assign the user the O365 chiclet in Okta and a license to access SharePoint Online in O365.

**Test provisioning**

Ensure that external users are added to the appropriate groups DDQ & FO and have the necessary permissions.

**SharePoint 365 Configuration:**

Configure SharePoint 365 for SSO:

In the SharePoint 365 admin center, navigate to the settings for authentication and configure SSO using the metadata provided by OKTA.

Ensure that SharePoint 365 trusts OKTA as an identity provider.

**Assign Permissions and Access:**

Define and assign the appropriate permissions and access levels for internal and external users in SharePoint365.

Ensure that external users have access only to the resources they need.

Performance

Okta being a SaaS based application, there is no explicit performance testing.

Observability and Supportability

Log Management, Real time Monitoring & Threshold Alerts.

Support Contacts: List support contacts for Okta, the application, and any third-party services involved in the integration level.

Okta Support Portal: Ensure that administrators have access to Okta’s support resources, including guides, troubleshooting documentation, and contact details for support teams.

In-App Troubleshooting: Provide access to helpful messages or troubleshooting wizards in case of failed authentication, issues with MFA, or application-specific issues.

Okta Knowledge Base: Make sure the team can easily access Okta’s Knowledge Base to find articles related to common integration issues, error codes, and best practices.

Test Cases

Test Case 1: Basic SSO Authentication

Objective: Verify that users can log in to the integrated application using Single Sign-On (SSO) via Okta.

Steps:

Open the integrated application’s login page.

Click on the SSO login button.

The user should be redirected to Okta’s login page.

Enter valid credentials (username and password).

The user should be redirected back to the application successfully and granted access.

Expected Result: User is successfully authenticated and redirected to the application.

**Test Case 2:** Invalid Credentials

Objective: Ensure the application denies access with invalid credentials.

Steps:

Open the integrated application’s login page.

Attempt to log in with invalid Okta credentials (incorrect username or password).

Verify that the user is presented with an error message (e.g., “Invalid credentials”).

Expected Result: Access is denied, and an error message is shown.

**Test Case 3:** MFA (Multi-Factor Authentication) Prompt

Objective: Verify that Okta MFA is triggered as expected based on policy.

Steps:

Attempt to log in to the integrated application using valid credentials.

If MFA is enabled for the user or group, ensure the MFA prompt (e.g., Okta Verify, SMS, or email) appears.

Complete the MFA step (e.g., approve a push notification or enter an OTP).

Verify the user is redirected back to the application successfully after MFA is completed.

Expected Result: MFA is successfully triggered and completed, allowing access to the application.

**Test Case 4:** Failed MFA Authentication

Objective: Ensure that access is denied if the user fails MFA.

Steps:

Attempt to log in using valid credentials.

On the MFA prompt, enter an incorrect or expired verification code.

Verify that the application denies access and an appropriate error message is displayed.

Expected Result: Access is denied, and the error message indicates the MFA failure (e.g., “Incorrect code”).

**Test Case 5:** SSO Timeout

Objective: Verify the SSO session expiration functionality.

Steps:

Log in to the integrated application via Okta.

Leave the application idle for a period (e.g., 30 minutes or configured session timeout duration).

After the session expires, try accessing a protected page.

Ensure the user is redirected to the Okta login page.

Expected Result: Session expires and user is asked to log in again.

**Test Case 6**: Group Assignment Sync

Objective: Verify that user group assignments in Okta are correctly synced to the integrated application.

Steps:

Create a user in Okta and assign them to specific groups.

* Verify that the assigned groups are mapped correctly to the user’s roles or permissions in the integrated application.

Log in to the application and ensure the user has the correct access based on their group membership.

Expected Result: User’s group memberships and associated permissions are correctly applied in the application.

**Test Case 7:** SAML Assertion Validation

Objective: Verify that Okta SAML assertions are correctly validated by the integrated application.

Steps:

Log in to the integrated application via SSO.

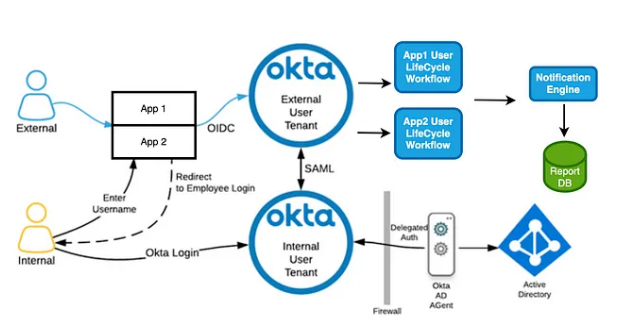
Capture and inspect the SAML assertion sent from Okta.

Validate that the assertion contains the correct user attributes (e.g., NameID, roles, email).

Ensure the assertion is signed and encrypted (if configured).

Expected Result: The SAML assertion is correctly formatted and validated by the application.

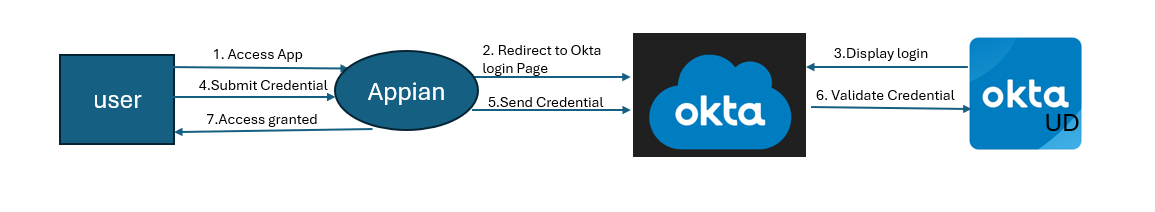
**Data Models**



**Database Structure**

This is a SAAS based app there is no database structure.

**Data Flow Diagrams**



**Data Management Strategies**

**Data Categorisation**

Data categorization in Okta during application onboarding refers to classifying and managing data based on its sensitivity, criticality, and how it interacts with Okta’s identity management systems. Define data sensitivity level, apply RBAC, enforce compliance.

**Data Retention**

The System Log data is maintained for 90 days and Service backup data is automatically purged six months after it is first generated.

**Backup & Recovery**

In general, there are no Okta backup or rollback utilities available. This includes deleted Okta Integration Network (OIN) and custom application data. Our recommendation is to make note of the configuration prior to making any changes. If a change needs to be reverted, it can be manually reverted to the previous settings. Okta has several APIs available to collect configuration information to store in case there is a need to manually revert any changes. In addition to the API, there is also the Rockstar browser plugin, which can be used to pull information regarding users and groups, among other things.

Workato

Workato is a cloud-based integration and automation platform designed to help organizations optimize workflows by seamlessly connecting various applications, services, and systems. Appian is expected to integrate with multiple external technologies, including Salesforce, DocuSign, and SharePoint. Workato will provide a standardized API interface to Appian, enabling access to all required services. This delivery includes several API endpoints as part of the solution.

Diagram

workspace "Software System - Waystone" {
    model {
        
        Appian = softwareSystem "Appian"
        Workato = softwareSystem "Workato"{
            
            Docusign_api = container "APIs collection"{
                description "Set of APIs to communicate with Docusign"
                
                CreateDocument = component "Create New Document"
            }
            
        }
        DocuSign = softwareSystem "Docusign"
        SharePoint = softwareSystem "Sharepoint"
        SalesForce = softwareSystem "Salesforce"
        ExchangeServer = softwareSystem "Exchange"


        Appian -> Workato "API End Point"
        Workato -> DocuSign "Workato Connector" 
        Workato -> SharePoint  "Workato Connector / REST API"
        Workato -> SalesForce "Workato Connector"
        Workato -> ExchangeServer "Workato Connector"
        
        }

    views {
        systemContext Workato "Workato" {
            include *
            autolayout
        }
        
        container Workato "Project-Star" {
            include *
            autolayout lr
        }
        component Docusign_api  "Fund_Static_PowerAutomate"{
            include Appian->
            include ->Workato->
            include ->DocuSign->
            autolayout
        }

    theme default
    styles {

            element "Software System" {
                background #1168bd
                color #ffffff
            }
            element "Existing System" {
                background #999999
                color #ffffff
            }
            element "Container" {
                background #438dd5
                color #ffffff
            }
            element "Web Browser" {
                shape WebBrowser
            }
            element "Mobile App" {
                shape MobileDeviceLandscape
            }
            element "Database" {
                shape Cylinder
            }
            element "Component" {
                background #85bbf0
                color #000000
            }
            element "Failover" {
                opacity 25
            }
            element "Self-Hosted" {
                background #999999
                color #ffffff
            }

            element "sFTP" {
                background #999999
                shape Folder
            }
            element "External System" {
                background #999999
                color #ffffff
            }

            element "New" {
                background #1FAB1B
                color #ffffff
            }

            element "Code Change" {
                background #E4A108
                color #ffffff
            }

            relationship "Messenger" {
                color #000FFF
                 properties {
                    "Channel" "Messenger"
                }
            }

            relationship "MessageClient" {
                color #FF0000
                 properties {
                    "Channel" "MessageClient"
                }
            }

            relationship "HTTP" {
                color #F300FF
                style dotted
            }

            relationship "WCF" {
                color #F364FF
                style dotted
            }

            relationship "Command" {               
                style solid
            }

            relationship "Query" {
                style dashed               
            }
        }
    }
}

The Appian workflow seamlessly orchestrates integrations with multiple systems through Workato’s APIs to streamline project operations. It begins by calling Workato’s Salesforce APIs to retrieve essential project details. Next, the workflow utilizes Workato’s SharePoint APIs to create a dedicated working directory, organizing all project-related documents in a centralized location. At any point during the workflow, the Appian process can call Workato’s Exchange Server API to send notifications to relevant users. Once the documents are prepared, the workflow leverages Workato’s DocuSign APIs to send them for signing, specifying recipients and notifications to expedite approvals.

Activity Diagrams

Sequence Diagrams

All APIs are grouped into their respective systems.

~~DocuSign APIs~~

**~~Endpoint:~~** ~~POST /document-signing~~

~~sequenceDiagram
    participant User
    participant Appian
    participant Workato
    participant DocuSign
    
    User->>Appian: Submit document for signing
    Appian->>Workato: Send API request (POST /document-signing)
    Workato->>DocuSign: Forward API request
    DocuSign-->>Workato: Response (200 OK or error)
    Workato-->>Appian: Response with status
    Appian-->>User: Return status
~~

**~~Narrative~~**

~~The user initiates the document signing process via the Appian. Appian sends a request to Workato, which then forwards it to DocuSign. Once DocuSign processes the request, the response flows back through Workato to Appian and finally to the user.~~

**~~Endpoint:~~** ~~GET /signed-document~~

~~sequenceDiagram
    participant User
    participant Appian
    participant Workato
    participant DocuSign
    
    User->>Appian: Request signed document
    Appian->>Workato: Send API request (GET /signed-document)
    Workato->>DocuSign: Forward API request
    DocuSign-->>Workato: Signed document data
    Workato-->>Appian: Return signed document
    Appian-->>User: Deliver signed document
~~

**~~Narrative~~**

~~The user requests a signed document via Appian. Appian passes the request to Workato, which retrieves the document from DocuSign. The response follows the same chain back, ultimately delivering the signed document to the user.~~

**~~Endpoint:~~** ~~GET /document-status~~

~~sequenceDiagram
    participant User as User System
    participant Appian as Appian System
    participant Workato as Workato Integration
    participant DocuSign as DocuSign API
    
    User->>Appian: Check Document Status (EnvelopeId)
    Appian->>Workato: Send API Request
    Workato->>DocuSign: GET /document-status
    DocuSign->>DocuSign: Validate EnvelopeId
    alt Success
        DocuSign-->>Workato: 200 OK (Document status)
        Workato-->>Appian: Success Response
        Appian-->>User: Return Document Status
    else Failure
        DocuSign-->>Workato: 400/401/500 Error (Failure details)
        Workato-->>Appian: Failure Response
        Appian-->>User: Document status retrieval failed
    end~~

**~~Narrative~~**

~~This flow describes how the user checks the status of a document in the signing process. The request flows from Appian to Workato, which interacts with DocuSign. The status response is sent back through Workato and Appian to the user.~~

**~~Endpoint:~~** ~~DELETE /retract-document~~

~~sequenceDiagram
    participant User as User System
    participant Appian as Appian System
    participant Workato as Workato Integration
    participant DocuSign as DocuSign API
    
    User->>Appian: Retract Document (EnvelopeId)
    Appian->>Workato: Send API Request
    Workato->>DocuSign: DELETE /retract-document
    DocuSign->>DocuSign: Validate EnvelopeId
    alt Success
        DocuSign-->>Workato: 200 OK (Document retracted successfully)
        Workato-->>Appian: Success Response
        Appian-->>User: Document retracted successfully
    else Failure
        DocuSign-->>Workato: 400/401/500 Error (Failure details)
        Workato-->>Appian: Failure Response
        Appian-->>User: Document retraction failed
    end~~

**~~Narrative~~**

~~This flow illustrates how a document is retracted from the signing process. The user requests the retraction via Appian, which forwards the request to Workato. Workato then communicates with DocuSign. The response flows back through Workato and Appian to the user.~~

Salesforce APIs

**Endpoint**: GET/project-details

sequenceDiagram
    participant Appian
    participant Workato
    participant Salesforce

    Appian->>Workato: GET /project-details
    Workato->>Salesforce: GET /project-details
    Salesforce-->>Workato: 200 OK\nProject Details
    Workato-->>Appian: 200 OK\nProject Details
    Salesforce-->>Workato: Error Response
    Workato-->>Appian: Error Response

**Narrative**:

Appian sends a request to Workato to retrieve project details from Salesforce. Workato forwards the request to Salesforce. Salesforce processes the request, retrieves the details, and sends a response to Workato, which relays the information back to Appian.

**Endpoint**: PUT/update-engagement-status

sequenceDiagram
    participant Appian
    participant Workato
    participant Salesforce

    Appian->>Workato: PUT /update-engagement-status
    Workato->>Salesforce: PUT /update-engagement-status
    Salesforce-->>Workato: 200 OK (Status Updated)
    Workato-->>Appian: 200 OK (Status Updated)
    Salesforce-->>Workato: Error Response
    Workato-->>Appian: Error Response

**Narrative**:

Appian initiates a request to update the engagement status of a project in Salesforce. This request is sent to Workato, which validates and forwards it to Salesforce. Salesforce updates the engagement status and sends a confirmation response to Workato, which relays it to Appian.

**Endpoint**: POST /upload-document

sequenceDiagram
    participant Appian
    participant Workato
    participant Salesforce
    Appian->>Workato: POST /upload-documents
    Workato->>Salesforce: POST /CreateContentDocumentVersion
    Salesforce-->>Workato: 200 OK (DocumentContentVersion-Created)
    Workato->>Salesforce: POST /ContentDocumentLink
    Salesforce-->>Workato: 200 OK (ContentDocumentLinked)
    Workato-->>Appian: 200 OK (Document Uploaded)
    Salesforce-->>Workato: Error Response
    Workato-->>Appian: Error Response

**Narrative**:

The process begins when Appian sends a document upload request to Workato. Workato uploads the document to Salesforce, creating a content version and linking it to a record, both confirmed with 200 OK responses. Finally, Workato notifies Appian of the successful upload or relays any error response from Salesforce.

SharePoint APIs

**Endpoint**: POST /create-folder

sequenceDiagram
    participant User
    participant Appian
    participant Workato
    participant SharePoint
    
    User->>Appian: Request to create a folder
    Appian->>Workato: Send API request (POST /create-folder)
    Workato->>SharePoint: Forward API request
    SharePoint-->>Workato: Response (Folder created successfully)
    Workato-->>Appian: Return status
    Appian-->>User: Deliver folder creation status


**Narrative**:

The user requests folder creation via Appian. Appian relays the request to Workato, which communicates with SharePoint to execute the operation. The result of the operation is returned to the user through Appian.

**Endpoint**: POST /upload-document

sequenceDiagram
    participant User
    participant Appian
    participant Workato
    participant SharePoint
    
    User->>Appian: Upload a document
    Appian->>Workato: Send API request (POST /upload-document)
    Workato->>SharePoint: Forward API request
    SharePoint-->>Workato: Response (Document uploaded successfully)
    Workato-->>Appian: Return confirmation
    Appian-->>User: Deliver upload status


**Narrative**:

The user uploads a document using Appian. Appian sends the request to Workato, which processes the upload with SharePoint. A confirmation flows back through the chain to notify the user of the upload status.

**Endpoint**: DELETE /delete-document

sequenceDiagram
    participant User
    participant Appian
    participant Workato
    participant SharePoint
    
    User->>Appian: Request to delete a document
    Appian->>Workato: Send API request (DELETE /delete-document)
    Workato->>SharePoint: Forward API request
    SharePoint-->>Workato: Response (Document deleted successfully)
    Workato-->>Appian: Return confirmation
    Appian-->>User: Deliver deletion status


**Narrative**:

The user requests the deletion of a document via Appian. The request is processed by Workato and forwarded to SharePoint. The status of the deletion is sent back through the chain to the user.

**Endpoint**: POST /grant-access

sequenceDiagram
    participant User
    participant Appian
    participant Workato
    participant SharePoint
    
    User->>Appian: Request to grant access
    Appian->>Workato: Send API request (POST /grant-access)
    Workato->>SharePoint: Forward API request
    SharePoint-->>Workato: Response (Access granted successfully)
    Workato-->>Appian: Return confirmation
    Appian-->>User: Deliver access status


**Narrative**:

The user assigns access permissions for a file using Appian. Appian communicates the request to Workato, which interacts with SharePoint, to grant the requested permissions. A confirmation message is returned to the user.

**Endpoint**: POST /revoke-access

sequenceDiagram
    participant User
    participant Appian
    participant Workato
    participant SharePoint
    
    User->>Appian: Request to revoke access
    Appian->>Workato: Send API request (POST /revoke-access)
    Workato->>SharePoint: Forward API request
    SharePoint-->>Workato: Response (Access revoked successfully)
    Workato-->>Appian: Return confirmation
    Appian-->>User: Deliver access revocation status


**Narrative**:

The user revokes access permissions for a file using Appian. Appian sends the request to Workato, which processes it with SharePoint. The result of the operation is communicated back to the user.

**Endpoint**: GET /retrieve-document

sequenceDiagram
    participant User
    participant Appian
    participant Workato
    participant SharePoint
    
    User->>Appian: Request to retrieve a document
    Appian->>Workato: Send API request (GET /retrieve-document)
    Workato->>SharePoint: Forward API request
    SharePoint-->>Workato: Response (Document retrieved successfully)
    Workato-->>Appian: Return the document
    Appian-->>User: Deliver the document


**Narrative**:

The user retrieves a document stored in SharePoint via Appian. Appian sends the request to Workato, which fetches the document from SharePoint. The document is returned to the user via Appian.

Exchange Server APIs

**Endpoint**: POST/send-email

sequenceDiagram
    participant Appian
    participant Workato
    participant ExchangeServer
    
    Appian->>Workato: POST /send-email
    Workato->>ExchangeServer: POST /send-email
    ExchangeServer-->>Workato: 200 OK (Email sent successfully)
    Workato-->>Appian: 200 OK (Email sent successfully)



**Narrative**:

The **Exchange Server API** facilitates seamless email delivery by integrating with Workato to connect Appian and the Exchange Server. Appian initiates the process by sending a POST /send-email request to the Workato API, containing email details such as recipients, subject, body, and optional attachments. Workato processes this request and forwards it to the Exchange Server, which handles the actual email delivery. Once the email is successfully sent, the Exchange Server responds to Workato with a confirmation. Workato then relays this response back to Appian, ensuring the system receives real-time feedback on the status of the email request. This streamlined process ensures efficient and reliable email communication.

Data / Persistence

* + - 1. Data Models

*No Data models are used in Workato recipes*

* + - 1. Database Structures

*Not Applicable*

* + - 1. Data Flow Diagrams

*Not Applicable*

* + - 1. Data Management Strategies

*Not Applicable*

* + - 1. Data Categorisation

*Not Applicable*

* + - 1. Data Retention

*Not Applicable*

Backup and Recovery

*Not Applicable*

Security

Workato leverages OAuth 2.0 to provide a robust and flexible framework for API security. OAuth 2.0 is an industry-standard protocol for authorization, designed to allow secure access to resources without exposing user credentials. This protocol supports various grant types, making it adaptable to different use cases.

**Key Features of OAuth 2.0 in Workato:**

1. **Granular Permissions**: OAuth 2.0 allows fine-grained access control, enabling you to specify the exact permissions required for each API interaction. This minimizes the risk of over-permissioning and enhances security.
2. **Token-Based Authentication**: Instead of using traditional username and password combinations, OAuth 2.0 uses tokens to authenticate API requests. These tokens are time-bound and can be easily revoked if compromised, adding an extra layer of security.
3. **Client Credentials Grant**: For machine-to-machine communication, Workato supports the Client Credentials grant type. This allows applications to authenticate and interact with APIs securely without user intervention.
4. **Refresh Tokens**: OAuth 2.0 supports the use of refresh tokens, which can be used to obtain new access tokens without requiring the user to re-authenticate. This ensures seamless and secure long-term interactions with APIs.

By implementing OAuth 2.0, Workato ensures that API interactions are secure, scalable, and compliant with modern security standards. This approach not only protects sensitive data but also provides seamless and efficient user experience.

Integrations

*Workato will be exposing API endpoints for every action required by appian for example, API to upload documents to share point, delete documents from share point, get data from salesforce, etc..*

*Details of APIs is listed below.*

API Endpoints

The APIs are categorized into four distinct systems: Salesforce, DocuSign, Exchange Server, and SharePoint. A total of fourteen new API recipes will be created in Workato, with details for each outlined below.

~~DocuSign APIs~~

~~This API suite facilitates seamless integration with DocuSign for managing electronic document workflows. The APIs provides the functionality to submit documents for signing, track their status, retrieve signed copies, and retract documents from the signing process.~~

**~~Authentication~~**~~:~~

~~The API uses~~ **~~OAuth 2.0~~** ~~for secure access. You must include a valid access token in the Authorization header.~~

**~~Example Header~~**~~:~~

|  |
| --- |
| ~~Authorization: Bearer <access-token>~~ |

**~~Base URL~~**~~:~~ [~~https://apim.eu.workato.com/exchange~~](https://apim.eu.workato.com/exchange)

**~~Version~~**~~: V1~~

**~~Available Endpoints~~**~~:~~

|  |  |  |
| --- | --- | --- |
| ~~Endpoint~~ | ~~Method~~ | ~~Description~~ |
| ~~/document-signing~~ | ~~POST~~ | ~~Allows submission of a document to DocuSign to initiate the signing process.~~ |
| ~~/signed-document~~ | ~~GET~~ | ~~Enables retrieval of completed and signed documents from DocuSign for storage or further processing.~~ |
| ~~/document-status~~ | ~~GET~~ | ~~Fetches the status of a document in the signing process. Useful for monitoring and reporting on signing progress~~ |
| ~~/retract-document~~ | ~~DELETE~~ | ~~Allows the cancellation or retraction of a document from the signing process before completion~~ |

**~~Endpoint~~**~~: POST/document-signing~~

**~~Request Parameters:~~**

|  |  |  |  |
| --- | --- | --- | --- |
| ~~Parameter~~ | ~~Type~~ | ~~Required~~ | ~~Description~~ |
| ~~SignerEmail~~ | ~~String~~ | ~~Yes~~ | ~~The email address of the recipient who will sign the document.~~ |
| ~~DocumentName~~ | ~~String~~ | ~~Yes~~ | ~~The name of the document being sent for signing.~~ |
| ~~DocumentContent~~ | ~~String~~ | ~~Yes~~ | ~~The content of the document in Base64-encoded format.~~ |
| ~~SignerName~~ | ~~String~~ | ~~Yes~~ | ~~The name of the recipient who will sign the document.~~ |

**~~Example Request~~**

|  |
| --- |
| ~~{~~  ~~"SignerEmail ": "user@example.com",~~  ~~"documentName": "ContractAgreement.pdf",~~  ~~"documentContent": "VGhpcyBpcyBhIHNhbXBsZSBjb250ZW50IGluIEJhc2U2NA==",~~  ~~"SignerName": "Bob",~~  ~~}~~ |

**~~Response~~**

|  |  |
| --- | --- |
| ~~Status Code~~ | ~~Description~~ |
| ~~200 OK~~ | ~~Document sent successfully for signing.~~ |
| ~~400 Bad Request~~ | ~~Invalid input parameters.~~ |
| ~~401 Unauthorized~~ | ~~Invalid or expired access token.~~ |
| ~~500 Internal Server Error~~ | ~~Server-side issue.~~ |

**~~Example Response:~~**

**~~Response Body for Success~~**

|  |
| --- |
| ~~{~~  ~~"envelopeId": "abc123xyz456",~~  ~~"status": "Success",~~  ~~"message": "Document sent for signing."~~  ~~}~~ |

**~~Response Body for Failure~~**

|  |
| --- |
| ~~{~~  ~~"status": "Failure",~~  ~~"error": "Invalid user email or document content."~~  ~~}~~ |

**~~Endpoint~~**~~: GET/signed-document~~

**~~Request Parameters:~~**

|  |  |  |  |
| --- | --- | --- | --- |
| ~~Parameter~~ | ~~Type~~ | ~~Required~~ | ~~Description~~ |
| ~~EnvelopId~~ | ~~String~~ | ~~Yes~~ | ~~The unique ID of the envelope containing the signed document.~~ |

**~~Example Request~~**

|  |
| --- |
| ~~GET /signed-document?envelopeId=abc123xyz456~~ |

**~~Response~~**

|  |  |
| --- | --- |
| ~~Status Code~~ | ~~Description~~ |
| ~~200 OK~~ | ~~Signed document retrieved successfully.~~ |
| ~~400 Bad Request~~ | ~~Invalid input parameters.~~ |
| ~~401 Unauthorized~~ | ~~Invalid or expired access token.~~ |
| ~~500 Internal Server Error~~ | ~~Server-side issue.~~ |

**~~Example Response:~~**

**~~Response Body for Success~~**

|  |
| --- |
| ~~{~~  ~~"documentName": "Signed\_Contract.pdf",~~  ~~"documentContent": "VGhpcyBpcyBhIHNhbXBsZSBzaWduZWQgZG9jdW1lbnQgY29udGVudCBpbiBCYXNlNjQ="~~  ~~}~~ |

**~~Response Body for Failure~~**

|  |
| --- |
| ~~{~~  ~~"status": "Failure",~~  ~~"error": "Envelope ID not found."~~  ~~}~~ |

**~~Endpoint~~**~~: GET/document-status~~

**~~Request Parameters:~~**

|  |  |  |  |
| --- | --- | --- | --- |
| ~~Parameter~~ | ~~Type~~ | ~~Required~~ | ~~Description~~ |
| ~~EnvelopId~~ | ~~String~~ | ~~Yes~~ | ~~The unique ID of the envelope containing the document.~~ |

**~~Example Request~~**

|  |
| --- |
| ~~GET /document-status?envelopeId=abc123xyz456~~ |

**~~Response~~**

|  |  |
| --- | --- |
| ~~Status Code~~ | ~~Description~~ |
| ~~200 OK~~ | ~~Document status retrieved successfully.~~ |
| ~~400 Bad Request~~ | ~~Invalid input parameters.~~ |
| ~~401 Unauthorized~~ | ~~Invalid or expired access token.~~ |
| ~~500 Internal Server Error~~ | ~~Server-side issue.~~ |

**~~Example Response:~~**

**~~Response Body for Success~~**

|  |
| --- |
| ~~{~~  ~~"documentName": "Contract\_Agreement.pdf",~~  ~~"status": "Signed"~~  ~~}~~ |

**~~Response Body for Failure~~**

|  |
| --- |
| ~~{~~  ~~"status": "Failure",~~  ~~"error": "Envelope ID not found."~~  ~~}~~ |

**~~Endpoint~~**~~: DELETE/retract-document~~

**~~Request Parameters:~~**

|  |  |  |  |
| --- | --- | --- | --- |
| ~~Parameter~~ | ~~Type~~ | ~~Required~~ | ~~Description~~ |
| ~~EnvelopId~~ | ~~String~~ | ~~Yes~~ | ~~The unique ID of the envelope containing the document to be retracted.~~ |

**~~Example Request~~**

|  |
| --- |
| ~~DELETE /retract-document?envelopeId=abc123xyz456~~ |

**~~Response~~**

|  |  |
| --- | --- |
| ~~Status Code~~ | ~~Description~~ |
| ~~200 OK~~ | |  | | --- | | ~~The document was retracted successfully.~~ | |
| ~~400 Bad Request~~ | ~~Invalid input parameters.~~ |
| ~~401 Unauthorized~~ | ~~Invalid or expired access token.~~ |
| ~~500 Internal Server Error~~ | ~~Server-side issue.~~ |

**~~Example Response:~~**

**~~Response Body for Success~~**

|  |
| --- |
| ~~{~~  ~~"status": "Success",~~  ~~"message": "Document retracted successfully."~~  ~~}~~ |

**~~Response Body for Failure~~**

|  |
| --- |
| ~~{~~  ~~"status": "Failure",~~  ~~"error": "Envelope ID not found or already completed."~~  ~~}~~ |

Salesforce APIs

This API suite enables seamless integration with Salesforce, providing functionality to retrieve project details, track updates, manage engagement status, and handle document uploads. These endpoints are designed to streamline operations and ensure effective data synchronization with Salesforce.

**Authentication**:

The API uses **OAuth 2.0** for secure access. You must include a valid access token in the Authorization header.

**Example Header**:

|  |
| --- |
| Authorization: Bearer <access-token> |

**Base URL**: <https://apim.eu.workato.com/exchange>

**Version**: V1

**Available Endpoints**:

|  |  |  |
| --- | --- | --- |
| Endpoint | Method | Description |
| /project-details | GET | Retrieves detailed information about projects stored in Salesforce. |
| /update-engagement-status | PUT | Allows updating the engagement status of projects or accounts in Salesforce to reflect the latest developments. |
| /upload-documents | POST | Enables uploading documents directly to Salesforce, associating them with specific project. |

**Endpoint**: GET/project-details

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Required | Description |
| EngagementNumber | String | Yes | The unique ID of the engagement/project to retrieve details for. |
| BusinessUnit | string | Yes | The identifier of the business unit associated with the project. |

**Example Request**

|  |
| --- |
| GET /project-details?engagementNumber=12345&businessUnit=BU6789 |

**Response**

|  |  |
| --- | --- |
| Status Code | Description |
| 200 OK | Project details retrieved successfully. |
| 400 Bad Request | Invalid input parameters. |
| 401 Unauthorized | Invalid or expired access token. |
| 500 Internal Server Error | Server-side issue. |

**Example Response:**

**Response Body for Success**

|  |
| --- |
| {  "engagementId": "12345FFDBH22456FFD",  "engagementNumber": "ENG-2023-7890",  "engagementName": "Global Equity Fund",  "contractingEntity": "Alpha Contracting Ltd",  "fundDomicile": "Luxembourg",  "legalVehicle": "SICAV",  "strategy": "Equity",  "sfdrCategory": "Article 8",  "investmentManager": "Alpha Asset Management",  "administrator": "Beta Fund Services",  "depositary": "Gamma Depositary Ltd",  "distributor": "Delta Distribution Co.",  "registeredOffice": "1 Finance Street, Luxembourg",  "coSec": "Epsilon CoSec Services",  "mlro": "Zeta Compliance Ltd",  "registrationAgent": "Eta Registration Services",  "minimumMancoFee": 50000,  "onboardingFees": 10000,  "businessUnitId": "BU6789",  "status": "Active"  } |

**Response Body for Failure**

|  |
| --- |
| {  "status": "Failed",  "error": "Engagement Number not found."  } |

**Endpoint**: PUT/update-engagement-status

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Required | Description |
| EngagementID | String | Yes | The unique ID (18 char) of the engagement/project to be updated. |
| ManCOID | String | Yes | The unique ID (18 char) of the MANCO to be updated. |
| BD-Handover-Date | Date | No | BD Handover Date in DD/MM/YYYY format. |
| DD-Legal-Date | Date | No | DD and Legal Date in DD/MM/YYYY format. |
| Board-Approval-Date | Date | No | Board Approval Date in DD/MM/YYYY format. |
| Op-Readiness-Date | Date | No | Operational Readiness Date in DD/MM/YYYY format. |
| Business-Handover-Date | Date | No | Business Handover Date in DD/MM/YYYY format. |

**Example Request**

|  |
| --- |
| {  "engagementID": "123456789GH125FFDE",  "ManCoID": "9984SDFFFDRE99087F",  "bd-handover-date": "21/04/2024",  } |

**Response**

|  |  |
| --- | --- |
| Status Code | Description |
| 200 OK | Engagement status updated successfully. |
| 400 Bad Request | Invalid input parameters. |
| 401 Unauthorized | Invalid or expired access token. |
| 500 Internal Server Error | Server-side issue. |

**Example Response:**

**Response Body for Success**

|  |
| --- |
| {  "status": "Success",  "message": "Engagement status updated successfully."  } |

**Response Body for Failure**

|  |
| --- |
| {  "status": "Failure",  "error": "Invalid engagement number or status value."  } |

**Endpoint**: POST/upload-document

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Required | Description |
| EngagementID | String | Yes | The unique ID of the engagement/project to associate the document with. |
| BusinessUnitID | String | Yes | The unique ID of the Business Unit to associate the document with. |
| FileCategory | String | Yes | Category the file belongs to |
| FileName | String | Yes | The name of the file to be uploaded (including extension). |
| FileContent | String | Yes | File Content as base 64 string. |

**Example Request**

|  |
| --- |
| {  "engagementID": "12345DKFICM22FFD",  “businessunitID”: “5467211SDLKFLSD1”,  "fileCategory": "client contract",  "fileName": "ProjectDetails.pdf",  "fileContent":"VGhpcyBpcyBhIHNhbXBsZSBzaWduZWQgZG9jdW1lbnQgY29udGVudCBp…."  } |

**Response**

|  |  |
| --- | --- |
| Status Code | Description |
| 200 OK | Document uploaded successfully. |
| 400 Bad Request | Invalid input parameters. |
| 401 Unauthorized | Invalid or expired access token. |
| 500 Internal Server Error | Server-side issue. |

**Example Response:**

**Response Body for Success**

|  |
| --- |
| {  "status": "Success",  "message": "Document uploaded successfully."  } |

**Response Body for Failure**

|  |
| --- |
| {  "status": "Failure",  "error": "Invalid folder path or engagement number."  } |

SharePoint APIs

This API suite facilitates interaction with SharePoint for managing files and folders, enabling seamless document storage, access control, and user permissions. The five endpoints are designed to streamline document management and ensure secure access to resources.

**Authentication**:

The API uses **OAuth 2.0** for secure access. You must include a valid access token in the Authorization header.

**Example Header**:

|  |
| --- |
| Authorization: Bearer <access-token> |

**Base URL**: <https://apim.eu.workato.com/exchange>

**Version**: V1

**Available Endpoints**:

|  |  |  |
| --- | --- | --- |
| Endpoint | Method | Description |
| /create-folder | POST | Creates a new folder in a specified SharePoint library or directory to organize documents. |
| /upload-document | POST | Uploads document to a designated folder in SharePoint for easy access and storage. |
| /delete-document | DELETE | Removes a specific document from a designated SharePoint folder. |
| /grant-access | POST | Assigns access permissions to a specific user for a selected file in SharePoint. |
| /revoke-access | POST | Removes previously granted access permissions for a specific user on a selected file. |
| /retrieve-document | GET | Retrieve document from a specified SharePoint folder. |

**Endpoint**: POST/create-folder

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Required | Description |
| SiteName | string | Yes | The name of the SharePoint site where the folder will be created. |
| ParentFolder | string | Yes | The path of the parent folder where the new folder will be created. |
| FolderName | string | Yes | The name of the folder to be created. |

**Example Request**

|  |
| --- |
| {  "siteName": "ProjectSite",  "parentFolder": "/SharedDocuments/Projects",  "folderName": "NewProjectFolder"  } |

**Response**

|  |  |
| --- | --- |
| Status Code | Description |
| 200 OK | Folder created successfully. |
| 400 Bad Request | Invalid input parameters. |
| 401 Unauthorized | Invalid or expired access token. |
| 500 Internal Server Error | Server-side issue. |

**Example Response:**

**Response Body for Success**

|  |
| --- |
| {  "status": "Success",  "FolderId": "12345",  "URL": "https://sharepoint.example.com/sites/ProjectSite/SharedDocuments/Projects/NewProjectFolder",  "message": "Folder created successfully."  } |

**Response Body for Failure**

|  |
| --- |
| {  "status": "Failed",  "error": "Parent folder not found."  } |

**Endpoint**: POST/upload-document

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Required | Description |
| SiteName | String | Yes | The name of the SharePoint site where the file will be uploaded. |
| ParentFolder | String | Yes | The path of the parent folder where the file will be stored. |
| FileName | String | Yes | The name of the file to be uploaded (including extension). |
| FileContent | String | Yes | The content of the file, encoded in Base64 format. |

**Example Request**

|  |
| --- |
| {  "siteName": "ProjectSite",  "parentFolder": "/SharedDocuments/Projects",  "fileName": "ProjectPlan.pdf",  "fileContent": "JVBERi0xLjMKJb/0TR..." // Base64-encoded file content  } |

**Response**

|  |  |
| --- | --- |
| Status Code | Description |
| 200 OK | File uploaded successfully. |
| 400 Bad Request | Invalid input parameters. |
| 401 Unauthorized | Invalid or expired access token. |
| 500 Internal Server Error | Server-side issue. |

**Example Response:**

**Response Body for Success**

|  |
| --- |
| {  "documentId": "12345",  "linkURL": "https://sharepoint.example.com/sites/ProjectSite/SharedDocuments/Projects/ProjectPlan.pdf",  "fileName": "ProjectPlan.pdf"  } |

**Response Body for Failure**

|  |
| --- |
| {  "status": "Failed",  "error": "Parent folder not found."  } |

**Endpoint**: DELETE/delete-document

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Required | Description |
| SiteName | String | Yes | The name of the SharePoint site where the document is located. |
| Folder | String | Yes | The path of the folder containing the document. |
| FileName | string | Yes | The name of the file to be deleted (including extension). |

**Example Request**

|  |
| --- |
| {  "siteName": "ProjectSite",  "folder": "/SharedDocuments/Projects",  "fileName": "OutdatedPlan.pdf"  } |

**Response**

|  |  |
| --- | --- |
| Status Code | Description |
| 200 OK | File deleted successfully. |
| 400 Bad Request | Invalid input parameters. |
| 401 Unauthorized | Invalid or expired access token. |
| 500 Internal Server Error | Server-side issue. |

**Example Response:**

**Response Body for Success**

|  |
| --- |
| {  "status": "Success",  "message": "File deleted successfully."  } |

**Response Body for Failure**

|  |
| --- |
| {  "status": "Failed",  "error": "File not found in the specified folder."  } |

**Endpoint**: POST/grant-access

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Required | Description |
| UserEmail | String | Yes | The email address of the user to whom access is being granted. |
| SiteName | String | Yes | The name of the SharePoint site where the document is located. |
| AccessLevel | String | Yes | The level of access to be granted. Supported values: Read, Write, Full Control. |
| Folder | String | Yes | The path of the folder containing the file. |
| FileName | String | Yes | The name of the file to grant access to. |

**Example Request**

|  |
| --- |
| {  "userEmail": "user@example.com",  "siteName": "ProjectSite",  "accessLevel": "Read",  "folder": "/SharedDocuments/Projects",  "fileName": "ProjectPlan.pdf"  } |

**Response**

|  |  |
| --- | --- |
| Status Code | Description |
| 200 OK | Access granted successfully. |
| 400 Bad Request | Invalid input parameters. |
| 401 Unauthorized | Invalid or expired access token. |
| 500 Internal Server Error | Server-side issue. |

**Example Response:**

**Response Body for Success**

|  |
| --- |
| {  "status": "Success",  "message": "Access granted successfully."  } |

**Response Body for Failure**

|  |
| --- |
| {  "status": "Failed",  "error": "File not found in the specified folder."  } |

**Endpoint**: POST/revoke-access

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Required | Description |
| UserEmail | String | Yes | The email address of the user whose access is being revoked. |
| SiteName | String | Yes | The name of the SharePoint site where the document is located. |
| Folder | String | Yes | The path of the folder containing the file. |
| FileName | String | Yes | The name of the file to revoke access from. |

**Example Request**

|  |
| --- |
| {  "userEmail": "user@example.com",  "siteName": "ProjectSite",  "folder": "/SharedDocuments/Projects",  "fileName": "ProjectPlan.pdf"  } |

**Response**

|  |  |
| --- | --- |
| Status Code | Description |
| 200 OK | Access revoked successfully. |
| 400 Bad Request | Invalid input parameters. |
| 401 Unauthorized | Invalid or expired access token. |
| 500 Internal Server Error | Server-side issue. |

**Example Response:**

**Response Body for Success**

|  |
| --- |
| {  "status": "Success",  "message": "Access revoked successfully."  } |

**Response Body for Failure**

|  |
| --- |
| {  "status": "Failed",  "error": "File not found in the specified folder."  } |

**Endpoint**: GET/GetDocument

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Required | Description |
| SiteName | String | Yes | The name of the SharePoint site where the document is located. |
| Folder | String | Yes | The path of the folder containing the document. |
| FileName | string | Yes | The name of the file to be deleted (including extension). |

**Example Request**

|  |
| --- |
| GET /api/GetDocument?SiteName=ProjectHub&Folder=Reports/2025&FileName=Summary.pdf |

**Response**

|  |  |
| --- | --- |
| Status Code | Description |
| 200 OK | The document was successfully retrieved. |
| 400 Bad Request | Invalid input parameters. |
| 401 Unauthorized | Invalid or expired access token. |
| 500 Internal Server Error | Server-side issue. |

**Example Response:**

**Response Body for Success**

|  |
| --- |
| {  "DocumentName": "Summary.pdf",  "DocumentContent": "JVBERi0xLjQKJcTl8uXrp/Og0MTGCjIgMCBvYmoKPDwv..."  } |

**Response Body for Failure**

|  |
| --- |
| {  "status": "Failed",  "error": "File not found in the specified folder."  } |

Exchange Server APIs

This API provides seamless integration with the Exchange Server to enable programmatic email sending

**Authentication**:

The API uses **OAuth 2.0** for secure access. You must include a valid access token in the Authorization header.

**Example Header**:

|  |
| --- |
| Authorization: Bearer <access-token> |

**Base URL**: <https://apim.eu.workato.com/exchange>

**Version**: V1

**Endpoint**:

|  |  |  |
| --- | --- | --- |
| Endpoint | Method | Description |
| /send-email | POST | Allows sending emails through the Exchange Server by specifying recipients, subject, and email body. |

**Request Parameters:**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Type | Required | Description |
| To | Array | Yes | List of recipient email addresses |
| From | String | No | Sender’s email address. If not provided, then “DoNotReply@waystone.com” |
| CC | Array | No | List of CC recipient email addresses. |
| Bcc | Array | No | List of BCC recipient email addresses. |
| Subject | String | Yes | Subject line of the email. |
| Body | String | Yes | Main content of the email. Supports HTML or plain text. |
| attachment | Array | No | List of attachments in Base64 encoding. Each item must include filename, content, and mime Type. |

**Example Request**

|  |
| --- |
| {  "to": ["recipient@example.com"],  "from": "sender@example.com",  "cc": ["cc@example.com"],  "bcc": ["bcc@example.com"],  "subject": "Project Update",  "body": "<h1>Hello Team</h1><p>Here are the latest updates...</p>",  "attachments": [  {  "filename": "report.pdf",  "content": "JVBERi0xLjMKJb/0TR==",  "mimeType": "application/pdf"  }  ]  } |

**Response**

|  |  |
| --- | --- |
| Status Code | Description |
| 200 OK | Email sent successfully. |
| 400 Bad Request | Invalid input parameters. |
| 401 Unauthorized | Invalid or expired access token. |
| 500 Internal Server Error | Server-side issue. |

**Example Response:**

**Response Body for Success**

|  |
| --- |
| {  "status": "Sent",  "message": "Email sent successfully."  } |

**Response Body for Failure**

|  |
| --- |
| {  "status": "Failed",  "error": "Invalid recipient email address."  } |

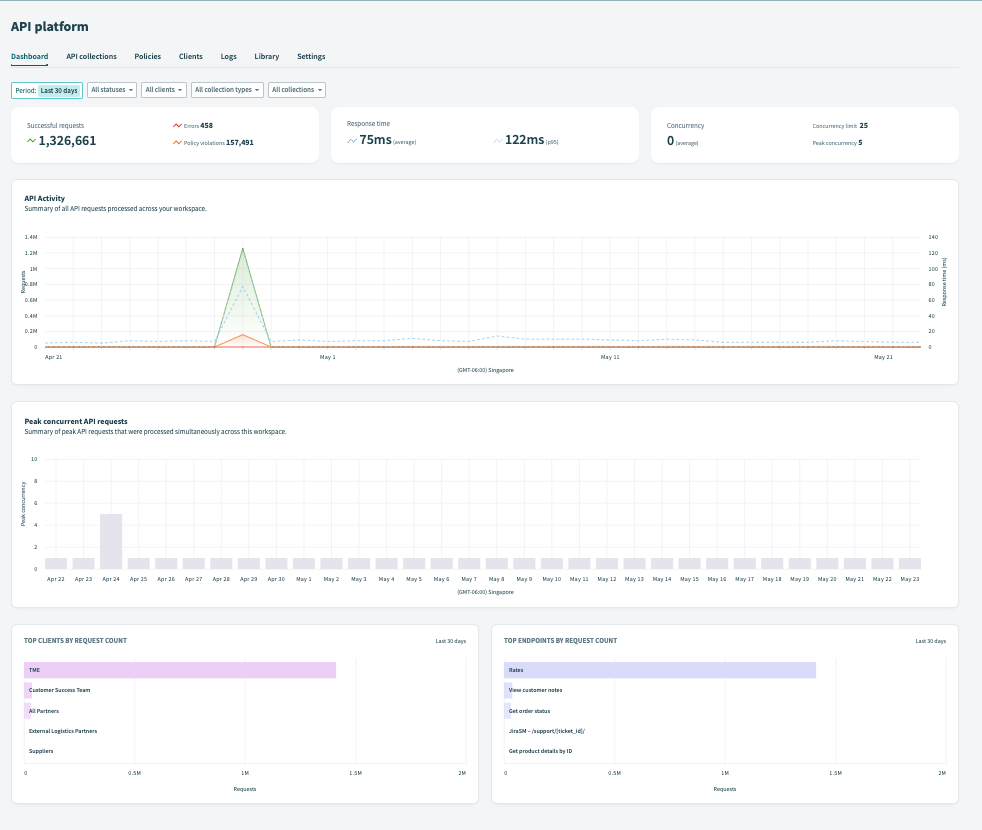
Performance

Workato doesn't provide specific performance benchmarks for API recipes, as performance can vary based on factors like recipe complexity, external system response times, and data volume.

Leverage Workato's API monitoring tools to track metrics such as response times, error rates, and traffic patterns. This data can help identify bottlenecks and areas for optimization.

Observability and Supportability

The Workato API Platform Dashboard is an essential tool for developers and administrators, providing real-time insights into API performance and usage. With features like response time tracking, error rate monitoring, and traffic pattern analysis, the dashboard helps ensure your APIs are running smoothly and efficiently. It also highlights peak concurrent requests and top clients, enabling you to optimize and scale your services effectively. By leveraging these comprehensive metrics, you can maintain high performance, reliability, and security for your API ecosystem.



The Workato API Platform Dashboard is a powerful tool for monitoring and managing your APIs. Here are some key features:

1. **Real-Time Data Visualization**: The dashboard provides real-time insights into the performance of your API endpoints and collections.
2. **API Activity Graphs**: These graphs summarize all API requests processed, helping you identify trends, spikes, or drops in API usage.
3. **Peak Concurrent Requests**: This feature shows the highest number of simultaneous API requests, helping you plan for capacity and ensure your infrastructure can handle peak loads.
4. **Top Clients and Endpoints**: The dashboard highlights the most active API consumers and the most frequently accessed endpoints.
5. **Policy Violations**: Detect and analyze any abnormal API calls that violate predefined policies.
6. **Customizable Filters**: You can apply various filters to modify the dashboard output, such as date ranges, specific clients, and collections.

Resilience and Recovery

Not Applicable

Test Cases

*[Describe any expected test case required, for example Penetration, Performance, Model Office etc. Any specific test cases should be documented based on rationale of each test case]*

1. Security
   1. Security policies
   2. Authentication methods
2. User authentication details provided in 2.3 Security.
3. Control access to resources details provided in 5.1.1 access control.
4. Integration Security/Authentication

To align with Waystone defined architecture, Appian (both application due diligence & client onboarding) will have communicate/integrate with all third party API via Workato except OKTA(single sign on). There would no point-to-point integration between Appian & third parties. While integrating with Workato, oAuth 2.0 Client Grant Credential would be used as authentication type. Below image provides the integration architecture as a solution for both the applications.

A white rectangular object with black text

Description automatically generated

* 1. Encryption standards

*[Specify the encryption standards and protocols that will be employed to protect data at rest and in transit. This section should include details on the types of encryption algorithms used, key management practices, and compliance with industry standards and regulations]*

* 1. Access controls

As Due diligence & Client onboarding applications is going to be used by three different jurisdictions. i.e. IRE, Lux & UK. As per current server configuration, same Appian instances & database would be use for all three jurisdictions. Users would be mapped against their assigned jurisdiction in the application so they would be only able to see the any kind of data/details which would be assigned to them. Users would not be able to view the data/details from cross jurisdiction until they are authorized for the same jurisdiction.

1. Deployment Approach.

Appian

While doing the deployment, will follow direct deployment

* Direct deployment provides a guided push-button deployment of applications, packages, plug-ins, and database scripts across connected environments.
* For deployments to higher environments such as Production, user can require a review process by enabling it.
* The Automated Import Manager automates the process of *inspecting the application*, *executing DDL scripts*, *updating CDTs*, *republishing the appropriate datastores*, and *importing the application*.

A close-up of a black screen

Description automatically generated

Deployment steps

Appian deployment Step

The compare and deploy feature guides you through a few straightforward steps.

* 1. [Select target environment](https://docs.appian.com/suite/help/24.4/Deploy_to_Target_Environments.html#select-the-target-environment).
  2. Select the items you want to deploy. These include your [application](https://docs.appian.com/suite/help/24.4/Deploy_to_Target_Environments.html#prepare-an-application) or [package](https://docs.appian.com/suite/help/24.4/Deploy_to_Target_Environments.html#prepare-a-package), as well as [application configurations](https://docs.appian.com/suite/help/24.4/Deploy_to_Target_Environments.html#add-application-configurations), [database scripts](https://docs.appian.com/suite/help/24.4/Deploy_to_Target_Environments.html#add-db-scripts), and [plug-ins](https://docs.appian.com/suite/help/24.4/Deploy_to_Target_Environments.html#add-plugins).
  3. [Inspect your deployment](https://docs.appian.com/suite/help/24.4/Deploy_to_Target_Environments.html#inspect-the-package) for internal issues.
  4. [Review your deployment settings](https://docs.appian.com/suite/help/24.4/Deploy_to_Target_Environments.html#review-details). This step allows you to upload an import customization file if needed.
  5. [Deploy to your target environment](https://docs.appian.com/suite/help/24.4/Deploy_to_Target_Environments.html#deploy).

OKTA deployment steps

There will be two Prod Okta instances: one for internal users and other is for external users.

Import the bulk user for external Okta .

For both instance we have SP metadata shared from apian side.

Both Internal & External Prod instance has three Appian application to test(dev,test,Prod).

For internal users AD group will be created for external user instance Okta groups will be created.

For external group bulk users will be imported into Okta group.

Assign the created groups in both the instances

Create update of users/groups manage from Appian side.

Workato Deployment steps

Workato portal provides a deployment portal to deploy projects to different environments.

1. Make sure admin has appropriate privileges to deploy in the production environment.
2. Open the project you plan to deploy and click the Deployments tab.
3. Click **Deploy to**. And options will appear. Select Prod.
4. Chose the assets you want to deploy to the production environment.
5. Ensure all dependent assets are included.
6. Click deploy to initiate the deployment.
7. Reconnect connections after deployment completion.

Rollback plans

Appian

Database rollback

Database scripts would be provided for rollback, which need to execute.

Appian Object

Before the deployment, package need to create with the required objects, are going to deploy. In the case of rollback, the same package need to again deploy/import in the target system.

A diagram of a company

Description automatically generated

OKTA

As this is for new app onboarding, we can save the configuration details into document and deactivate the application.

Workato

As this is the first time deployment of the workato Assets to the Production environment. In case of rollback, remove all Assets from the production environment.

Operational Procedures.

Defined in 2.4

1. Threat & Misuse Cases
2. *Correct users need to be assigned DDQ/FO task by the analyst/Project Manager otherwise task/document would be exposed to wrong users, and it can lead to data breach issues.*
3. Risk, Assumptions and Exclusions

Risks

1. *Systems are not monitored regularly.*
2. *Personal Email addresses are not authorised to be used only company emails are approved.*
3. *Third party API processing taking time delaying the respective steps/process.*

Assumptions

1. *Salesforce system will provide all required data items.*

Exclusions

1. *As Appian is Saas based, so threat modelling is not happening via Waystone.*
2. *Scope of the Business unit is limited to MANCO*
3. Glossary of Terms

*[Provide definitions for technical terms and acronyms used in the document. It ensures that all readers have clear understanding of the terminology]*

1. Associated Documents

*[List all documents that offer insight into or have been considered when designing the solution]*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ref | Title | Source | Issue Date | Version |
| *[Reference]* | *[Document Name]* |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

1. Review

*[Repeat the section below for each round of review, until either approved or cancelled]*

Outcome

|  |  |  |
| --- | --- | --- |
| Date | Outcome | Reviewer(s) |
| *[01/01/2024]* | *[Rejected / Accepted / Cancelled]* | *[Name of reviewer(s)]* |

|  |  |
| --- | --- |
| Section | Comment |
| *[Section identifier]* | *[Information regarding any caveats or comments with significant reasoning]* |
|  |  |
|  |  |