SPARK API Industry

Standard Recommendations

This document is a foundation to define what an API (Application Programming Interface) is, what it will be used for, and how it is different from other data exchange practices in the Retirement Plan Industry, such as SFTP (Secure File Transfer Protocol,) and other methods such as RPA. We will describe the typical fields needed in each API service, and how each could apply filters, along with how it would provide confirmation that the data was processed when data is uploaded. We will give an example of an API "payload,", as well as a couple examples of public APIs that can be used as models.

This document is not API requirements documentation. It is an overview to help TPAs and Record Keepers work together to build precise specifications at much higher detail.

We will also discuss security concepts, due to their essential role in data integration.

Data Connectivity Will Drive Market Share

The Pension Industry needs great improvements in the efficiency, accuracy and security of data exchange between Record Keepers and TPAs. Data connectivity drives innovation in client value, which positively impacts the sales pipeline.

This innovation not only drives costs down and increases efficiency for TPAs and Record Keepers, but, most importantly, it does so for Plan Sponsors. Better integration with TPAs will provide cleaner and more accurate data, resulting in increased client satisfaction and more sales opportunities for Record Keepers to provide additional services. For example, those who offer an HSA program, or other financial wellness programs, can expand qualified sales leads through better data, including predictive analytics models.

Ultimately, Record Keepers that drive connectivity to the TPA market will expand market share by increasing efficiency at all levels, attracting more partner and client interest and loyalty.

Connectivity Methods, API and SFTP

Record Keepers should invest in developing an API to support key business processes in the TPA industry and drive further innovation. The commonly accepted standard for an API is REST (REpresentational State Transfer), which is recommended to ensure a consistent and secure protocol. A RESTful API will typically transfer data in a text based JSON (JavaScript Object Notation) format. See Appendix B for an example of JSON.

An API may require significant investment, since the process around the API can be as important as the connection method. Often, APIs process data in real time, or provide reports in real time, but different strategies should be considered based on business needs. The later section, "Data Interchange Approach", will explain in further detail.

Data exchange technology, such as SFTP, can still provide significant benefits and may require less investment, since many Record Keepers already implement SFTP. SFTP often transfers CSV and XML files. Unfortunately, the process behind SFTP is as important as an API. SFTPs are often built without the two-way "request" and "response" methods needed to avoid manual labor. When a file is submitted to SFTP (the request) in order to check if the file was processed without error (the response), staff must navigate to the Record Keeper website manually. Since many SFTP servers were set up years ago, they were built assuming human intervention, and this will take some effort to engineer out of the process.

In the long run, the API is preferred for long-term integration needs of the industry, and both API and SFTP share the same need to engineer the underlying process for best results for the business.

The Role of RPA Bots for the Short Term

This document recognizes the growing popularity of RPA (Robotic Process Automation), which provides tools for businesses to create and schedule bots to crawl Record Keeper websites, essentially logging in to download information while unattended. The RPA industry concentrates on automating tedious tasks performed by humans, but RPA can add further security risk and complications, ultimately limiting the efficiency and innovation that APIs provide. Nevertheless, Record Keepers may need to embrace RPA in the short term if there are no other options for the TPA.

Concerns regarding RPA include:

1. Users share logins to RPA bots, which results in authorized access points being unattended, leaving more surface area for bad actors to take control of sessions.
2. Multi-factor authorization used with phones and email help improve security with the initial login process, but given the duration these sessions are authorized for, and that bots run unattended, they present more opportunity for access to unauthorized users.
3. Most RPA bots are created and run by business users and not by IT, which often are setup and run without extra security protocols available.

In comparison, API security practices reduce the number of human interactions. Coupled with best practice security protocols, they provide clear identification that can be revoked when suspicious activity is detected. APIs allow multiple proven security options to authenticate and identify the service and its purpose, leaving much less surface area for exposure and reducing identity theft and wrongful distribution liability. One example of these protocols is to use client certificates for API communication.

The Need for Better Security Protocols

Record Keepers must hold security as a top priority in all data integrations due to the liability in the industry for identity and retirement theft. In most situations, an authentication login for an API, SFTP and website access leverages a username and some type of password. APIs and SFTP credentials often have very long, computer generated passwords.

APIs and SFTP logins do not have multi-factor authentication, (through a text on your phone, for example) since they are set up for computers to use instead of people. This reduces the number of logins to manage and limits access to select IT professionals, mitigating risk.

It is highly suggested that the additional security measures below are considered to help improve the identification of those accessing Record Keeper data.

1. Access for a particular API or SFTP login is restricted by the public internet IP address of the TPA. This requires the TPA to have a static IP address, which will need additional planning and configuration.
2. Certificates can be used on the computer that is accessing the API or SFTP server. These computers are often central servers and not TPA staff's desktop workstations, so they do not change much over time. Certificates should be created to identify each TPA, and for the ability to revoke individually if the activity is suspect. Although more effort is required in management, an optimized process can be efficient. More importantly, it is more effective in responding to any suspected breach. If Certificates are used, restricting public internet IP addresses is not necessary.
3. Logs should be created in the Record Keeper system detailing what data is accessed and analyzed over time. This will support audits of data access and help analyze non-typical activity that can disable the login until further review.
4. Data provided through API and SFTP should reduce the number of times Social Security and Birth Dates are provided, showing participant identifiers in most data requested and forcing the TPA to access a single point of reference for census data where the participant identifier can be matched to their Social Security Number.

Data Interchange Approach: Scheduled vs Immediate vs Delayed

In planning to provide an API, how to approach the process around the API is important. This can impact how easy and useful the API is to the integrations for the TPA. The following text describes options to consider.

Scheduled Response

Since SFTP use is common in the Pension Industry, the Scheduled method will be discussed to compare what is customary for an API.

Data interchanges such as SFTP make it easier and more secure to transfer data downloads and reports. SFTP typically requires some type of scheduling process that creates reports in an SFTP folder that the TPA can download or that can be directly copied to a SFTP site of the TPA.

Data can also be exported to the Record Keeper for processing via SFTP by uploading the file to the SFTP folder. A schedule is also used to sweep the SFTP folder during the day to look for new files to process. However, acceptance of the data changes (or the explanation of any data errors) is commonly done by a human logging into the website to view the import results. This scheduled approach requires human intervention, is less convenient, and limits overall efficiency.

An API can be structured to work with scheduled reports or scheduling processing. It simply requires some extra effort for the TPA to manage, to keep track of what is available and to continue checking for new data.

Immediate Response

In most cases, an API returns the requested data immediately: like a handshake, the request to and response from the API are simultaneous. An API is normally designed to provide a predefined "menu" of data. The API can simply provide a predefined report, or it may allow filters that limit the data returned. Typical examples of data needed in an API include participant current deferral percentages, census data, trust statements, and transaction details.

An API can also push data changes to the Record Keeper system, such as deferral rates or vesting percentage. When an API pushes changes, it gets an immediate response to confirm the data validity. In cases where complex data operations are required on the uploaded data, requiring more time to process, a second API request (handshake) may be necessary to confirm if the data was processed successfully, or if there were errors, and to obtain the error details.

For larger companies with big data, providing an API with an immediate data response requires planning, since an API opens the door for ‘reports on demand’ that could overtax IT resources if thousands of APIs were requested at the same time. To counter the risk of overtaxing the system, the API provides limits on how many API requests can happen per minute, how much data is returned at once (returned in pages that have been accessed in separate requests), and they can also limit the overall number of requests per day.

Delayed Response

Most Record Keepers don’t provide immediate reporting results for custom filtered reports on their TPA portal websites. Reports are often requested and (depending on how busy their systems are) are available for download minutes later. Or, many times, the report is run ahead of time and is waiting for the TPA to download.

For the same reasons, large updates given to the Record Keeper may take several minutes, even hours, to process. So immediate response to the submitted data may not be feasible.

In the above situations, an API would send either a request for a report, or submit a file for processing. The response would simply show acceptance for a certain identifier. A separate API

call would be programmed to check for the status of the identifier and would either download the report when ready or process the response to the uploaded file when available.

For Record Keepers to adopt an API quickly into their infrastructure, a delayed response approach, similar to how custom reports are provided today, is an option. However, this delayed response requires more overhead and complexity to use the API for software innovation. Nevertheless, the benefits are still great, and programming code can be simplified when the immediate response method is eventually supported.

Initial Requirements for Data to be Included in the API

This section focuses on the primary data fields required for each API service, and the basic structure of how each API service should work.

This is intended to start the conversation with each Record Keeper about mapping all fields that relate to special circumstances and to their special service offerings. Our goal is to create an industry standard for all common data components for Record Keepers, and to build extensions to handle fields unique to each Record Keeper.

A standard for data integration is important, since every company in the industry defines payroll integration differently (either 360 or 180), making it challenging. It is difficult for Plan Sponsors (Clients), payroll companies, and TPAs to build efficiencies around varied definitions and integrations. Furthermore, vendor data is inconsistent, momentum stutters when changing between vendors, which impacts clients' satisfaction and retention. The end game is good, clean data, efficiently and securely transferred between RKs and/or TPAs.

There are six initial API services that would help improve both census review, trust reconciliation, disclosures, and payroll integration and are shown in the following outline.

1. **Employee Census Download**
    a. Census and Indicative
    b. Vesting
2. **Employee Census Upload**
    a. Census and Indicative
    b. Vesting
    c. Contributions
3. **Deferral Elections and Loans Download**
    a. Deferrals
        i. Deferral Changes
        ii. Deferral Complete List
    b. Loans
        i. Loan Changes
        ii. Loan Complete List
4. **Statements Download**
    a. Annual Trust Statements by Participant

   b. Quarterly Trust Statements by Participant
5. **Contribution Payments Download**
   a. Contribution Payment Date
   b. Contribution Payment Allocation

Each of the above outlined points is discussed in the following sections.

Participant Download (Diagram 1)
The foundation for data integration is the participant data, and it can also be critical in helping Record Keepers find opportunities to provide additional value, such as wellness programs. In addition, ensuring better integration with participant data helps provide statements under the new electronic disclosure rules that continue to expand, and in locating lost participants when it comes to “force-outs.” This information can be used to support payroll integration and reviewing a company census each year.

It is also important to discuss what is the primary unique identifier of a participant and what peripheral data can be used to help resolve errors when they exist, such as an SSN error. Ideally, there should be an internal Record Keeper Identifier that can be used to match a participant to related deferral changes and contribution details. The SSN should be available as reference only and in time deprecated from use in many of the reports to help conceal privacy data. The SSN is included below by default, since this transition away from showing SSNs on all reports will take time.

As shown in the Diagram, outside the typical Client Identifier, EE Name, RK Identifier for the participant and the SSN, there are 3 sets of data that could be downloaded. These include Eligibility data, Indicative data and Vesting data

The fields below should be provided to download **Eligibility and Indicative** data...

- Contract Number (Client Number)
- RK Unique Identifier for the Participant (if available)
- Employee ID Number assigned by Payroll Company (if available)
- SSN
- First Name
- Last Name
- Middle Initial
- Name Prefix
- Date of Birth
- Hire Date (Original Date of Hire)
- Rehire Date
- Date of Eligibility

(For dual eligibility plans, may need to be specific to 401k or have one for each source)

- Term Date
- Marital Status (for RMDs)

- Employee Status (Temp, Leave, Seasonal, PT, FT, etc.)
- Last Status Effective Date
- Union Code (Suggest a uniform description for this)
- State of Residence (We need this as sometimes the residence state differs from the location of the business. We need this for beneficiaries.)
- Address 1
- Address 2
- City
- State
- Zip Code
- Country
- Phone - Work
- Phone - Home
- Phone - Cell
- Email - Work
- Email - Home

The searchable filters should include Contract Number at a minimum, but other filters will be appreciated, such as RK Unique Identifier and SSN.

To help provide checks and balances to data integration, it would be ideal to have a second API service for census data that states the status changes over time, either for each participant or as a group. Without this confirmation of what the Record Keeper has noted as a change, the data integration must calculate the changes based on the differences noted between data received, or it will not be able to catch when data is changed and changed back. This can be considered in later specifications but should be a high priority consideration early in the design.

The fields below should be provided to download **Vesting** data...

- Contract Number (Client Number)
- RK Unique Identifier for the Participant (if available)
- Employee ID Number (if available)
- SSN
- First Name
- Last Name
- Middle Initial
- Effective Date
- Money Type
- Vesting %
- Vesting Years of Service
- Long Term Part-Time

Participant Upload (Diagram 2)
As noted in the diagram, the upload for participant data should include the same data that is downloadable that was described in the last section to support payroll integration and year end uploading of vesting percentages. The data to be able to upload includes Eligibility data, Indicative data and Vesting data.

Additionally noted in the diagram, contribution data must be uploaded as part of the payroll integration process. After payroll is reviewed and audited for accuracy and eligibility, the final contributions for each employee must be submitted to the Record Keeper. This API Service should include the following fields at a minimum.

Employee Information
- Contract Number
- Division (if applicable)
- RK Unique Identifier for the Participant (if available)
- Employee ID Number (if available)
- SSN
- First Name
- Last Name
- Middle Initial
- Name Prefix
- Date of Birth
- Additional indicative data may be uploaded along with contributions

Contribution Information
- Pay Period
- Period Hours
- Period Gross Compensation
- YTD Hours
- YTD Gross Compensation
- Contribution Details
    - Type (which could include):
        - EE Deferral
        - EE Roth
        - EE Roth After Tax
        - EE Non-Roth After Tax
        - EE Safe Harbor NonElect
        - ER Match
        - ER Safe Harbor Match
        - ER Student Loan Match
        - ER QNEC
        - ER Roth Matching
        - ER Profit Sharing
        - PLA ESA

        - QACA
        - Tax Credit Payment
    - Recordkeeper Type Code
        - As prescribed by the RK
    - Fund (If there are multiple cash accounts to fund)
    - Amount

Loan Payment Information

- Loan Number
- Payment Amount
- Indicate if more than one loan

In submitting the contribution data to the API Service, it is expected a unique identifier will be returned with the status of acceptance. Initially, this may be only "Received", and a second API Service can be constructed to provide the final status of the uploaded data. This second status service would accept the unique identifier for the submitted data and return "Success" when applicable. If there were data errors, a list of error details by participants would be returned.

Deferral Elections and Loans Download (Diagram 3)
It is important for payroll integration that before payroll is run, the TPA can verify the latest deferral percentages, evaluate eligibility, and compare to any changes made in payroll. As the diagram shows, this includes downloading changes to election percentages and changes to loans.

Participant **Deferral Elections** Example Fields

- Contract Number
- RK Unique Identifier for the Participant (if available)
- Employee ID Number (if available)
- SSN
- Employee Fields for Reference (Assists when identifiers are in question)
    - First Name
    - Last Name
    - Middle Initial
    - Name Prefix
- Effective Date
- Before Tax Deferral %
- Before Tax Deferral $
- Roth Deferral %
- Roth Deferral $
- After Tax Non-Roth Deferral %
- After Tax Non-Roth Deferral $
- Annual Base Salary
- Bonus Pay

- Event Type (Codes for Auto Enrollment, Rehire, etc.)
- Contribution Type (Codes for Elective, Catch Up, etc.)

(We'll need to catch-up for Secure 2.0)

- Pay Type (Regular or Bonus, which could note if the election is a special election that is temporary for a bonus)
- Last Changed Date (if available)

The above data should be provided as of the current date requested, and it would be ideal if a history of changes could be provided when a filter is requesting changes over a period of time.

Tracking loans from their inception through the term of the loan is a key requirement. The fields below should be available and provided for any loan that is still active when requested.

Loan Detail Example Fields

- Contract Number
- RK Unique Identifier for the Participant (if available)
- Employee ID Number (if available)
- SSN
- Employee Fields for Reference (Assists when identifiers are in question)
    - First Name
    - Last Name
    - Middle Initial
    - Name Prefix
- Loan Identifier
- Event Date
- Event Type (Loan withdrawal or payment)
- Qualified Disaster Loan
- First Payment Date (Applicable if event type is withdrawal)
- Current Loan Balance
- Payment Amount Goal
- Last Payment Amount Made
- Last Payment Date
- Payment Suspension Date
- Maternity Leave
- Military Leave
- Loan Last Modified Date (if terms or other changes were made)

Filters for the API service should include a Contract Number and a date range filter Event Date to capture changes over a select period.

Statements Download (Diagram 4)

As noted in the diagram, trust statements are needed to support the annual and possibly quarterly reconciliation process. These statements are needed in data form, such as CSV, XML

or JSON so the data can be integrated into software solutions. In addition, a PDF version is also requested to support the auditing processes to validate totals when required.

See the diagram for further details.

Contribution Payments Download (Diagram 5)
As part of payroll integration, once the contribution data is accepted by the Record Keeper, it is important the TPA can confirm that the Plan Sponsor sends payment on time. The following fields should be included:

Contribution Payments Example Fields

- Contract Number
- Payment Received Date
- Payment Invested Date
- Payment Amount
- Contribution Made Timely (for 3(16)s)

In addition, the API service should provide a more detailed download that shows how the funds were allocated.

Contribution Payments Allocated Example Fields

- Contract Number
- Payment Received Date
- Payment Invested Date
- Unique RK Participant Identifier
- SSN (Optional)
- Employee First Name
- Employee Last Name
- Payment Amount

Examples of APIs from Other Companies

In our experience, some APIs are better than others. There are simpler versions such as with Clockify, a time tracker application that integrates into payroll services, https://clockify.me/developers-api. This documentation creates some struggle, since it is less clear and accurate, but it is still very workable and its usage limits are not too restrictive.

There are more sophisticated APIs that provide better documentation, and even tools to test the API with a multitude of programming languages. PrismHR's API is excellent, https://api-docs.prismhr.com/, and should be considered one of the examples to model.

Addendum A - Diagrams

Diagram 1 - Census, Indicative, Vesting Data Download

Supports Census Review and Payroll Integration

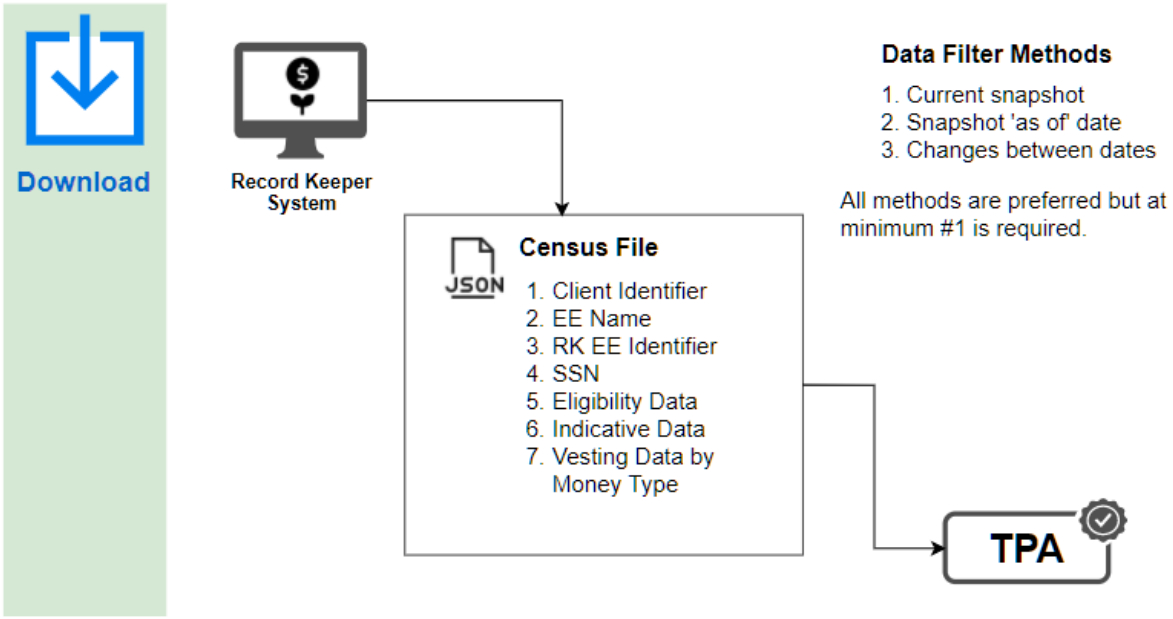


Diagram 2 - Census, Indicative, Vesting and Contribution Data Upload

Supports Census Review and Payroll Integration

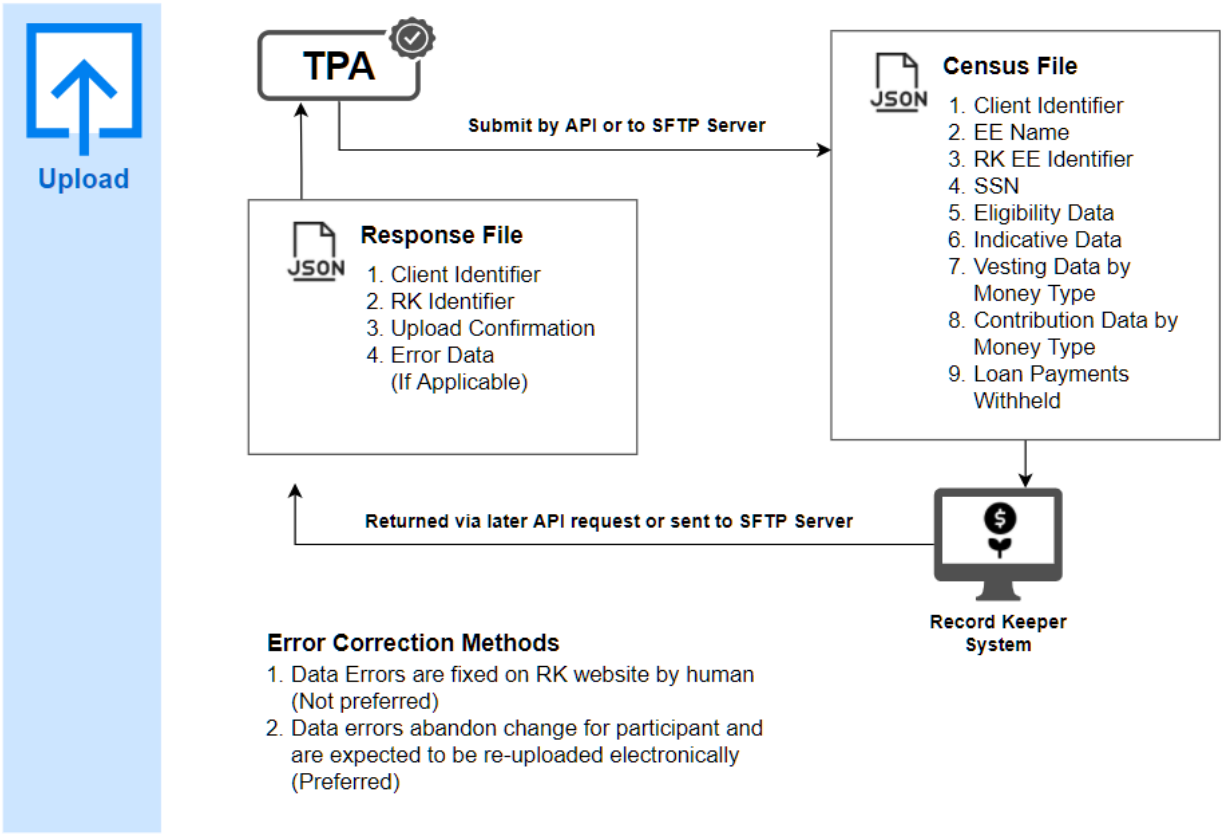
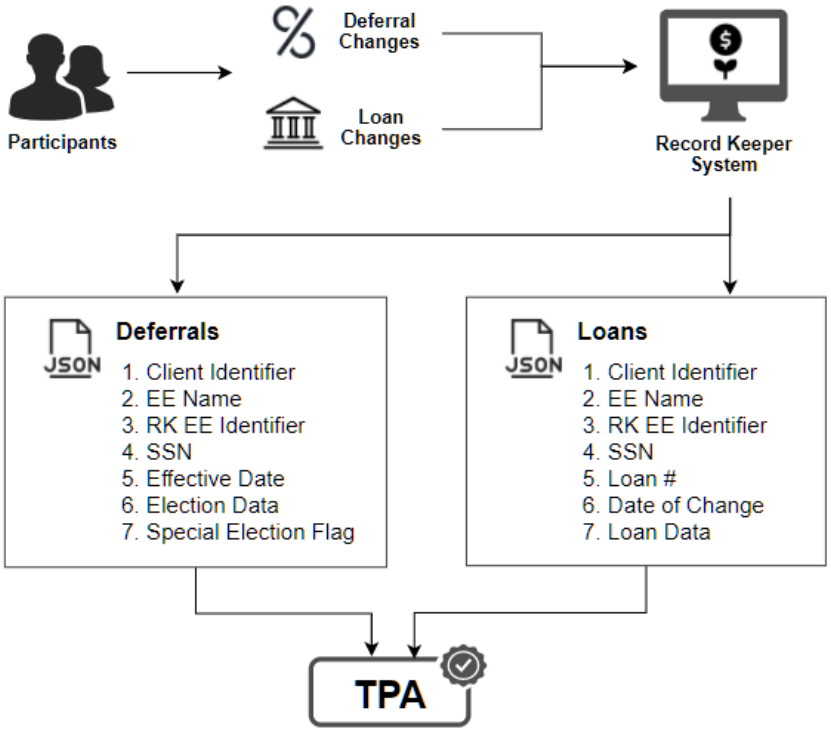


Diagram 3 - Deferral Elections and Loans

Supports Payroll Integration

Download



Data Filter Methods

1. Current snapshot (preferred)
2. Changes between dates (preferred)
3. Snapshot 'as of' date (preferred)
4. Changes for a set period, Year, Quarter or Month

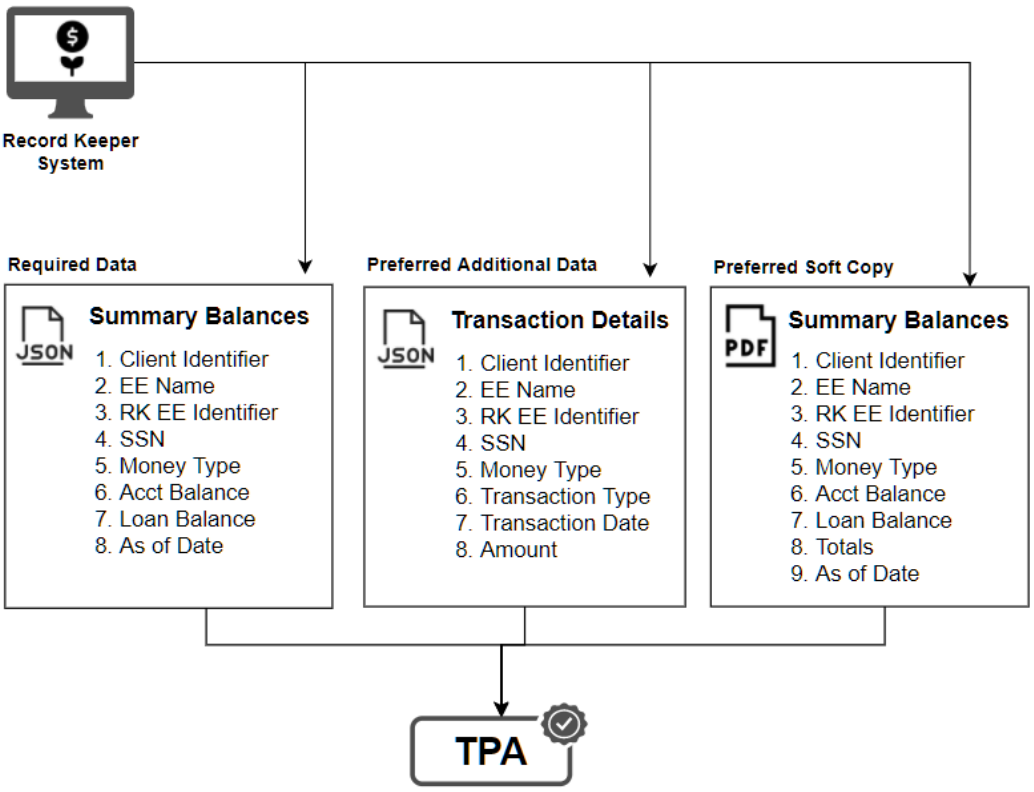
Data Transfer Methods

1. Push to target SFTP server as changes are made (Preferred)
2. Push to RK SFTP server for collection by TPA system
3. By API Request from TPA system

Diagram 4 - Participant Balances (Annual and Quarterly)

Supports Trust Reconciliation

Download



Data Filter Methods

1. Specified Quarter or Year
2. For Detail File, filter by transaction date is preferred

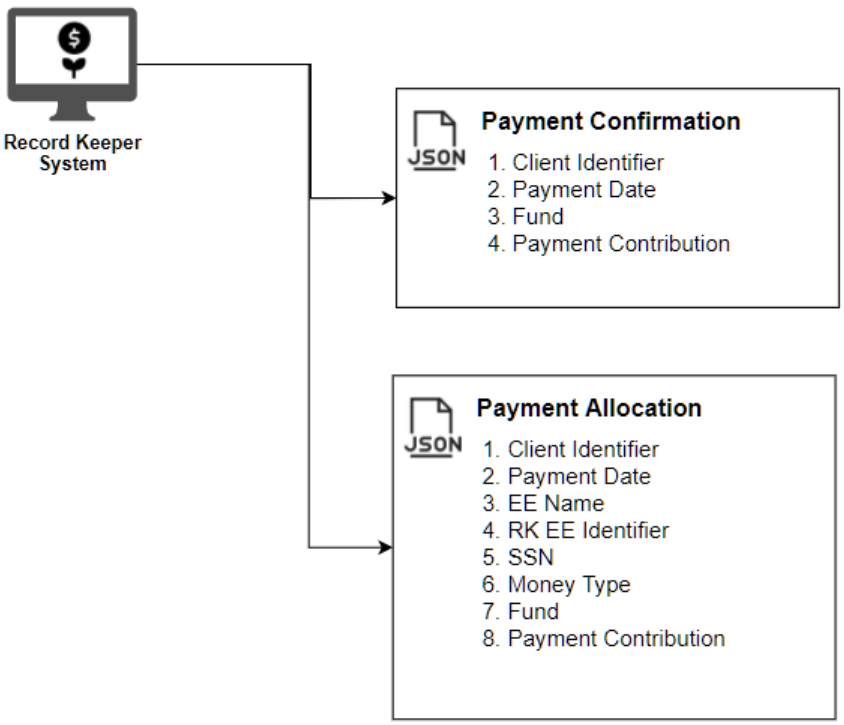
Data Transfer Methods

1. Push to target SFTP server as Quarter and Year data available (Preferred)
2. Push to RK SFTP server for collection by TPA system
3. By API Request from TPA system

Diagram 5 - Contribution Payments

Supports Trust Reconciliation and Payroll Integration

Download



Data Filter Methods

1. Payment Date Filter
2. Client Filter

Addendum B - Example JSON

Example JSON Data for an API Service

An API works by first authorizing the request, which is out of the scope of this document. In response to a request for a certain contract, the API will return the data in a specific format called JSON. Software developers are familiar with the JSON format and can use the data to integrate TPA solutions. Below is an example that shows the Request Detail in the first section with two data points, date stamp and detail count. Then the Census Detail section shows two contracts.


```
{
  "request detail": {
    "date stamp": "2020-08-10 05:01:10",
    "detail count": 2
  },
  "census detail": [
    {
      "contract_number": 20390,
      "rk_identifier": 511470,
      "ssn": "325-35-8727",
      "first_name": "Sherye",
      "last_name": "Wainscot",
      "name_prefix": "Mrs",
      "address_1": "97531 Northfield Way",
      "address_2": null,
      "city": "Roanoke",
      "state": "VA",
      "zip_code": "24034",
      "country": "USA",
      "state_of_residence": "NY",
      "email": "swainscot0@independent.co.uk",
      "birth_date": "1976-03-07",
      "hire_date": "2013-07-10",
      "employee_status": "active",
      "last_status_effective_date": "2020-02-17"
    },
    {
      "contract_number": 20390,
      "rk_identifier": 426817,
      "ssn": "622-22-6041",
      "first_name": "Rickie",
      "last_name": "Ulrik",
      "name_prefix": "Rev",
      "address_1": "0 Banding Center",
      "address_2": null,
      "city": "Orlando",
      "state": "FL",
      "zip_code": "32854",
      "country": "USA",
      "state_of_residence": "FL",
      "email": "rulrik1@nhs.uk",
      "birth_date": "1966-07-22",
      "hire_date": "2015-12-02",
      "employee_status": "leave",
      "last_status_effective_date": "2019-12-16"
    }
  ]
}
```

Addendum C: “GET” File Standard


```
{
"person": {
"individualId": 0,
"name": {
"first": "string",
"middle": "string",
"last": "string",
"suffix": "string",
"title": "string",
"mailingName": "string"
},
"birthDate": "2024-05-02",
"ssn": {
"ssn": "string",
"extension": 0,
"extensionReasonCode": "string"
},
"gender": "MALE|FEMALE|NON-BINANRY|UNKNOWN",
"maritalStatus": "MARRIED|SEPARATED|DIVORCED|WIDOWED",
"deathDate": "2024-05-02",
"languageCode": "str",
"lastNonAdminPaidOffLoanDate": "2024-05-02",
"ethinticityCode": "st",
"nonResidentAlienIndicator": true,
"deathNotificationDate": "2024-05-02",
"birthDateDefaultIndicator": true,
"contact": {
"addresses": [
{
"effectiveDate": "2024-05-02",
"terminationDate": "2024-05-02",
"primaryResidenceIndicator": true,
"addressLine1": "string",
"addressLine2": "string",
"addressLine3": "string",
"county": "string",
"city": "string",
"stateOrProvince": "st",
"country": "string",
"type": "string",
"postalCode": {
"zipCode": 0,
"zipCodePlus4": 0,
```

"zipCodePlus4Plus2": 0
},
"relativeIndividualId": 0,
"defaulCode": "st",
"restrictIndicator": true, "validationSourceCode": "string", "mailHoldDate": "2024-05-02",
"doNotValidateIndicator": true
}
],
"phones": [
{
"type": "MOBILE|OFFICE|HOME|FAX|PERSONAL|OTHER",
"countryCode": "string", "number": "string", "extension": "string", "primaryUse": true,
"internationalPhoneNumberSource": "string"
}
],
"emails": [
{
"address": "string",
"type": "string",
"primaryUse": "s", "emailStatus": {
"invalidElectronicAddressFlag": true,
"invalidElectronicAddressReasonCode": "0001|0008|0012|0013|0014|0015|0016|0017|0022",
"invalidElectronicAddressDate": "2024-05-02",
"confirmedElectronicAddressFlag": true
}
}
],
"sites": [
{
"socialMediaName": "string", "url": "string",
"userName": "string", "followers": 0,
"posts": 0,
"connections": 0, "verified": true
}
]
},
"dataProcessingDate": "2024-05-02",
"dataProcessingContactVerificationDate": "2024-05-02"
},
"employment": [
{
"groupClientId": 0,

"employerAssignedId": "string",
"hireDate": "2024-05-02",
"terminationDate": "2024-05-02",
"percentOfOwnership": 0, "highlyCompensatedIndicator": true, "officerIndicator": true,
"jobDescription": "string", "employmentType": "string", "terminationReasonCode": "string",
"soxReportIndicator": true, "employerClassificationCode": "string", "insiderIndicator": true,
"superOfficerIndicator": true, "unionIndicator": true, "overseasIndicator": true,
"fullTimePartTimeIndicator": "PART-TIME|FULL-TIME", "employerSSOId": "string",
"federalInsuranceContributionActCompensationIndicator": true,
"terminationNotificationDateTime": "2024-05-02T18:20:38.247Z", "overseasEffectiveDate":
"2024-05-02",
"eventId": 0,
"dataProcessingTerminationChangeDateTime": "2024-05-02T18:20:38.247Z",
"dataProcessingEmployerClassificationChangeDateTime": "2024-05-02T18:20:38.247Z",
"leaveOfAbsenceReasonCode": "string"
}
],
"income": [
{
"groupClientId": 0,
"hireDate": "2024-05-02",
"employeeTypeCode": "s", "salary": 0, "salaryAmountQualifier": "st", "familyIncome": 0,
"familyGroup": 0, "familyRelationship": "string", "taxWithHoldExemptNumber": 0,
"taxFilingStatusCode": "s", "taxExemptIndicator": true, "employeeTaxExempted": "s",
"commissionIndicator": true, "effectiveDate": "2024-05-02"
}
],
"participantAccounts": [
{
"individualId": 0,
"groupAccountId": "string",
"accountStatus": "string",
"accountSubsetStatus": "string",
"disabilityDate": "2024-05-02",
"partCashHoldCode": "str",
"effectiveDate": "2024-05-02",
"statusCode": "st",
"statusEffectiveDate": "2024-05-02",
"statusSubCode": "str",
"statusChangeDataProcessingDate": "2024-05-02",
"ownershipIndicator": "N|Q|T",
"takeOverTypeCode": "stri",
"ownershipSettlementDate": "2024-05-02",
"appCompletionCode": "string",

"pinAuthCode": "s",
"restrictCode": "stri",
"restrictNarrative": "string",
"eligibility": [
{
"individualId": 0,
"groupAccountId": "string",
"participationDate": "2024-05-02",
"participationDateSource": "stri",
"eligibilityIndicator": true,
"suppressAutoEnrollIndicator": true,
"inEligibilityReasonCode": "string",
"eligibilityDate": "2024-05-02",
"enrollInviteDate": "2024-05-02",
"enrollInviteRelayDate": "2024-05-02",
"enrollInviteNotificationDate": "2024-05-02",
"enrollPINDate": "2024-05-02",
"outOfRetirement": "s",
"crossPlanId": "s",
"eligibilityRuleNotifyId": 0,
"eligibilityRuleCriticalId": 0,
"enrollmentSuppressIndicator": true,
"longTermPartTermIndicator": true,
"standardMoneyTypeCode": "str",
"groupDefinedMoneyTypeNumber": 0,
"dataProcessingDate": "2024-05-02",
"dataProcessingContactVerificationDate": "2024-05-02"
}
],
"divisions": [
{
"payrollDivisionBasis": "stri",
"payrollDivisionValue": 0,
"payFrequency": "MONTHLY|SEMI-MONTHLY|BI-WEEKLY|WEEKLY|QUARTERLY| ANNUALLY",
"terminationDate": "2024-05-02",
"groupClientSubsetBasis": "stri", "groupClientSubsetValue": 0,
"effectiveDate": "2024-05-02", "clientDivisionCode": "string", "usePayrollIndicator": true
}
]
}
]
}