FINWORKS
INFORMATION TECHNOLOGY SOLUTIONS

Accounts Application Program Interface (API)

The URL generally has the form /api/<resource>/<action>, which could either retrieve a result or perform some action.

When retrieving a result, the API expects a HTTP GET method, and when performing an action it expects a HTTP POST.

The resource /api/accounts refer to all the accounts in the database. To address a specific account, use a URL of the form /api/accounts/{account number}/<action>.

Successful results are returned as HTTP 200 or 201 responses, with the body containing JSON data. Defined error responses will be in the HTTP 400 range, and undefined server errors in the HTTP 500 range.

Error responses contain the following JSON format:

```
{
	"errors" : [
		{
			"description" : "<error description>",
			"code" : "<error code>",
			"index" : <integer>
		}
	]
}
```

where the error description is a detailed description of the error, and the error code is a short reference to the error.

Index is an optional error field and is present when the request data is an array of objects. It indicates the position of the offending object in the request array.

FINWORKS
INFORMATION TECHNOLOGY SOLUTIONS

Supported API calls

1 Total market value 3
2 Market value 3
3 Create account and person 4
4 Create account and company 8
5 Create account 11
6 Method of payment 16
7 Create investment 17
8 Update recurring investment 20
9 Create withdrawal 21
10 Individual statement 28
11 Combined statement 31
12 Holdings 34
13 Available Holdings 35
14 Advisor accrued fees 37
15 Advisor fees (taken) 38
16 Policy document 39
17 Account details 40
18 Update Account details 44
19 Available instruments 47
20 Documents 48
21 Instructions 49
22 Investment Summary Since Inception 52
23 Investment Summary for Period 54
24 Products 55
25 Product available instruments 56
26 Transactions for Period 57
27 Banks 60
28 Value available for withdrawal 60
29 Regulation 28 status for new accounts 61

1 Total market value

GET: /api/accounts/totalMarketValue (as of today)

GET: /api/accounts/totalMarketValue?date={date} (as of date specified)

Note: Date must be ISO format string YYYY-MM-DD

Example JSON response:


```
{
        "amount" : "1199.9800",
        "currency" : "ZAR",
        "accrued fees" : "0.60",
        "date" : "2021-12-31"
}
```


2 Market value

GET: /api/accounts/{accountNumber}/marketValue (as of today)

GET: /api/accounts/{accountNumber}/marketValue?date={date} (as of date specified)

Note: Date must be ISO format string YYYY-MM-DD

Example JSON response:


```
{
        "amount" : "599.9900",
        "currency" : "ZAR",
        "accrued fees" : "0.02",
        "amount in default currency" : "599.9900",
        "accrued fees in default currency" : "0.02",
        "amount in preferred currency" : "599.9900"
        "accrued fees in preferred currency" : "0.02",
        "date" : "2021-12-31"
}
```


Error examples:

HTTP response	Code	Description
400	AccountInactiveError	Account is inactive
404	AccountNotFound	Could not find account

3 Create account and person

POST: /api/accounts/createPersonAndAccount

Example JSON request:


```
{
	"idNumber": "1234567890",
	"identificationDocumentType": "Passport",
	"firstNames": "Jane",
	"surname": "Doe",
	"initials": "J",
	 "middleNames": "Angela Prime",
	"title":"Mrs",
	"countryOfBirth":"MU",
	"accountNumber": "CUSTOM1004",
	"productName": "The Investment",
	"clientGroup": "Direct Client",
	"dateOfBirth": "1991-11-19",
	"gender": "Female",
	"email": "jane@doe.com",
	"taxNumber": "P1234056",
	"cellPhoneNumber": "+27128092343",
	"workPhoneNumber": "+27128092343",
	"homePhoneNumber": "+27128092343",
	"distributionOption": "reinvestDistribution",
	"contractStatus":"New business",
	"address": {
		"line1": "Unit 2",
		"line2": "17 Awkward Road",
		"line3": "",
		"city": "Gotham",
		"code": "1234",
		"country": "ZA",
		"type": "Residential"
	},
	"nationality": "ZA",
	"countryForTaxPurposes": "ZA",
	"isTaxResidentInCountryForTaxPurposes": true,
	"isUSACitizen": false,
	"isPoliticallyExposed": false,
	"receivesIncomeFromUSA": false,
	"employmentStatus": "Self-employed",
	"occupation": "Accountant",
	"occupationIndustry": "Accountancy",
	"sourceOfFunds": "Transfer from RFI",
	"totalAnnualEarnings": "<25000",
	"bankDetails": {
		"verified":true,
		"holderName": "John Doe",
		"accountNumber": "1234567890",
```

```
                "branchCode": "222626",
                "type": "Current",
                "bankName": "First National Bank",
                "branchName": "Centurion",
                "currencyIdentifier": "zar",
                "swiftBankIdentifierCode": "777",
                "intermediaryName": "NEDWXYZYYYY",
                "internationalBankAccountNumber": "GB82WEST12345698765432"
        },

        "communicationOptions": {
                "automaticallyEmailStatement": true,
                "communicationPreference": "Email",
                "transactionNotificationPreferences": {
                        "CompleteWithdrawal":"Email",
                        "SuccessfulLogin": "Both"
                }},
        "Fees": {
                "Ongoing fee method": "From Instrument",
                "Ongoing fee instrument": {
                        "Instrument id": 23456789},
                "feeRates": [{
                                        "feeType": "Administration Fee",
                                                "rate":{
                                        "type": "Percentage",
                                        "value": "0.15"},
                                        "isOverride": true
                                }],
        "entityRelationships": [
                {
                        "relationshipType": "Spouse",
                        "idNumber": "8888",
                        "firstNames": "John",
                        "initials": "J",
                        "surname": "Doe"
                }
        ]
}
```


Notes:

Mandatory fields are:

- idNumber
- firstNames
- surname
- initials (defaults to first letter of firstNames)
- productName

The following restrictions apply:

- productName: must be the name of an existing product in the system

- dateOfBirth: a valid date in various formats, preferably ISO format
- clientGroup: the name of a valid client group as set up in the system configuration
- gender: Male or Female
- email: a valid email address
- address - type: Residential, Registered, Business or Postal
- address - country: a valid ISO country code (as set up in the system configuration)
- countryOfBirth: a valid ISO country code (as set up in the system configuration)
- title: a valid title (as set up in the system configuration)
- contractStatus: One of the statuses in System settings/ Account statuses. The transition from the existing status to the new status should also be valid
- distributionOption : can be one of: reinvestDistribution,payoutDistribution or leaveDistributionInCash

- accountNumber: if specified in the call, any alpha numeric characters may be used. An error will be returned if a contract exists already exists on the system with the same accountNumber

- under communicationOptions, the following apply:
    - communicationPreference can be one of: 'Post' 'Email' 'No consent given' or 'Opted-out'
    - automaticallyEmailStatement is a boolean
    - transactionNotificationPreferences is a list of types: SubmitInvestment, CompleteInvestment, SubmitWithdrawal, CompleteWithdrawal, SubmitSwitchInstruction, CompleteSwitchInstruction, SubmitUnitTransferInInstruction, CompleteUnitTransferInInstruction, SubmitUnitTransferOutInstruction, CompleteUnitTransferOutInstruction, SubmitReinvestmentSwitchInstruction, CompleteReinvestmentSwitchInstruction, ApprovedChangeRequest, SuccessfulLogin, SendAutomaticStatement, the preference can be one of ‘Both’ ‘Email’ ‘SMS’ or ‘None’ . By default, none are set

Optional fields with their defaults if not present:

- "identificationDocumentType": "Identity Document",
- "identificationDocumentCountry"
    - use ISO Code
    - If not provided, we will default to default country on System -> Configuration
- accountNumber: a system assigned number starting with the prefix set up in the system configuration
- firstNames: blank
- surname: blank
- clientGroup: set to the system default
- cityOfBirth: blank
- dateOfBirth: nil
- gender: blank
- email: blank
- cellPhoneNumber: blank
- workPhoneNumber: blank
- homePhoneNumber: blank
- address:
    - city, line1, line2, line3, code: blank
    - type: Residential
    - country: default country set up in system configuration

- taxNumber: blank
- employmentStatus: Options - System - Risk Assessment Employment Status Type
- occupation: Options - System - Risk Assessment Occupation Type
- occupationIndustry: Options - System - Industries
- sourceOfFunds: Options - System - Risk Assessment Source Of Funds
- totalAnnualEarnings: Options - System - Risk Assessment Total Annual Earning
- bankDetails
- advisor
    - advisorMandate - see create account for details
- Fees - see Create Account for more information
- livesAssured - see Create Account for more information
- cashProvision - see Create Account for more information
- lowCashProvision - see Create Account for more information
- Interest or Dividend Withholdings Tax Exemption Status, Reason Code and Reason Description: See Entities API

Example JSON response:


```
{
    "accountNumber": "CUSTOM1004",
    "entityUniqueId": 123456
}
```


Error examples:

HTTP response	Code	Description
400	ExistingPersonWithID NumberFound	A person matching the idNumber already exists
400	MalformedRequest	Malformed request. Field "idNumber" is compulsory
400	MalformedRequest	Malformed request. Invalid JSON
400	MalformedRequest	Invalid title \"Mrs\"
400	ValidateError	If the entity is interest withholding tax exempt and \"other\" is chosen as the reason, please provide a detailed description for the reason

		• See Entities API document for details

4 Create account and company

POST: /api/accounts/createCompanyAndAccount

Example JSON request:


```
{
        "companyName": "ABC PRINTERS (PTY)LTD",
        "companyType": "Listed company",
        "registrationNumber": "71/12707/07",
        "registeredCountry": "ZA",
        "registrationDate": "1999-01-02",
        "accountNumber": "CUSTOM1005",
        "clientGroup": "Direct Client",
        "distributionOption": "reinvestDistribution",
        "productName": "The Investment",
        "email": "abcprinters@mail.com",
        "taxNumber": "1234/123/12/1",
        "isTaxResidentInCountryForTaxPurposes": false,
        "cellPhoneNumber": "+27123457890",
        "workPhoneNumber": "+27123457891",
        "homePhoneNumber": "+27123457892",
        "address": {
           "line1": "Unit 2",
           "line2": "17 Awkward Road",
           "line3": "Ghost Town",
           "city": "Gotham",
           "code": "1234",
           "country": "ZA",
           "type": "Residential"
        },
        "advisor": {
           "advisorCode": "AGFSPB",
           "brokerageCode": "000017"
        },
        "bankDetails": {
           "verified":true,
           "holderName": "John Doe",
           "accountNumber": "1234567890",
           "branchCode": "222626",
           "type": "Current",
           "bankName": "First National Bank",
           "branchName": "Centurion",
           "currencyIdentifier": "zar",
```

```
        "swiftBankIdentifierCode": "777",
        "intermediaryName": "NEDWXYZYYYY",
        "internationalBankAccountNumber": "GB82WEST12345698765432"
      },
      "entityRelationships": [
        {
                "relationshipType": "Director",
                "idNumber": "8888",
                "firstNames": "John",
                "initials": "J",
                "entityType": "Person",
                "surname": "Doe"
        }
      ]
}
```


Notes:

Mandatory fields are:

- productName
- companyName
- companyType
- registrationNumber
- entityRelationships-entityType

The following restrictions apply:

- productName: must be the name of an existing product in the system
- companyType: the name of a valid Company type as set up in the system configuration
- clientGroup: the name of a valid client group as set up in the system configuration

Optional fields with their defaults if not present:

- accountNumber: a system assigned number starting with the prefix set up in the system configuration. If specified in the call, any alpha numeric characters may be used. An error will be returned if a contract exists already exists on the system with the same accountNumber
- registeredCountry: default country set up in system configuration
- registrationDate: blank
- email: blank
- cellPhoneNumber: blank
- workPhoneNumber: blank
- homePhoneNumber: blank
- address:
    - city, line1, line2, line3, code: blank

    - type: Residential
    - country: default country set up in system configuration
- taxNumber: blank
- bankDetails
- advisor
    - advisorMandate - see create account for details
- Fees - see Create account
- Interest or Dividend Withholdings Tax Exemption Status, Reason Code and Reason Description: See Entities API

Example JSON response:

{

"accountNumber": "CUSTOM1005",

"entityUniqueId": 123456

}

Error examples:

HTTP response	Code	Description
400	ExistingCompanyWithRegitrationNumberFound	A company matching the registrationNumber already exists
400	CompanyNotFound	Could not find company
400	ValidateError	If the entity is interest withholding tax exempt and \"other\" is chosen as the reason, please provide a detailed description for the reason • See Entities API document for details

5 Create account

POST: /api/accounts/createAccount

Example JSON request:


```
{
	"idNumber": "1234567890",
	"accountNumber": "CUSTOM1004",
	"policyNumber": "POL1004",
	"productName": "The Investment",
	"clientGroup": "Direct Client",
	"advisor": {
		"advisorCode": "AD1234",
		"brokerageCode": "BR777",
		"advisorMandate": {
			"changeDetails": false,
			"invest": false,
			"switch": false,
			"withdraw": false
	}
	},
	"Fees": {
		"Ongoing fee method": "From Instrument",
		"Ongoing fee instrument": {
			 "Instrument id": 23456789
		},
		"feeRates": [
			{
				"feeType": "Administration Fee",
				"rate": {
					"type": "Percentage",
					"value": "0.15"
				},
				"isOverride": true
			}
		],
		"feeSharing": [
			{
				"feeType": "Administration Fee",
				"sharing": [
					{
						"recipientCode": "DCA001xxx",
						"recipientType": "distributor",
						"percentage": {
							"type": "Percentage",
							"value": "50"
						}
```

```
                                        },
                                        {
                                                "recipientCode": "AACL001xxx",
                                                "recipientType": "advisor",
                                                "percentage": {
                                                        "type": "Percentage",
                                                        "value": "50"

                                                }
                                        }
                                ]
                        }
                ]
        },
        "contractOwners": [
                {
                        "idNumber": "1234567890"
                },
                {
                        "idNumber": "1234567891"
                }
        ],
        "beneficiaries": [
                {
                         "idNumber": “xxx”,
                         “isPrimary” :  true,
                         “relationshipType” : “Spouse”,
                         "benefit": {
                                "type": "Percentage",
                                "value": "50.00"
                        }
                },
                {
                        "idNumbe": “xxx”,
                        “isPrimary” :  false,
                        “relationshipType” : “Child”,
                        "benefit": {
                                "type": "Percentage",
                                "value": "50.00"
                        }

                }
        ],
        "livesAssured": [
                {
                        "idNumber": "xxx"
                },
```

```
                {
                        "idNumber": "yyy"
                }
        ],
        "cashProvision": {
                "isOverridden": true,
                "reason": "Minimum liquidity required for monthly withdrawals",
                "percentage": {
                        "type": "Percentage",
                        "value": "20"
                }
        },
        "lowCashProvision": {
                "isOverridden": true,
                "reason": "Set to maintain minimum trading buffer",
                "percentage": {
                        "type": "Percentage",
                        "value": "5"
                }
        }
}
```


Notes:

Mandatory fields are:

- idNumber (for person) or registrationNumber (for company) or entityUniqueId (for either company or person)
- productName

The following restrictions apply:

- productName: must be the name of an existing product in the system
- clientGroup: the name of a valid client group as set up in the system configuration
- beneficiaries: “value” must be submitted as a “number” (i.e. “100”, and not 100)
- beneficiaries: if included (see defaults note below), “relationshipType” must be an existing Relationship Type on the system

Optional fields with their defaults if not present:

- accountNumber: a system assigned number starting with the prefix set up in the system configuration. If specified in the call, any alpha numeric characters may be used. An error will be returned if a contract exists already exists on the system with the same accountNumber
- clientGroup: set to the system default
- advisor: An existing advisor with the code = advisorCode, and brokerage code = brokerageCode will be set on the contract
    - "advisorMandate" tag can only be used if the API user has Compliance Override permission
- beneficiaries - default empty list

- contractOwners - there will always be at least one, which will be the entity for which this account is created. The following fields can be used to identify the contract owners: For persons: idNumber or entityUniqueId, for company: registrationNumber or entityUniqueId.
- "Fees" section:
    - "Ongoing fee method" is one of
        - "Proportionately"
        - "From Instrument"
        - "From Instrument Then Proportionately"
    - "Ongoing fee instrument is null if the ongoing-fee method is "Proportionately" else it is as above.
    - “feeSharing”
        - "recipientType” can be either contract/advisor/brokerage/distributor
        - "recipientCode" will be the code depending on the recipientType provided. For example if advisor is provided in recipientType, then recipientCode will be the advisor code.
        - If recipientType is contract then recipientCode has to be a fee portfolio contract number
    - "feeRates"
        - feeType (as defined under System>>Fee types -Transaction label)
        - rate (this can either be defined as amount (Money), percentage or sliding scale.
          1. Amount contains 3 fields - e.g.

          "type": "Money",

          "value": "85.11",

          currency": "ZAR"

          2. Percentage contains 2 fields - e.g.

          "type": "Percentage",

          "value": "0.15"

          3. Sliding scale contains 2 fields - e.g.

          "type": "Sliding Scale",

          "name": "Admin fee scale".

          Note - Sliding scales are defined under System>>Sliding scales.

        - "isOverride": true

          Note: If "isOverride" is set to false, the overridden rate on the contract will be removed, and the rate will default to the value set on the contract.

Example JSON response:

```
{"accountNumber" : "CUSTOM1004"}
```

Error examples:

HTTP response	Code	Description
400	PersonNotFound	Could not find person
400	CompanyNotFound	Could not find company
400	MissingEntityIdentifierField	One of the following fields is required: "idNumber" or "registrationNumber" or "entityUniqueId"
400	MalformedRequest	Malformed request. Field "idNumber" is compulsory
400	MalformedRequest	Malformed request. Invalid JSON
400	MalformedRequest	Advisor not linked to brokerage with given brokerage code
404	AdvisorNotFound	Could not find an advisor with the given code
400	InvalidJSONError	Field \"value\" is expected to be a string

6 Method of payment

GET: /api/accounts/methodOfPayment

This should be used to determine the method of payment that can be used in order to create an investment. The methods of payment are static for a system (setup at system setup time under the menu Banking/Payment methods); and the names may differ from system to system. The call is only necessary once, if you then hardcode the names in your calling application.

Example response:

{"data":

[

{"isElectronicTransfer": false, "isDeposit": true, "Name": "Cheque Deposit", "isElectronicCollection": false},

{"isElectronicTransfer": false, "isDeposit": false, "Name": "Electronic/Internet Transfer", "isElectronicCollection": false},

{"isElectronicTransfer": true, "isDeposit": false, "Name": "Electronic Collection by the Administrator", "isElectronicCollection": true}

]}

7 Create investment

POST: /api/accounts/{accountNumber}/investments

Example JSON request: /api/accounts/CUSTOM1004/investments

```
{
        "type" :  "Once Off",
        "amount" : 1234.56,
        "currency" : "ZAR",
        "sourceOfFunds" : "Salary",
        "authoriseAutomatically" : true,
        “status” : “New”,
        “amortisation” : {
                “shouldAmortiseInitialFees”:true,
                “amortisationPeriod”:365
        },
        "purpose" : "Investment",
        "buys" : {
                "INSTRMT1" : "75.00%",
                "INSTRMT2" : "25.00%"
        },
        "methodOfPayment" : "Electronic Collection by the Administrator"
}
OR
{
                "frequency" : "bi-annually",
                "dayOfMonth" : 15,
                "amount" : 1234.56,
                "sourceOfFunds" : "Salary",
                "currency" : "ZAR",
                "type" : "Recurring",
                “commencementDate” : “2024/08/05”,
                “escalationRate” : 10,
                “nextEscalationDate” : “2025/08/05”,
                "bankDetails": {
                "accountNumber": "1234567890"
        }
}
```

Notes:

type - can be one of 'Once Off' or ‘Recurring’; defaults to 'Once Off' , if not specified

- If type = ‘Recurring’, the following fields can also be specified:
    - frequency = one of 'monthly' 'quarterly' 'bi-annually' 'annually' ; defaults to monthly if not specified
    - dayOfMonth (integer) = eg. 15; should be 0 for the last day of the month, defaults to 1 (first of the month) if not specified

- bankDetails = existing bank details can be specified to be linked to a recurring investment. Refer to the "bankDetails" example above.
- Also, the contract needs to have a valid debit order account, otherwise the error:

  "description": "Please select a debit order bank account", "code": "CantSubmitInstruction" will be received

For the recurring type, the method of payment defaults to Debit Order.

For the once-off type, the method of payment can be ‘Electronic Collection by the Administrator’ or ‘Electronic/Internet Transfer’. It defaults to ‘Electronic/Internet Transfer’.

Currencies with identifiers are configured in the system configuration.

"sourceOfFunds" is optional and is also configured in the system configuration. If a value other than the available source of funds types on the configuration is provided, the source of funds on the instruction will default to “other”.

"authoriseAutomatically" is optional. By default the instruction will be submitted, however, an investment cannot be authorised unless it is matched. In order to streamline the process of authorising investments this field can be set to true, in which case the investment will be authorised automatically when it has been matched. Leaving this field out, or setting it to false, will leave the investment in the "submitted" state, where it will remain until the investment is manually authorised.

“status” can be provided as “New” or “Submitted” and will be submitted by default. Status can be provided together with "authoriseAutomatically".

"buys" is also optional. This allows specification of which instruments to create buys for. The instrument as specified by the instrument code must also be available on the product of the investor account. The percentage for each buy must add up to 100%, and can be in string or number format. As the "buys" field is optional, if no instrument is provided in the payload, the default options will be applicable. Where the investor has existing holdings, the investment will be allocated to the first fund of the investor’s holdings. Where the investor does not have existing holdings, the instrument will default to the first available instrument on the product.

"purpose" is optional and must use a purpose name returned in /api/instructions/purposes where supporteInstruction contains Investment

“commencementDate” is optional and defaults to the current date if not set, or if “escalationRate” has been included without the “nextEscalationDate” on a recurring investment instruction

“escalationRate” is only applicable when “type” : “recurring” and “nextEscalationDate” is completed. If “nextEscalationDate” is not completed, the escalation rate will not be set on the recurring instrument instruction and the commencement date will default to the current date.

“nextEscalationDate” is required when “escalationRate” is used on a recurring investment instruction.

“amortisationPeriod” is required when “shouldAmortiseInitialFees” is set to true.

JSON response:

HTTP: 201

{"Unique Id" : 12345678}

Error examples:

HTTP response	Code	Description
400	AccountInactiveError	Account is inactive
404	AccountNotFound	Could not find account
400	MalformedRequest	Malformed request. Amount value "-100.00" must be greater than 0
400	MalformedRequest	Malformed request. Invalid currency "LIBRE"
400	MalformedRequest	Malformed request. Field "amount" is compulsory
400	InstrumentNotAvailable	Instrument with code: "INSTRMT3" could not be found.
400	InstrumentNotAvailable	Instrument with code: "INSTRMT4" is not available for Investor Account: "accountNumber"
400	CantSubmitInstruction	The instrument allocations do not add up to 100.00% for the R 1,234.56 investment on "accountNumber"
400	InvalidInstruction Purpose	The instruction purpose is invalid for the instruction type or the instruction purpose could not be found
400	MalformedRequest	Malformed request. Unknown instruction type (if not one of 'Once Off' or ‘Recurring’)
400	CantSubmitInstruction	Please select a debit order bank account (eg. method of payment is ‘Electronic Collection by the Administrator’ and the bank account submitted is not a debit order bank account on the contract)
400	BankAccountNotFound	Bank account not found on investor

8 Update recurring investment

POST: /api/accounts/updateInvestment/{uniqueInvestmentId}

You need to use the unique investment id returned by either of these calls

GET: /api/accounts/{accountNumber}/instructions

GET: /api/accounts/{accountNumber}/instructions/active

For an investment that is in the "Ready to Recur (auto recur)" status:

Example JSON request: /api/accounts/updateInvestment/221159168

```
{
        "frequency" : "bi-annually",
        "dayOfMonth" : 15,
        "amount" : 10000,
        "authoriseAutomatically" : true,
        "currency" : "ZAR",
        "bankDetails": {
        "accountNumber": "1234567890"
        }
}
```

1. It removes all the current buys if they exist. And then has an optional "buys" field (Look at the POST: /api/accounts/{accountNumber}/investments details for exactly how it handles the buys field when it exists and when it doesn't exist).
2. "sourceOfFunds", "purpose", and "methodOfPayment" are all additional optional fields.
3. It then tries to submit the instruction while resuming warnings.
4. It then updates the "authoriseAutomatically" field if present

Notes:
○ For additional commentary on fields and errors, please look at Create Investment.
○ You are also able to update the bank details linked to a recurring investment with existing bank details available to the investor. New bank details cannot be created via this call.

HTTP response	Code	Description
404	NotReadyToRecur	Investment not ready to recur
400	BankAccountNotFound	Bank account not found on investor

JSON response:

HTTP: 201

{"Unique Id" : 221159168}

(same as what was passed in, if update is successful)

9 Create withdrawal

POST: /api/accounts/{accountNumber}/withdrawals

Here are some important fields that determine how the withdrawal instruction will sell.

1. "workIn", which can be "amount" or "units". This is optional. If not present, the default is "amount". If "workIn" is "units", this will imply "Units as %", which means when specifying "sells", we expect percentages per instrument.

2. "captureBy", which can be "entire instruction" or "per instrument". This is also optional.

3. "amount", which is a positive currency amount. This is used when working in amounts, capturing by entire instruction and no sells are specified.

4. "instructionPercentage", which is a number > 0 and <= 100, and only used when working in units, capturing by entire instruction and no sells are specified.

5. "status", which can be "Authorised", "Pending Authorisation", “New” or "Submitted". Instruction may only be submitted as “Authorised” if the API user has ‘may authorize own instruction’ permission. If this field is not set then the withdrawal will not process. If not provided, instruction will be created with status Submitted.

6. “type” - can be one of “Once Off” or “Recurring”; defaults to “Once Off” , if not specified

7. “commencementDate” - sets first recurrence date, subsequent recurrences will take place on the 1st if “type” is “Recurring”. If not provided, this will default to current date.

8. “clientAccountUniqueId” - specify the client account you would like to use for the withdrawal. Please note that pot restrictions will apply to the withdrawal.

Summary of default behaviour

* If there is an "amount" and "sells", the default is "per instrument".

* If there is an "amount", and no "sells" (this field is not present), it is "entire instruction".

* If there is no "amount", "sells" are compulsory, the default is "per instrument".

Selling behaviour options

This section shows how to specify what the withdrawal should sell.

Proportional amount sell over all instruments

You post an "amount" field, and no "sells". The system will create a withdrawal that will proportion the amount over available instruments. If there is only one instrument available, the system will sell the entire amount from that instrument. It will pay the specified amount to the investor.

Example JSON request:

```
{
        "workIn" : "amount",    # Optional, but clarifies that this is the default
        "captureBy" : "entire instruction",   # Optional, but clarifies that this is the default
        "amount" : 1234.56,
        "currency" : "ZAR",
        "status" : "Authorised",
        "bankDetails" : {"accountNumber" : "1234567890"}
}
```

Proportional percentage sell over all instruments

You post the "instructionPercentage" field, and no "sells". The system will create a withdrawal to the total value of the provided percentage, over all available instruments. It will pay the proceeds received from the instrument dealers to the investor.

Example JSON request:

```
{
        "workIn" : "units",    # Compulsory for this type of withdrawal
        "captureBy" : "entire instruction",   # Compulsory for this type of withdrawal.
        "instructionPercentage" : 40,    # Compulsory for this type of withdrawal.
        "bankDetails" : {"accountNumber" : "1234567890"}
}
```

Sell per instrument by amount

You provide the amounts per instrument in the "sells" object. The system will create a withdrawal that sells each instrument by the specified amount for each sell. It will pay the total of all the amounts in the sells object to the investor.

Example JSON request:

```
{
        "workIn" : "amount",    # Optional, but clarifies the default.
        "captureBy" : "per instrument",   # Optional, but clarifies the default.
        "amount" : 1234.56,   # Optional. But validated to be = to the sum of amounts on the sells
        "bankDetails" : {"accountNumber" : "1234567890"},
        "sells" : {
                "INSTRMT1" : 234.56,
                "INSTRMT2" : 1000.00
                },  # Compulsory per instrument. Format: {"Instrument code", amount value}
}
```

Sell per instrument by percentage (work-in = units)

You provide percentage values in the "sells" object. The system will create a withdrawal that sells the specified percentage of available units in each instrument. It will pay the sum of the proceeds for each sell received from the instrument dealers to the investor.

Example JSON request:

```
{
        "workIn" : "units",    # Compulsory
        "captureBy" : "per instrument",   # Compulsory
        "bankDetails" : {"accountNumber" : "1234567890"},
        "sells" : {
                "INSTRMT1" : 100.0,
                "INSTRMT2" : 50.0
                },  # Compulsory per instrument. Format: {"Instrument code", amount value}
}
```

Proportional percentage sell over all instruments **(recurring withdrawal)**

```
        {
        "workIn" : "amount",
        "captureBy" : "entire instruction",
        "amount" : 5000,
                "currency" : "ZAR",
        "status" : "Submitted",
        "bankDetails" : {"accountNumber" : "1234567890"},
        “type” : “Recurring”,
        "commencementDate" : "2025/09/03"
        }
```

The bank account

For creating a withdrawal, we have 2 options: use an existing bank account or create a new bank account.

The system will check for the existence of a bank account matching the "accountNumber" on the contract owner. If none is found, the system will check for a third party bank account matching the "accountNumber". If found and the API user does not have the permission "Third party withdrawal", an error is raised. If a matching bank account is found, this bank account is used and no further "bankAccount" fields are used. This means that if the user of the API intends to use

an existing bank account, only the "accountNumber" has to be provided and all the other fields become optional. Some examples of this are in the examples above.

Please use the Entities API to update bank accounts.

To create a new bank account with a withdrawal, the example below shows what fields are available.

Example JSON showing optional fields that can be specified when creating a new bank account.

Example Partial JSON request:

```
"bankDetails" :
        {
        "holderName" : "John Doe",
        "accountNumber" : "1234567890",
        "branchCode" : "222626",
        "type" : "Current",
        "bankName" : "First National Bank",
        "branchName" : "Centurion",   # Optional
        "currencyIdentifier" : "zar",   # Optional, defaults to preferred currency
        "swiftBankIdentifierCode" : "777",    # Optional
        "intermediaryName" : "NEDWXYZYYYY",   # Optional
        "internationalBankAccountNumber" : "GB82WEST12345698765432"    # Optional
        }
```

Currency

The currency of a withdrawal is the same as the currency of the bank account (that will be paid to) on a withdrawal. All amount fields are interpreted in this currency. The system currently does not support a currency on a withdrawal that is different from the currency of a bank account. The "currencyIdentifier" in the "bankDetails" field has to be provided if the denominated currency of the bank account being paid to determines the currency of the withdrawal.

If the denominated currency of the bank account being paid to is not correct on the system, the payment instruction sent to the payment agent (or the bank directly) will be received in the contract preferred currency. This simply implies that the bank will do foreign exchange outside of the system.

Notes:

Possible account types: "**Current**", "**Transmission**" or "**Savings**".

Possible bank names: as configured under FinWorks system configuration -> Banks.

Currency identifier as configured under FinWorks system configuration.

"status" is optional. By default the instruction will be submitted, and will be attempted to be authorised in the case shown in the example. "Authorised" and "Submitted" are the only accepted input; specifying "Submitted" will do nothing as it is submitted by default.

"purpose" is optional and must use a purpose name returned in /api/instructions/purposes where supportedInstruction contains Withdrawal

The value for each sell depends on presence of other fields as follows:

Field	Specified	Sells value meaning
workIn	no	amount will be assumed.
workIn	yes, with "captureBy" : "entire instruction"	“sells” must not be specified. Applies “amount” or “instructionPercentage” to all instruments.
workIn	yes, with "captureBy" : "per instrument"	“sells” must be specified. The value provided per instrument in the “sells” object will mean the following: if "workIn" : "amount" then it’s a currency value. Else if “workIn” : “units” then a percentage is provided per instrument.

JSON response:

HTTP: 201

{"Unique Id" : 12345678}

Error examples:

HTTP response	Code	Description
400	AccountInactiveError	Account is inactive
404	AccountNotFound	Could not find account
400	MalformedRequest	Malformed request. Amount value "-100.00" must be greater than 0
400	MalformedRequest	Malformed request. Invalid currency "LIBRE"
400	MalformedRequest	Malformed request. Field "amount" is compulsory
400	MalformedRequest	Malformed request. Field "bankDetails" is compulsory
400	MalformedRequest	Malformed request. Bank does not exist "Invalid bank"
400	MalformedRequest	Malformed request. Field "accountNumber" is compulsory
400	AvailableAmount Exceeded	There is not enough funds available to sell the requested amount
400	InvalidStatus	"AnInvalidStatus" is not a valid instruction status
400	InstrumentNotAvailable	Instrument with code: "INSTRMT3" could not be found.
400	InstrumentNotAvailable	Instrument with code: "INSTRMT4" is not available for Investor Account: "accountNumber"
400	NoAvailableHoldings	There are no available holdings in instrument: "INSTRMT1"
400	AvailableAmount Exceeded	There are not enough funds available in instrument: "INSTRMT1" to sell the requested amount
400	InvalidSellPercentage	The value on an instrument sell is not a valid percentage. Must be between 0 and 100 inclusive.

400	InvalidAmountComparisonWithSells	The total sell amount of all instruments does not match the optional amount specfiied

10 Individual statement

GET:

/api/accounts/{accountNumber}/statement?start_date={startDate}&end_date={endDate}&sections={list-of-section-names}&add_password={boolean}'

Content-Type: application/pdf

Example request:

/api/accounts/ACC123/statement?start_date=2005-01-01&end_date=2005-01-31

JSON response:

HTTP: 200

Content-Type: application/pdf

Notes:

Response contains pdf binary data.

All fields are mandatory except for 'sections' and 'add_password'

sections = comma separated list of section names

URLs must be *percent encoded*, eg. space = %20, comma = %2C

add_password = ‘true’ or ‘false’

if add_password is not specified, a password will be added to the document based on the value of the "Password protect PDF documents?" configuration setting.

Error examples:

HTTP response	Code	Description
400	Malformed request	Invalid date
500	MultipleAccountsFound	Found more than one matching account
400	AccountInactiveError	Account is inactive
404	AccountNotFound	Could not find account

Available sections:

- Account details
- Advisor details
- Advisor fee details
- Beneficiary details
- Investment Summary For Statement Period
- Investment Allocation
- Asset Allocation
- Default investment portfolio
- Investment summary since inception
- Investment Summary Graph
- Investment return
- Investment return graph
- Monthly market values
- Instrument performance
- Instrument cash flows
- Instrument cash flow summary
- Combined cash instrument transactions per currency
- Account holdings summary
- Account holdings summary (opening balance)
- Insurance instrument benefits
- Benefit description
- Currency exposure
- Instructions
- Cash flows
- Transaction history
- Total expense ratio
- Administration fee rates per year
- Fees
- Statement notes
- Income Review
- Bond Maturity
- Statement footer
- Statement cover letter
- Net Replacement Ratio

11 Combined statement

Multiple accounts statement for a particular account owner.

GET:
/api/accounts/statement?search_key={searchKey}&start_date={startDate}&end_date={endDate}&sections={list-of-section-names}&add_password={boolean}

Content-Type: application/pdf

Notes:

Response contains pdf binary data.

All fields are mandatory except for 'sections' and ‘add_password’

search_key = company registration number or person identity number.

sections = comma separated list of section names

URLs must be percent encoded, eg. space = %20, comma = %2C

add_password = 'true' or 'false'

if add_password is not specified, a password will be added to the document based on the value of the "Password protect PDF documents?" configuration setting.

Example request:

/api/accounts/statement?search_key=123456789&start_date=2005-01-01&end_date=2005-01-31&sections=Account%20details%2CBeneficiary%20details

JSON response:

HTTP: 200

Content-Type: application/pdf

Error examples:

HTTP response	Code	Description
400	Malformed request	Invalid date
403	Security	
404	AccountOwnerNotFound	Could not find account owner
404	AccountNotFound	Could not find account (** if there are no active accounts for the account owner)
500	MultipleAccountOwnersFound	Found more than one matching account owner

Available sections:

- Account details
- Advisor details
- Advisor fee details
- Beneficiary details
- Investment Summary For Statement Period
- Investment Allocation
- Asset Allocation
- Default investment portfolio
- Investment summary since inception
- Investment Summary Graph
- Investment return
- Investment return graph
- Monthly market values
- Instrument performance
- Instrument cash flows
- Instrument cash flow summary
- Combined cash instrument transactions per currency
- Account holdings summary
- Account holdings summary (opening balance)
- Insurance instrument benefits
- Benefit description
- Currency exposure
- Instructions
- Cash flows
- Transaction history
- Total expense ratio
- Administration fee rates per year
- Fees
- Statement notes
- Income Review
- Bond Maturity
- Statement footer
- Statement cover letter
- Net Replacement Ratio
- Combined holdings summary
- Combined holdings summary graph
- Combined Asset Allocation

12 Holdings

GET: /api/accounts/{accountNumber}/holdings (as of today)

GET: /api/accounts/{accountNumber}/holdings?date={date} (as of date specified)

Note: Date must be ISO format string YYYY-MM-DD

Example JSON response:


```
[ {
        "Contract id": 123123123,
        "Contract number": "POL3",
        "Client account id": 48044772352,
        "Instrument id": 1122334455,
        "Instrument grouping": “Public”,
        "Instrument code": "COR01",
        "Instrument name": "Coronation Balanced Fund",
        "Instrument account number": "0987654321",
        "Price date": "2022-02-01",
        "Date": "2022-02-01",
        "Units": {
                "type": "Unit",
                "Instrument id": 1122334455,
                "value": "440"
        },
        "Market Value in System Currency": {
                "type": "Money",
                "value": "440",
                "currency": "ZAR
        },
        "Market Value in Fund Currency": {
                "type": "Money",
                "value": "440",
                "currency": "ZAR
        },
        "Accrued fees": {
                "type": "Money",
                "value": "0.60",
                "currency": "ZAR
        },
        "Accrued interest": {
                "type": "Money",
                "value": "0.00",
                "currency": "ZAR
        },
        "Latest available price": {
                "type": "Price",
                "Instrument id": 1122334455,
                "value": "0.60",
```

```
                "currency": "ZAR
        }
} ]
```


Error examples:

HTTP response	Code	Description
400	AccountInactiveError	Account is inactive
404	AccountNotFound	Could not find account

13 Available Holdings

GET: /api/accounts/{accountNumber}/availableHoldings (as of today)

GET: /api/accounts/{accountNumber}/availableHoldings?date={date} (as of date specified)

Note: Date must be ISO format string YYYY-MM-DD

Example JSON response:


```
[ {
        "Contract id": 123123123,
        "Contract number": "POL3",
        "Client account id": 48044772352,
        "Instrument id": 1122334455,
        "Instrument grouping": “Public”,
        "Instrument code": "COR01",
        "Instrument name": "Coronation Balanced Fund",
        "Instrument account number": "0987654321",
        "Price date": "2022-02-01",
        "Date": "2022-02-01",
        "Units": {
                "type": "Unit",
                "Instrument id": 1122334455,
                "value": "440"
        },
        "Market Value in System Currency": {
                "type": "Money",
                "value": "440",
                "currency": "ZAR
        },
        "Market Value in Fund Currency": {
                "type": "Money",
```

```
                "value": "440",
                "currency": "ZAR
        },
        "Accrued fees": {
                "type": "Money",
                "value": "0.60",
                "currency": "ZAR
        },
        "Latest available price": {
                "type": "Price",
                "Instrument id": 1122334455,
                "value": "0.60",
                "currency": "ZAR
        }
} ]
```


Error examples:

HTTP response	Code	Description
400	AccountInactiveError	Account is inactive
404	AccountNotFound	Could not find account

14 Advisor accrued fees

GET:
/api/accounts/advisorAccruedFees?code={code}&start_date={startDate}&end_date={endDate}

Example JSON response:


```
{
	"Instrument code" : "ZAR",
	"Investor name" : "ivan l",
	"Fee type" : "Annual adviser fee",
	"Amount in system currency" : {
		"currency" : "ZAR",
		"value" : "10.49",
		"type" : "Money"
	},
	"Amount in fee currency" : {
		"currency" : "ZAR",
		"value" : "10.49",
		"type" : "Money"
	},
	"Adviser code" : "AGFSPB",
	"Instrument name" : "Cash in ZAR",
	"Adviser name" : "Pompies P",
	"Transaction date" : "2005/06/12",
	"Account number" : "POL3",
	"Product" : "Discretionary",
	"Preferred currency" : "ZAR",
	"Fee currency" : "ZAR"
}
```


Error examples:

HTTP response	Code	Description
404	AdvisorNotFound	Invalid advisor code

15 Advisor fees (taken)

GET:
/api/accounts/advisorFees?code={code}&start_date={startDate}&end_date={endDate}

Example JSON response:


```
{
        "Instrument Code" : "ZAR",
        "Investor Name" : "ivan, I",
        "Investor ID" : "6611235090089",
        "Units" : {
                "currency" : "ZAR",
                 "value" : "10.49",
                 "type" : "Money"
        }.
        "Adviser code" : "AGFSPB",
        "Adviser name" : "Pompies P",
        "Transaction Date" : "2005-06-12",
        "Adviser Fee Period" : "2005/05/13 to 2005/06/12",
        "Amount" : {
                "currency" : "ZAR",
                "value" : "10.49",
                "type" : "Money"
        },
        "Product" : "Discretionary",
        "Instrument" : "Cash in ZAR",
        "Fee Type" : "Ongoing Adviser Fee",
        "Account Number" : "POL3",
        "Price" : {
                "value" : "1.0000",
                "currency" : "ZAR",
                "Instrument id" : <cash instrument id>,
                "type" : "Price"
        }
}
```


Error examples:

HTTP response	Code	Description
404	AdvisorNotFound	Invalid advisor code

16 Policy document

GET: /api/accounts/{accountNumber}/policyDocument?add_password={boolean}'

Content-Type: application/pdf

Example request:

/api/accounts/ACC123/policyDocument

/api/accounts/ACC123/policyDocument?add_password=true

Note: if add_password is not specified, a password will be added to the document based on the value of the "Password protect PDF documents?" configuration setting.

Notes:

Response contains pdf binary data.

All fields are mandatory except for ‘add_password’

add_password = 'true' or 'false'

if add_password is not specified, a password will be added to the document based on the value of the "Password protect PDF documents?" configuration setting.

JSON response:

HTTP: 200

Content-Type: application/pdf

Response contains pdf binary data.

Error examples:

HTTP response	Code	Description
500	MultipleAccountsFound	Found more than one matching account
404	AccountNotFound	Could not find account

17 Account details

GET: /api/accounts/{accountNumber}

Example request:

/api/accounts/ACC123

Example JSON response:


```
{
        "Unique Id": 123456789014,
        "Product": "Discretionary",
        "Account Number": "ACC123",
        "Contract Number": "ACC123",
        "Policy Number":"POL1004",
        "Model Portfolio Name": "Clever Conservative",
        "Owners": [
                {
                        "Entity": "/api/entities/123456789015",
                        "Entity Id": 123456789015,
                        "Name": "Mr Gene Fraser"
                }
        ],
        "Advisor": {
                "Advisor Entity Unique Id": 378231296,
                "Advisor Name": "John Clever",
                "Advisor Code": "ADV001",
                "Brokerage Entity Unique Id": 275446528,
                "Brokerage Code": "CLE001",
                "Brokerage Name": "Clever Advice"
        },
        "Advisor Mandate": {
                "switch": true,
                "changeDetails": true,
                "invest": true,
                "withdraw": true
        },
        "Beneficiaries": [
                {
                        "Entity": "/api/entities/123456789017",
                        "Entity Id": 123456789017,
                        "Name": "Mr John Seymor",
                        "Benefit": {
                                "type": "Percentage",
                                "value": "100.00"
                        }
                }
        ],
        "LivesAssured": [
                {
```

```
                "Entity Id": 123456789017,
                "Name": "Mr John Seymor"
        }
],
"Product Details": {
        "Product Name": "Itransact Securities Investment Plan",
        "Product id": 160275969,
        "Description": "Itransact Securities Investment Plan",
        "Status": {
                "active": "true",
                "identifier": "Existing business"
        }
},
"Communication Options": {
        "communicationPreference": "None",
        "transactionNotificationPreferences": {
                "CompleteSwitchInstruction": "None",
                "SubmitUnitTransferOutInstruction": "None",
                "SubmitReinvestmentSwitchInstruction": "None",
                "SubmitSwitchInstruction": "None",
                "SubmitUnitTransferInInstruction": "None",
                "SubmitWithdrawal": "None",
                "CompleteReinvestmentSwitchInstruction": "None",
                "CompleteWithdrawal": "None",
                "CompleteUnitTransferOutInstruction": "Both",
                "ApprovedChangeRequest": "Email",
                "SuccessfulLogin": "Sms",
                "SubmitInvestment": "None",
                "CompleteInvestment": "None",
                "CompleteUnitTransferInInstruction": "None",
                "SendAutomaticStatement": "Email"
        },
        "automaticallyEmailStatement": false
},
"Fees": {
        "Ongoing fee method": "From Instrument",
        "Ongoing fee instrument": {
                "Instrument id": 23456789,
                "Instrument code": "ZAR",
                "Instrument name": "Cash in ZAR"
        },
        "feeRates": [
                {
                        "feeType": "Administration Fee",
                        "rate": {
                                "type": "Percentage",
                                "value": "0.15"
                        },
                        "isOverride": true
                }
```

```
            ],
"Fee sharing": [
    {
        "feeType": "Administration Fee",
        "sharing": [
            {
                "percentage": {
                    "type": "Percentage",
                    "value": "50.00"
                },
                "contractNumber": "ccc"
            },
            {
                "percentage": {
                    "type": "Percentage",
                    "value": "50.00"
                },
                "contractNumber": "ddd"
            }
        ]
    },
    "Market value": {
            "currency": "ZAR",
            "amount": "77444.0600",
            "accrued fees": "27.9200",
            "date": "2023-06-29"
    },
    "Representative": {
            "Entity": "/api/entities/123456789018",
            "Name": "Ms Sarah Ann Fraser"
    },
    "Client Accounts": [
            {
                    "Unique Id": 12345,
                    "Name": "Default",
                    "Model Portfolio Id": null,
                    "Model Portfolio Name": ""
            },
            {
                    "Unique Id": 34567,
                    "Name": "Savings - ABC Conservative",
                    "Model Portfolio Id": 45678,
                    "Model Portfolio Name": "ABC Conservative"
            }
    ],

    "Cash Provision": {
            "reason": "Minimum liquidity required for monthly withdrawals",
            "percentage": {
                    "value": "10.00",
```

```
                        "type": "Percentage"
                },
                "isOverridden": true
        },
        "Low Cash Provision": {
                "isOverridden": true,
                "reason": "Set to maintain minimum trading buffer",
                "percentage": {
                        "type": "Percentage",
                        "value": "20"
                }
        }

}
```


Notes:

In the "Fees" section:

- "Ongoing fee method" is one of
  - "Proportionately"
  - "From Instrument"
  - "From Instrument Then Proportionately"
- "Ongoing fee instrument is null if the ongoing-fee method is "Proportionately".
- Beneficiary details returned contains the Client Number of the entity, visible in Person Details card -> View Details.
- Cash Provision and Low Cash Provision will only be included if it is overridden on a contract. The defaults on the product applies if there are no overrides

Error examples:

HTTP response	Code	Description
404	AccountNotFound	Could not find account

18 Update Account details

POST: /api/accounts/{accountNumber}

Example request:


```
{
"contractStatus": "New business",
"distributionOption": "reinvestDistribution",
"advisor": {
        "advisorCode": "AGFSPB",
        "brokerageCode": "000017",
        "advisorMandate": {
                "changeDetails": false,
                "invest": false,
                "switch": false,
                "withdraw": false
            }
        },
"communicationOptions": {
        "communicationPreference": "No consent given",
        "automaticallyEmailStatement": false,
         "transactionNotificationPreferences": {
                "SuccessfulLogin": "SMS",
                "SubmitInvestment": "Email"
                }
        },
"contractOwners": [
                {
                    "idNumber": "1234567890"
```

```
                },
                {
                        "idNumber": "1234567891"

                }
        ],

"beneficiaries": [
                {
                        "idNumber": "1234567878",
                        "benefit": {
                                "type": "Percentage",
                                "value": "40.0"
                        }
                },
                {
                        "idNumber": "1234567879",
                        "benefit": {
                                "type": "Percentage",
                                "value": "40.0"
                        }
                }
        ]

}
```


With this call one can update the following:

- distributionOption : can be one of: reinvestDistribution,payoutDistribution or leaveDistributionInCash

- contract status: One of the statuses in System settings/ Account statuses. The transition from the existing status to the new status should also be valid
- advisor
    - If the advisor tag is provided, then advisorCode is mandatory
    - If you want to remove/unlink an advisor from a contract, then pass the advisor tag as follows: `"advisor": { "advisorCode": null}`
    - "advisorMandate" tag can only be used if the API user has Compliance Override permission
- communication options:

    communicationPreference: valid options: Post, Email, No consent given, Opted-out
- Fees - see Create account
- Beneficiaries - If the call contains the ‘beneficiaries’ key, all the beneficiaries details should be provided, and the “value” field should be submitted as a “number” (i.e. “100”, and not 100). The beneficiaries will be updated accordingly. The percentages should add up to 100%. ID number is the actual ID number of the person.
- livesAssured
    - See create account for detail on the format of this array.
    - If the existing life assured is required to remain in place, and an additional life assured must be added, both the existing and new lives assured must be included in the call. If an existing life assured is not included in the call, that life assured will be removed from the contract.

19 Available instruments

GET: /api/accounts/{accountNumber}/instruments

Example request:

/api/accounts/ACC123/instruments

Example JSON response:


```
{
        "data": [
                {
                        "Instrument id": 390850560,
                        "Instrument provider unique id": null,
                        "Code": "ZAR",
                        "Name": "Cash in ZAR",
                        "Instrument grouping": "Public",
                        "Instrument provider": ""
                },
                {
                        "Instrument id": 517341184,
                        "Instrument provider unique id": null,
                        "Code": "COREQ",
                        "Name": "Coronation Equity Fund",
                        "Instrument grouping": "Public",
                        "Instrument provider": "Coronation"
                }
        ],
        "size": 2
}
```


Error examples:

HTTP response	Code	Description
404	AccountNotFound	Could not find account

20 Documents

GET: /api/accounts/{accountNumber}/documents

Example request:

/api/accounts/ACC123/documents

Example JSON response:


```
{
        "data": [
                {
                        "Document id":          "/api/documents/825280512",
                        "Document Unique Id"   825280512,
                        "Type":                 "Contribution Certificate",
                        "Label":                "a document",
                        "Date":                 "2021/08/11"
                },
                {
                        "Document id":          "/api/documents/214897664",
                        "Document Unique Id"   214897664,
                        "Type":                 "Application Form",
                        "Label":                "another document",
                        "Date":                 "2021/08/11"
                }
        ],
        "size": 2
}
```


Error examples:

HTTP response	Code	Description
404	AccountNotFound	Could not find account

21 Instructions

GET: /api/accounts/{accountNumber}/instructions

For active instructions only:

GET: /api/accounts/{accountNumber}/instructions/active

Example request:

/api/accounts/ACC123/instructions

Example JSON response:


```
{
        "data": [
                {
                        "Instruction id":           "/api/instructions/516098816",
                        "Instruction Unique Id":  516098816,
                        "Type":                        "Once Off Investment",
                        "Status":                      "Submitted",
                        "Amount":                    "R xxx",
                        "Contributions New": [ //will replace the current Contributions array
                                {
                                   "Instrument code": "FNBEQF",
                                   "Price": {
                                      "type": "Price",
                                      "currency": "ZAR",
                                      "Instrument id": xxx,
                                      "value": "108.46"
                                   },
                                   "Units": {
                                      "type": "Unit",
                                      "Instrument id": xxx,
                                      "value": "1"
                                   },
                                   "Instrument id": xxx,
                                   "Instrument name": "FNB GLOBAL 1200 EQUITY FoF ETF",
                                   "Type": "Buy",
                                   "Amount": {
                                      "type": "Money",
                                      "currency": "ZAR",
                                      "value": "xxx"
                                   },
                                   "Percentage": "100.00%",
                                   "Status": "Allocated"
                                }
                        ]
                        "Contributions": [//to be deprecated
                                {
                                        "Type":                    "Buy",
```

```
                                        "Sell Instrument":      "",
                                        "Buy Instrument":       "aFundCode",
                                        "Amount":               "R 2,100.00",
                                        "Price":                "2.1000",
                                        "Units":                "1000.00",
                                        "Status":               "Active"
                                }
                                ]

                },
                {
                        "Instruction id":           "/api/instructions/418818304",
                        "Instruction Unique Id":    418818304,
                        "Type":                     "Once Off Withdrawal",
                        "Status":                   "New",
                        "Amount":                   "R 0.00",
                        "Contributions": [ //to be deprecated
                                {
                                        "Type":                 "Sell",
                                        "Sell Instrument":      "aFundCode",
                                        "Buy Instrument":       "",
                                        "Amount":               "0.00",
                                        "Price":                "2.1000",
                                        "Units":                "0.00",
                                        "Status":               "Active"
                                }
                                ]
                }
        ],
         "size": 2
}
```


Both entry points will return an investment in ready to recur status.

Such an investment can be rejected, updated (eg. frequency, amount and contributions) and re-submitted

Example response for Recurring investment, in Ready to recur status:

{"data": [

{"Instruction Unique Id": 221159168,

"Instruction id": "/api/instructions/221159168",

"Type": "Recurring Investment",

"Status": "Ready to Recur (auto recur)",

"Amount": "R 2,500.00",

"Contributions": [{"Type": "Buy", "Sell Instrument": "", "Buy Instrument": "COR01", "Amount": "R 2,500.00", "Price": "1.0000", "Units": "0.0000", "Status": ""}]

}

], "size": 1}

Error examples:

HTTP response	Code	Description
404	AccountNotFound	Could not find account

22 Investment Summary Since Inception

GET: /api/accounts/{accountNumber}/investmentSummarySinceInception

- as of today

GET: /api/accounts/{accountNumber}/investmentSummarySinceInception?end_date={date}

- as of date specified

Note: Date must be ISO format string YYYY-MM-DD

Example JSON response:


```
{
	"Withdrawals / Transfers Out": {
	"type": "Money",
	"currency": "USD",
	"value": "0.00"
	},
	"End date": "2022-06-01",
	"IRR Cumulative": {
	"type": "Percentage",
	"value": "-25.07"
	},
	"Start date": "2022-01-01",
	"Closing balance": {
	"type": "Money",
	"currency": "USD",
	"value": "231844.20"
	},
	"Opening balance": {
	"type": "Money",
	"currency": "USD",
	"value": "308826.60"
	},
	"Gain/Loss": {
	"type": "Money",
	"currency": "USD",
	"value": "-76982.40"
	},
	"Investments / Transfers In": {
	"type": "Money",
	"currency": "USD",
	"value": "0.00"
	},
	"IRR Annualised": {
	"type": "Percentage",
	"value": "-50.00"
	}
}
```

Error examples:

HTTP response	Code	Description
400	AccountInactiveError	Account is inactive
404	AccountNotFound	Could not find account

23 Investment Summary for Period

GET:
api/accounts/{accountNumber}/investmentSummaryForPeriod?start_date={startDate}&end_date={endDate}

Note: Date must be ISO format string YYYY-MM-DD

Example JSON response:


```
{
        "Withdrawals / Transfers Out": {
        "type": "Money",
        "currency": "USD",
        "value": "0.00"
        },
        "End date": "2022-06-01",
        "IRR Cumulative": {
        "type": "Percentage",
        "value": "-25.07"
        },
        "Start date": "2022-01-01",
        "Closing balance": {
        "type": "Money",
        "currency": "USD",
        "value": "231844.20"
        },
        "Opening balance": {
        "type": "Money",
        "currency": "USD",
        "value": "308826.60"
        },
        "Gain/Loss": {
        "type": "Money",
        "currency": "USD",
        "value": "-76982.40"
        },
        "Investments / Transfers In": {
        "type": "Money",
        "currency": "USD",
        "value": "0.00"
        },
        "IRR Annualised": {
        "type": "Percentage",
        "value": "-50.00"
        }
}
```

Error examples:

HTTP response	Code	Description
400	AccountInactiveError	Account is inactive
404	AccountNotFound	Could not find account

24 Products

GET: /api/accounts/products

Example request:

/api/accounts/products

Example JSON response:


```
{
        "products":
                [
                        {
                        "Product id": 609157888,
                        "Product name": "Discretionary"},
                        {"Product id": 896356096,
                        "Product name": "Retirement annuity"}
                ]
}
```

25 Product available instruments

GET: /api/accounts/products/{uniqueId}/availableInstruments

Example request:

/api/accounts/products/16420686081/availableInstruments

Example JSON response:


```
{
        "data": [
                {
                        "Instrument id": 52565504,
                        "Instrument provider unique id": 11223344,
                        "Code": "COREQ",
                        "Name": "Coronation Equity Fund",
                        "Instrument grouping": "Public",
                        "Instrument provider": "Coronation"
                },
                {
                        "Instrument id": 389797632,
                        "Instrument provider unique id": 55667788,
                        "Code": "ZAR",
                        "Name": "Cash in ZAR",
                        "Instrument grouping": "Public",
                        "Instrument grouping": "Public",
                        "Instrument provider": ""
                }
        ],
        "size": 2
}
```


Error examples:

HTTP response	Code	Description
500	NotFound	Invalid uniqueId

26 Transactions for Period

Transactions for period have 2 forms.

26.1. For a contract:

GET:
api/accounts/{accountNumber}/transactionsForPeriod?start_date={startDate}&end_date={endDate}

Note: Date must be ISO format string YYYY-MM-DD

Example JSON response:


```
{
        "data": [
                {
                        "Description": "\"Buy (Once Off Investment)\"",
                        "Date": "2005-05-13",
                        "Instrument code": "AGMF",
                        “Instrument id”: 123456789,
                        "Transaction id": 375453696,
                        "Instruction id": 987654321,
                        "Instrument name": "Allan Gray Money Market Fund",
                        "Amount": {
                                "type": "Money",
                                "currency": "ZAR",
                                "value": "600.00"},
                        "Balance": {
                                "type": "Money",
                                "Instrument id”: 123456789,
                                "value": "700.00"},
                        "Units": {
                                "type": "Money",
                                "Instrument id”: 123456789,
                                "value": "600.00"},
                        "Gain/Loss": {
                                  "type": "Money",
                                  "currency": "ZAR",
                                  "value": "0.00”}
                },
                {
                        "Description": "\"Sell (Once Off Withdrawal)\"",
                        "Date": "2005-05-16",
                        "Instrument code": "AGMF",
                        "Transaction id": 948439808,
                        "Instruction id": 987654321,
                        "Instrument name": "Allan Gray Money Market Fund",
                        "Amount": {"type": "Money", "currency": "ZAR", "value": "-360.00"},
                        "Balance": {"type": "Money", "Instrument id: 123456789", "value": "240.00"},
                        "Units": {"type": "Money", "Instrument id: 123456789", "value": "-360.00"},
                        “Gain/Loss”: {"type": "Money", "currency": "ZAR", "value": "1234”}
                },
```

```
        {
                "Description": "\"Deposit matched\"",
                "Date": "2005-05-13",
                "Instrument code": "ZAR",
                "Transaction id": 110048000,
                "Instruction id": 987654321,
                "Instrument name": "Cash in ZAR",
                "Amount": {"type": "Money", "currency": "ZAR", "value": "600.00"},
                "Balance": {"type": "Money", "currency": "ZAR", "value": "600.00"},
                "Units": {"type": "Money", "currency": "ZAR", "value": "600.00"},
                “Gain/Loss”: null
        },
        {
                "Description": "\"Buy in Allan Gray Money Market Fund (AGMF)\"",
                "Date": "2005-05-13",
                "Instrument code": "ZAR",
                "Transaction id": 271667968,
                "Instruction id": 987654321,
                "Instrument name": "Cash in ZAR",
                "Amount": {"type": "Money", "currency": "ZAR", "value": "-600.00"},
                "Balance": {"type": "Money", "currency": "ZAR", "value": "0.00"},
                "Units": {"type": "Money", "currency": "ZAR", "value": "-600.00"},
                “Gain/Loss”: null
        },
        {
                "Description": "\"Sell of Allan Gray Money Market Fund\"",
                "Date": "2005-05-16",
                "Instrument code": "ZAR",
                "Transaction id": 732626944,
                "Instruction id": 987654321,
                "Instrument name": "Cash in ZAR",
                "Amount": {"type": "Money", "currency": "ZAR", "value": "60.00"},
                "Balance": {"type": "Money", "currency": "ZAR", "value": "360.00"},
                "Units": {"type": "Money", "currency": "ZAR", "value": "60.00"},
                “Gain/Loss”: null
        },
        {
                "Description": "\"Payment 2-POL3          to ***566 (Capitec-ZAR)\"",
                "Date": "2005-05-21",
                "Instrument code": "ZAR",
                "Transaction id": 601679872,
                "Instruction id": null,
                "Instrument name": "Cash in ZAR",
                "Amount": {"type": "Money", "currency": "ZAR", "value": "-360.00"},
                "Balance": {"type": "Money", "currency": "ZAR", "value": "0.00"},
                "Units": {"type": "Money", "currency": "ZAR", "value": "-360.00"},
                “Gain/Loss”: null
        }
],
 "size": 6
```

}

Notes:

- "Gain/Loss": will return the following:
  - "null" value for transactions on a cash instrument, or
  - {"type" : "Money", "currency" : "ZAR", "value" : "0.00"} where no gain/less is applicable or incurred on a non-cash instrument
  - "Gain/Loss": {"type": "Money", "currency": "ZAR", "value": "1234"}

26.2. For all the contracts the API user has access to

This second type can be paginated.

GET: api/accounts/transactionsForPeriod?start_date={startDate}&end_date={endDate}

/api/accounts/transactionsForPeriod?start_date=2004-05-14&end_date=2005-05-14

GET: api/accounts/transactionsForPeriod?start_date=2004-05-14&end_date=2005-05-14&records_per_page=4&page_number=2

Note: Date must be ISO format string YYYY-MM-DD

Example JSON response - unpaginated call:

{"data": [
{"Instrument code": "AGMF",
"Units": {"type": "Money", "currency": "ZAR", "value": "100.00"},
"Transaction id": "657000960",
"Processed date": "2005-05-13",
"Date": "2005-05-13",
"Instrument name": "Allan Gray Money Market Fund",
"Contract number": "POL2",
"Type": "ContributionTransaction",
"Sub type": "Buy",
"Amount": {"type": "Money", "currency": "ZAR", "value": "100.00"},
"Account number": "POL2-1",
"Type group": "",
"Description": "\"Buy (Once Off Investment)\"",
"Price": {"type": "Price", "currency": "ZAR", "Instrument id": 74040832, "value": "1.0000"}
},
{"Instrument code": "AGMF",
"Units": {"type": "Money", "currency": "ZAR", "value": "200.00"},
"Transaction id": "231671296",
"Processed date": "2005-05-13",
"Date": "2005-05-13",
"Instrument name": "Allan Gray Money Market Fund",
"Contract number": "POL4",
"Type": "ContributionTransaction",
"Sub type": "Buy",
"Amount": {"type": "Money", "currency": "ZAR", "value": "200.00"},
"Account number": "POL4-1",

"Type group": "",
"Description": "\"Buy (Once Off Investment)\"",
"Price": {"type": "Price", "currency": "ZAR", "Instrument id": 74040832, "value": "1.0000"}
}], "size": 2}

Error examples:

HTTP response	Code	Description
400	AccountInactiveError	Account is inactive
404	AccountNotFound	Could not find account

27 Banks

GET: /api/accounts/banks

Example request:

/api/accounts/banks

Example JSON response:


```
{
   "data": [
      "ABSA",
      "First National Bank",
      "Standard Bank",
      "Nedbank",
      "Capitec",
      "Standard Chartered Bank"
   ]
}
```


28 Value available for withdrawal

GET: /api/accounts/{accountNumber}/valueAvailableForWithdrawal

Example JSON response:


```
[ {
        "Client account number": "POL1-1",
        "Client account id": 123412341234,
        "Client account pot": “Savings component”,
        "Number of withdrawals allowed in tax year": "Many",
        "Number of withdrawals already loaded in tax year": "2",
        "Market Value in Preferred Currency": {
                "type": "Money",
                "value": "440",
                "currency": "ZAR
        },
        "Market Value in System Currency": {
                "type": "Money",
                "value": "440",
                "currency": "ZAR
        },
        "Minimum withdrawal amount": {
                "type": "Money",
                "value": "2000",
                "currency": "ZAR
        }
} ]
```

Error examples:

HTTP response	Code	Description
400	AccountInactiveError	Account is inactive
404	AccountNotFound	Could not find account

29 Regulation 28 status for new accounts

POST: /api/accounts/reg28Compliance

Returns the results of the regulation 28 validation of instrument/s based on the percentage allocation, and the maximums as per the Asset Classes (sub-classes) set on the configuration.

Example request:

```
{
        "funds" : {
                "COREQ" : "20%",
                "COR01" : "80%"
        }
}
```

Example JSON response:

{"isCompliant": "Yes",
"regulation28ComplianceFailureReason": "", }

{"isCompliant": "No",
"regulation28ComplianceFailureReason": "AssetAllocation: Local Bonds is not compliant. " }

Where regulation 28 maximums as well as regulation 28 per issuer limits are exceeded:

{"isCompliant": "No",
"regulation28ComplianceFailureReason": "Foreign-Equities with percentage 100.00% is not compliant with maximum of 25.00%, Regulation 28 issuer is not compliant for Equities; "}

Error examples:

HTTP response	Code	Description
500	Error	Duplicate key

404	InstrumentNotFound	Could not find instrument
400	MalformedRequest	No payload information