§170.315(G)(10) FHIR API Documentation

ZoomMD 4.1

ZoomMD API FHIR Documentation.pdf

TABLE OF CONTENTS

Introduction

This document describes the Integration with the SMART App Launch Framework via FHIR allows a third-party application to connect with Provider(s) using the ZoomMD Software and retrieve Electronic Health Record data from those Providers.

This API collection if for the health IT developers seeking to use our ONC 2015 Edition Cures Update Certified (g)(10) Standardized API for Health Level 7 (HL7®) Fast Healthcare Interoperability Resources (FHIR®) services.

These applications can launch from inside the user interface of ZoomMD. The framework supports Apps for use by clinicians, patients, and others via a Patient Portal or any FHIR system where a user can give permissions to launch an App. It provides a reliable, secure authorization protocol for a variety of App architectures, including Apps that run on an end-user's device as well as Apps that run on a secure server.

This documentation is intended for use by third-party application developers which will describe registration, syntax, functionality and errors/exceptions they will see when using the FHIR API to integrate with provider(s) using the ZoomMD software.

Registration

Developers wishing to integrate with the API must contact the PHI Provider that uses ZoomMD as their EMR. ZoomMD is an AWS cloud based EMR solution and as such, each API instance is separate based on the Provider (each Provider is considered a separate entity and will have their own API URL, registration for which third-party developers have access to their data via the FHIR API on their system, and each Provider is responsible for maintaining the rights of API accounts). Once a Provider decides to grant access to you as a third-party developer, they will set up a new developer account with access to the API based on the needs of the developer and agreed upon permissions with the Provider.

Configurations

The following are configuration settings that a third-party developer App needs to meet:

- The App MUST read and parse JSON responses.
- The App MUST assure that sensitive information (authentication secrets, authorization codes, tokens) are transmitted ONLY to authenticated servers, over TLS-secured channels.
- §170.315(g)(10) requires secure connection using TLS version 1.2 or higher.
- If the App is a Bulk Export application, the application MAY need to be setup to handle longer lasting connections based on the data that is being exported.
- The App MUST send requests over HTTPS. HTTP requests will be rejected.

It is published on the endpoint: https://www.zoommd.com/zoommd-file-api-endpoints

API Resources

API Resource response should request in **JSON** format.

FHIR standard **R4**

Allergies And Intolerances US Core Allergyintolerance Profile

US Core AllergyIntolerance Profile

US Core Data Element	FHIR Resource Field
Substance (Drug class)	AllergyIntolerance.code (SNOMED CT)
Substance (Medication)	AllergyIntolerance.code (RxNorm)
Reaction	AllergyIntolerance.reaction

Allergy Intolerance

For an example JSON response for Allergy Intolerance, please see the following: AllergyIntolerance.

This resource is called like the following:

```
GET [Base]/AllergyIntolerance?patient=516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve a specific AllergyIntolerance record "directly" using an ID:

```
GET [Base]/AllergyIntolerance/696897
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve the "_revinclude" information with any request using the following:

```
GET [Base]/AllergyIntolerance?_revinclude=Provenance:target&patient=516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

Search Parameter Summary:

Parameter	Type	Example
patient	reference	GET [base]/AllergyIntolerance?patient=516251

Search Parameter Combination Summary:

Parameter Combination	Type	Example
provenance:target+patient	reference	GET [base]/AllergyIntolerance?_revinclude=Provenance:target&patient=516251

Assessment And Plan Of Treatment - US Core Careplan Profile

US Core CarePlan Profile

US Core Data Element	FHIR Resource Data
narrative summary	CarePlan.text
status	CarePlan.status (like draft
intent	CarePlan.intent (like proposal
category	CarePlan.category
patient	CarePlan.subject

Care Plan

For an example JSON response for Care Plan, please see the following: CarePlan.

This resource is called like the following:

```
GET [Base]/CarePlan/_search
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve a specific CarePlan record "directly" using an ID

```
GET [Base]/CarePlan/4051
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve the "_revinclude" information with any request using the following:

```
GET [Base]/CarePlan?revinclude=Provenance:target&category=assess-plan&patient=516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

Search Parameter Summary:

Parameter	Type	Example
category	token	GET [base]/CarePlan?category=assess-plan
patient	reference	GET [base]/CarePlan?patient=516251

Search Parameter Combination Summary:

Parameter Combination	Type	Example
category+patient	token+reference	GET [base]/CarePlan?category=assess-plan&patient=516251

Careteam US Core Careteam Profile

US Core CareTeam Profile

US Core Data Element	FHIR Resource Data
patient	CareTeam.subject
participant role	CareTeam.participant[i].role (where i is the index of participant)
careteam member name	CareTeam.participant[i].member.name (where i is the index of participant)
careteam member identifier	CareTeam.participant[i].member.identifier
careteam member location	CareTeam.managingOrganization
careteam member telecom	CareTeam.telecom

Care Team

For an example JSON response for Care Team, please see the following: CareTeam.

This resource is called like the following:

```
GET [Base]/CareTeam?patient=516251&status=active
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve a specific CareTeam record "directly" using an ID

```
GET [Base]/CareTeam/516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve the "_revinclude" information with any request using the following:

```
GET [Base]/CareTeam?_revinclude=Provenance:target&patient=516251&status=proposed
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

Search Parameter Summary:

Parameter	Type	Example
patient	reference	GET [base]/CareTeam?patient=516251
status	token	GET [base]/CareTeam?status=active

Search Parameter Combination Summary:

Parameter Combination	Type	Example
patient+status	reference+token	GET [base]/CareTeam?patient=516251&status=active

Clinical Notes - US Core Documentreference

US Core DocumentReference

FHIR Resource: DocumentReference

- *Consultation Note*
- *Discharge Summary Note*
- *History & Physical*
- *Imaging Narrative*
- *Laboratory Report Narrative*
- *Pathology Report Narrative*
- *Progress Note*
- *Procedure Note*

US Core Data Element	FHIR Resource Data
status	documentReference.status
document category	documentReference.category
code	documentReference.type
patient	documentReference.subject
MIME type	documentReference.content.Typeformat
document URL	documentReference.content.attachment

Document Type	LOINC Code
Consultation Note	11488-4
Discharge Summary Note	18842-5
History & Physical	34117-2
Imaging Narrative	18748-4
Laboratory Report Narrative	11502-2
Procedure Note	28570-0
Progress Note	11506-3
Pathology Report Narrative	11526-1

Document Reference

This resource supports the Common Clinical Notes.

For an example JSON response, please see the following: DocumentReference.

This resource is called like the following:

```
GET [Base]/DocumentReference?patient=516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve a specific DocumentReference record "directly" using an ID

```
GET [Base]/DocumentReference?_id=1398530
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve the "_revinclude" information with any request using the following:

```
GET [Base]/DocumentReference?_revinclude=Provenance:target&patient=516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

Search Parameter Summary:

Parameter	Type	Example
_id	token	GET [base]/DocumentReference?_id=1398530
patient	reference	GET [base]/DocumentReference?patient=516251

Search Parameter Combination Summary:

Parameter Combination	Example
patient+category+date	DocumentReference?category=clinical-note&date=2023-12-12T06:07:13.000-08:00&patient=516251

Encounter US Core Encounter Profile

US Core Encounter Profile

US Core Data Element	FHIR Resource Data
status	Encounter.status
classification	Encounter.class
type	Encounter.type
patient	Encounter.subject

Encounter

For an example JSON response for Encounter, please see the following: Encounter.

You can retrieve a specific Encounter record "directly" using an ID

```
GET[Base]/Encounter/263703
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

Search Parameter Summary:

Parameter Combination	Type	Example
id	Token	GET [base]/Encounter/263703

Goals - US Core Goal Profile

US Core Goal Profile

US Core Data Element	FHIR Resource Data
status	Goal.achievementStatus
goal description	Goal.description
patient	Goal.subject

Goal

For an example JSON response for Goal, please see the following: Goal.

This resource is called like the following:

```
GET [Base]/Goal?patient=516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve a specific Goal record "directly" using an ID:

```
GET [Base]/Goal/4046
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve the "_revinclude" information with any request using the following:

```
GET [Base]/Goal?_revinclude=Provenance:target&patient=516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

Parameter	Type	Example
patient	reference	GET [base]/Goal?patient=516251

Health Concerns - US Core Condition Profile

US Core Condition Profile

US Core Data Element	FHIR Resource Data
status	Condition.status
category	Condition.category (like health concerns, problems & encounter diagnoses)
code	Condition.code (ICD-10-CM/SNOMED CT)
patient	Condition.subject

Condition

For an example JSON response for Condition, please see the following: Condition

This resource is called like the following:

```
GET [Base]/Condition?patient=516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve a specific Condition record "directly" using an ID:

```
GET [Base]/Condition/696871
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve the "_revinclude" information with any request using the following:

```
GET [Base]/Condition?_revinclude=Provenance:target&patient=516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

Search Parameter Summary:

Parameter	Type	Example
patient	Reference	GET [base]/Condition?patient=516251

Immunizations - US Core Immunization Profile

US Core Immunization Profile

US Core Data Element	FHIR Resource Data
status	immunization.status
vaccine code	immunization.vaccineCode
date	immunization.occurence
patient	immunization.patient

Immunizations

For an example JSON response for Immunization, please see the following: Immunization.

This resource is called like the following:

```
GET [Base]/Immunization?patient=516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve a specific Immunization record "directly" using an ID:

```
GET [Base]/Immunization/19526
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve the "_revinclude" information with any request using the following:

```
GET [Base]/Immunization?_revinclude=Provenance:target&patient=516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

Search Parameter Summary:

Parameter	Type	Example
patient	Reference	GET [base]/Condition?patient=516251

Laboratory - US Core Laboratory Result Observation Profile & Us Core Diagnosticreport

US Core Laboratory Result Observation Profile & US Core DiagnosticReport Profile for Laboratory Results Reporting.

Laboratory Result Observation

US Core Data Element	FHIR Resource Data
status	Observation.status
category	Observation.category
LOINC code	Observation.code
patient	Observation.subject

DiagnosticReport for Laboratory:

US Core Data Element	FHIR Resource Data
status	DiagnosticReport.status
category	DiagnosticReport.category
code	DiagnosticReport.code
patient	DiagnosticReport.subject
time	DiagnosticReport.issued
reported time	DiagnosticReport.effective

Diagnostic Report

This Resource covers Cardiology, Pathology, Radiology, Laboratory Result Reporting.

For an example JSON response for a Laboratory Result, please see the following: Laboratory.

This resource is called like the following:

```
GET [Base]/DiagnosticReport?patient=516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve a specific DiagnosticReport record "directly" using an ID

```
GET [Base]/DiagnosticReport/37208-Dia
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve the "_revinclude" information with any request using the following:

```
GET [Base]/DiagnosticReport?_revinclude=Provenance:target&category=LP29684-5&patient=516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

Search Parameter Summary:

Parameter	Type	Example
patient	Reference	GET [base]/DiagnosticReport?patient=516251

Search Parameter Combination Summary:

Parameter Combination	Example
patient+category	GET [base]/DiagnosticReport?category=LP29708-2&patient=516251
patient+category+date	GET [base]/DiagnosticReport?category=LAB&date=gt2023-12-10T00:00:00-08:00&patient=516251
patient+code	GET [base]/DiagnosticReport?code=30746-2&patient=516251

Medications – US Core Medication Profile, US Core Medicationrequest Profile

US Core Medication Profile

US Core MedicationRequest Profile

US Core Data Element	FHIR Resource Data
status	medicationRequest.status
intent code	medicationRequest.intent
medication	medicationRequest.medication
patient	medicationRequest.subject
date when written	medicationRequest.authoredOn
prescriber	medicationRequest.requester

Medication Request

For an example JSON response, please see the following: MedicationRequest.

This resource is called like the following:

```
GET [Base]/MedicationRequest?intent=original-order&patient=516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve a specific MedicationRequest record "directly" using an ID

```
GET [Base]/MedicationRequest/205115
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve the "_include" information with any request using the following:

```
GET [Base]/MedicationRequest?_include=MedicationRequest:medication&intent=original-
order&patient=516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve the "_revinclude" information with any request using the following:

```
GET [Base]/MedicationRequest?_revinclude=Provenance:target&intent=original-order&patient=516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

Search Parameter Summary:

Parameter	Type	Example
id	token	GET [base]/MedicationRequest/205115

Search Parameter Combination Summary:

Parameter Combination	Example
patient+intent	GET [base]/MedicationRequest?intent=original-order&patient=516251
patient+intent+status	GET [base]/MedicationRequest?intent=original-order&patient=516251&status=completed

Search Composite OR Summary:

Parameter	Type	Example
status	token	GET [base]/MedicationRequest?tatus=completed,stopped,draft
intent	token	GET [base]/MedicationRequest?intent=original-order,filler-order,instance-order

Patient Demographics - US Core Patient Profile

US Code Patient Profile

US Core Data Element	FHIR Resource Data
First Name Middle Name	Patient.name.given
Last Name	Patient.name.family
Previous Name	Patient.name
Suffix	Patient.name.suffix
US Core Birth Sex Extension	Patient.extension.BirthSex
Date of Birth	Patient.birthDate
US Core Race Extension	Patient.extension.race
US Core Extension Ethnicity	Patient.extension.ethnicity
Preferred Language	Patient.communication
Address	Patient.address
Phone Number	Patient.telecom

Patient

For an example JSON response for patient, please see the following: Patient

This resource is called like the following:

```
GET [Base]/Patient?_id=516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve a specific Patient record "directly" using an ID:

```
GET [Base]/Patient/516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve the "_revinclude" information with any request using the following:

```
GET [Base]/Patient?_id=516251&_revinclude=Provenance:target
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

Search Parameter Summary:

Parameter	Type	Example
_id	token	GET [base]/Patient?_id=516251
birthdate	date	GET [base]/Patient?birthdate=1981-09-19
family	string	GET [base]/Patient?family=Lynch
gender	token	GET [base]/Patient?gender=Male
identifier	token	GET [base]/Patient?identifier= LyncLei-111
name	string	GET [base]/Patient?name=Leif

Search Parameter Combination Summary:

Parameter Combination	Types	Example
birthdate+name	date+string	GET [Base]/Patient?birthdate=1981-09-19&name=Lynch
gender+name	token+string	GET [Base]/Patient?gender=male&name=Lynch

Problems - US Core Condition Profile

US Core Condition Profile

US Core Data Element	FHIR Resource Data
status	Condition.status
category	Condition.category (like health concerns, problems & encounter diagnoses)
code	Condition.code (ICD-10-CM/SNOMED CT)
patient	Condition.subject

Condition

For an example JSON response for Condition, please see the following: Condition

This resource is called like the following:

```
GET [Base]/Condition?patient=516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve a specific Condition record "directly" using an ID:

```
GET [Base]/Condition/696871
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve the "_revinclude" information with any request using the following:

```
GET [Base]/Condition?_revinclude=Provenance:target&patient=516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

Search Parameter Summary:

Parameter	Type	Example
patient	Reference	GET [base]/Condition?patient=516251

Procedures - US Core Procedure Profile

US Core Procedure Profile

US Core Data Element	FHIR Resource Data
status	Procedure.status
procedure code	Procedure.code
patient	Procedure.subject
procedure performed date	Procedure.performed

Procedure

For an example JSON response for Procedure, please see the following: Procedure

This resource is called like the following:

```
GET [Base]/Procedure?patient=516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve a specific Procedure record "directly" using an ID:

```
GET [Base]/Procedure/696979
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve the "_revinclude" information with any request using the following:

```
GET [Base]/Procedure?_revinclude=Provenance:target&patient=516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

Search Parameter Summary:

Parameter	Type	Example
patient	reference	GET [Base]/Procedure?patient=516251
date	date	GET [Base]/Procedure?date=2023-12-11T00:00:00-08:00

Search Parameter Combination Summary:

Parameter Combination	Types	Example
patient+date	reference+date	GET [Base]/Procedure?date=2023-12-11T00:00:00-08:00&patient=516251

Provenance - US Core Provenance Profile

US Core Provenance Profile

US Core Data Element	FHIR Resource Data
resources that Provenance is supporting	Provenance.target
date and time	Provenance.occurred
author organization	Provenance.agent.OnBehalfOf

Provenance

For an example JSON response for Provenance please see the following: AllergyIntolerance + Provenance Bundle.

You can retrieve a specific Provenance record "directly" using an ID:

```
GET [Base]/Provenance/3307
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

This resource does not support any parameters and can only be called directly.

US Core Observation Profile

US Core Observation Profile

US Core Data Element	FHIR Resource Data
status	Observation.status
code	Observation.code
patient	Observation.subject
smoking status recorded date	Observation.effective
value	Observation.value

Observation

This Resource covers Smoking Status, Pediatric Weight for Height, Laboratory Result, Pediatric BMI for Age, Pulse Oximetry and Pediatric Head Occipital-frontal Percentile.

For an example JSON response for Smoking Status, please see the following: Smoking Status.

For an example JSON response for Pediatric Weight for Height please see the following: Pediatric Weight for Height.

For an example JSON response for Laboratory Result, please see the following: Laboratory Result.

For an example JSON response for Pediatric BMI for Age, please see the following: Pediatric BMI for Age.

For an example JSON response for Pulse Oximetry please see the following: Pulse Oximetry.

For an example JSON response for Pediatric Head Occipital-frontal Percentile please see the following: Pediatric Head Occipital-frontal Percentile.

This resource is called like the following:

```
GET [Base]/Observation?code=72166-2&patient=516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve a specific Observation record "directly" using an ID

```
GET [Base]/Observation/696869
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve the "_revinclude" information with any request using the following:

```
GET [Base]/Observation?_revinclude=Provenance:target&code=72166-2&patient=516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

Search Parameter Combination Summary:

Parameter	Example
category (laboratory)	GET [base]/Observation?category=laboratory&patient=516251
category (vital-signs)	GET [base]/Observation?category=vital-signs&patient=516251
code (Smoking Status)	GET [base]/Observation?code=72116-2&patient=516251
code (Weight for Height)	GET [base]/Observation?code=77606-2&patient=516251
code (BMI)	GET [base]/Observation?code=59576-9&patient=516251
code (Pulse Oximetry)	GET [base]/Observation?code=2708-6&patient=516251
code (Pulse Oximetry)	GET [base]/Observation?code=59408-5&patient=516251
code (Body Height)	GET [base]/Observation?code=8302-2&patient=516251
code (Temperature)	GET [base]/Observation?code=8310-5&patient=516251
code (Blood Pressure)	GET [base]/Observation?code=85354-9&patient=516251
code (Body Weight)	GET [base]/Observation?code=29463-7&patient=516251
code (Head Occipital)	GET [base]/Observation?code=8289-1&patient=516251
code (Heart Rate)	GET [base]/Observation?code=8867-4&patient=516251
code (Respiratory Rate)	GET [base]/Observation?code=9279-1&patient=516251
date	GET [base]/Observation?category=vital-signs&date=2023-12-12T00:00:00-08:00&patient=516251

Parameter Combination	Example
patient+code	GET [Base]/Observation?code=8867-4&patient=516251
patient+category	GET [Base]/Observation?category=laboratory&patient=516251

UDI(s) For Patient's Implantable Device(s) - US Core Implantabledevice Profile

US Core ImplantableDevice Profile

US Core Data Element	FHIR Resource Data
code	Device.type
patient	Device.patient
UDI (optional)	Device.udiCarrier.deviceIdentifier
manufacture date(optional)	Device.manufactureDate
expiration date (optional)	Device.expirationDate
lot number (optional)	Device.lotNumber
serial number (optional)	Device.serialNumber

Device (Implantable Device)

For an example JSON response for Implantable Device, please see the following: Device.

This resource is called like the following:

```
GET [Base]/Device?patient=516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve a specific Device record "directly" using an ID

```
GET [Base]/Device/35
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve the "_revinclude" information with any request using the following:

```
GET [Base]/Device?_revinclude=Provenance:target&patient=516251
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

Search Parameter Summary:

Parameter	Type	Example
patient	reference	GET [base]/Device?patient=516251
intent	token	GET [base]/Device/35

Organization US Core Organization Profile

US Core Organization Profile

US Core Data Element	FHIR Resource Data
status	Organization.status
name	Organization.name

Organization:

For an example JSON response for Organization please see the following:

You can retrieve a specific Organization record "directly" using an ID:

```
GET [Base]/Organization/111
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

Search Parameter Summary:

Parameter	Type	Example
name	string	GET [base/Organization?name=fhir
address	string	GET [base/Organization?address=2473 South Road

Practitioner US Core Practitioner Profile

US Core Practitioner Profile

US Core Data Element	FHIR Resource Data
Identifier (NPI)	Practitioner.identifier
name	Practitioner.name

Practitioner

For an example JSON response for Practitioner please see the following: Practitioner

This resource is called like the following:

```
GET https://FHIR_URL/Practitioner
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve a specific Practitioner record "directly" using an ID:

```
GET [Base]/Practitioner/3307
Authorization: Bearer 454182a0-9b7e-41c8-9234-8f799c176c82
```

Search Parameter Summary:

Parameter	Type	Example
name	string	GET [base]/Practitioner?name=Cartwright
identifier	token	GET [base]/Practitioner?identifier=5923078435

Vital Signs – FHIR Core Vital Signs Profile

FHIR Core VitalSigns Profile

Profile Name	LOINC
Vital Signs Panel	85353-1
Respiratory Rate	9279-1
Heart rate	8867-4
Oxygen saturation	2708-6
Body temperature	8310-5
Body height	8302-2
Head circumference	9843-4
Body weight	29463-7
Body mass index	39156-5
Blood pressure systolic and diastolic	85354-9
Systolic blood pressure	8480-6
Diastolic blood pressure	8462-4

This resource is called like the following

```
GET [Base]/Observation/_search
Authorization: Bearer  454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve a specific Observation record "directly" using an ID

```
GET [Base]/Observation/5624058
Authorization: Bearer  454182a0-9b7e-41c8-9234-8f799c176c82
```

You can retrieve the "_revinclude" information with any request using the following:

```
GET [Base]/Observation?_revinclude=Provenance:target&code=85354-9&patient=516251
Authorization: Bearer  454182a0-9b7e-41c8-9234-8f799c176c82
```

Search Parameter Summary:

Parameter	Type	Example
category (vital-signs)	token	GET [base]/Observation?category=vital-signs&patient=516251

Bulk Data Export

ZoomMD will provide both "complete data" and also provide search with "_type" as a bulk data export.

For complete data:

[Base]/Group/[id]/$export

For _type data:

[Base]/Group/ [id]/$export?_type=Patient,Immunization

OAuth2 PKCE Flow

ZoomMD FHIR strongly suggests that you use the Proof Key for Code Exchange (PKCE) Flow for all public or untrusted clients. This includes native and single-page applications such as client-side JavaScript applications and mobile applications. PKCE is an OAuth2 security extension that builds upon the standard Authorization Code Flow. The PKCE Flow is used to prevent malicious attacks on applications that cannot securely store a client secret.

In order to use this flow, you must first create and configure a ZoomMD API application.

The PKCE Flow does not use the client secret key to authenticate. Instead, your application generates the following values to use:

1. code_verifier - A random cryptographic value to include in the authorization request.
2. code_challenge - A SHA-256 hash of the code_verifier value used in exchange for an access token (Bearer).

The Authorization Server stores the code_challenge and uses your application's authorization request to redirect the user to log in to authenticate with ZoomMD FHIR and to grant your application permission to their data. If successful, the Authorization Server responds and sends your client the authorization code, which is valid for one use.

Your application sends a verification request that includes both the authorization code and the code_verifier to the Authorization Server. The Authorization Server checks that the authorization code and code_verifier are valid for the user, then returns the ID token, access token, and optionally the refresh token that your application requires to make ZoomMD API calls to get the user's ZoomMD FHIR data. Use the access token to make requests using the ZoomMD API. Use refresh tokens to obtain new access and refresh tokens to ensure that your users only need to authenticate once.

To authenticate using the PKCE Flow, complete the steps that follow.

Step 1: Generate a Code Verifier and a Code Challenge

Using the PKCE Flow, you must create the cryptographically-random code_verifier value. Use SHA-256 to hash the code_verifier value as the code_challenge value.

Generate the Code Verifier

Generate a code_verifier value in your application and save it. The code_verifier should be a cryptographical value and have a minimum length of 43 characters and a maximum length of 128 characters. Later, you will use this value when:

- Generating a code_challenge value.
- Exchanging an authorization code for an access token.

For example:

```
/**
 * This method generates a secure string used as the PKCE code verifier value.
 * In PKCE, you use the code verifier value:
 * 1. When you generate a code challenge value.
 * 2. When you exchange an authorization code for a bearer JWT token.
 * @return A random code verifier value.
 */
public String generateCodeVerifier() {
    SecureRandom secureRandom = new SecureRandom();
    byte[] bytes = new byte[36];
    secureRandom.nextBytes(bytes);

    return Base64.getUrlEncoder().withoutPadding().encodeToString(bytes);
}
```

Generate the Code Challenge

Generate a code_challenge value in your application and save it. The code_challenge value is derived from your code_verifier value. Generate the code_challenge value by using SHA-256 to hash the code_verifier value, then base64 URL encode the results.

For example: code_challenge = BASE64URL-ENCODE(SHA256(ASCII(code_verifier)))

For more details, see the Proof Key for Code Exchange by OAuth Public Clients sections of RFC 7636.

The following example shows how to create a code_challenge value using SHA-256 to base64 encode the code_verifier value:

```
/**
 * This method generates a code challenge value by SHA-256 and base64 encoding the code verifier value.
 * In PKCE, you use the code challenge value when you construct the authorization request uri.
 * @param codeVerifierValue A random string value.
 * @return An encoded code challenge string value.
 * @throws NoSuchAlgorithmException
 */
public String generateCodeChallengeHash(String codeVerifierValue) throws NoSuchAlgorithmException {
    MessageDigest messageDigest = MessageDigest.getInstance("SHA-256");
    byte[] codeVerifierBytes = codeVerifierValue.getBytes(StandardCharsets.US_ASCII);
    byte[] digest = messageDigest.digest(codeVerifierBytes);

    return Base64.getUrlEncoder().withoutPadding().encodeToString(digest);
}
```

Step 2: Create the Authorization Request

Create the authorization request used to direct users to ZoomMD FHIR to authenticate their user account and to authorize your application to access their ZoomMD FHIR data.

To create an authorization request, use the authorization endpoint https://fhir.zoommd.com/oauth/authorization and include all required request query parameters.

For example, the following shows an encoded URL authorization request:

```
https://fhir.zoommd.com/oauth/authorization?client_id={your_client_id}&redirect_uri=https%3A%2F%2Flocalhost%3A8888&&response_type=code&code_challenge={generated code_challenge value}&code_challenge_method={S256}&state=235o250eddsdff&scope={patient_data%20offline_access}
```

Request Parameters

- **code_challenge** - *Required*. The code_challenge value generated by your application.
- **code_challenge_method** - *Required*. The hashing algorithm you used to generate the code_challenge. This value should always be SHA-256.
- **client_id** — *Required*. The API key for your application. You can view the API keys for all of your applications or create a new application on the My Applications page.
- **redirect_uri** — *Required*. Specify the absolute URI that you want ZoomMD FHIR to use when redirecting a user to your application. After a user authorizes your application, ZoomMD FHIR redirects the user to your application and appends the authorization code and state (and optionally the scope) values to the URI as query parameters. Wildcards are not supported except at the domain root level.
- **response_type** — *Required*. This flow uses code as the value to request an authorization code from ZoomMD FHIR.
- **state** - *Required*. To prevent cross-site request forgery, specify the arbitrary state string value you want to use to uniquely identify a user's session.
- **scope** — *Optional*. A list of the scopes that your application requires. The ZoomMD API currently supports the patient read, offiline_access, openid.

  The offline_access scope is required for returning refresh tokens.

- **nonce** — *Optional*. For the purpose of mitigating replay attacks, specify the string value to use to associate a client session with an id_token. The id_token includes information (claims) about the user and, if specified, the nonce value. The id_token is encoded as a JWT and returned in the response when you request an access token. To verify the nonce string value, decode the id_token.

Example Authorization Request

JAVA

```
/**
    * Use the URIBuilder from org.apache.http to construct an authorization request URI.
    * @param codeChallengeValue The code challenge value that your application encoded.
    * @param redirectUri URI used to redirect users to your application after they complete authorization.
    * @param clientId Your application's client id.
    * @param state A random string value used to help protect your application from cross-site request forgery.
    * @return The full authorization URI
    * @throws URISyntaxException
    */
   public String constructAuthorizationUri(String codeChallengeValue, String redirectUri, String clientId, String state) throws URISyntaxException {
       return new URIBuilder("https://fhir.zoommd.com/oauth/authorization")
               .addParameter("client_id", clientId)
               .addParameter("redirect_uri", redirectUri)
               .addParameter("scope", "offline_access openid")
```

```
            .addParameter("response_type", "code")
            .addParameter("state", state)
            .addParameter("code_challenge", codeChallengeValue)
            .addParameter("code_challenge_method", "S256")
            .build()
            .toString();
    }
```

Step 3: Get Authorization

When your application sends an authorization request to the Authorization Server, it directs the user to ZoomMD FHIR. ZoomMD FHIR prompts the user to sign in to authenticate their user account. Next, ZoomMD FHIR displays the **Permission Request** screen to allow the user to authorize your application access to their data.

If the user authorizes your application, ZoomMD FHIR appends the authorization `code` to the `redirect_uri` returned in the authorization response. The authorization `code` is used in exchange for an access token and can be used once.

Step 4: Get the Access and Refresh Tokens

To request an `access_token` and `refresh_token`, make a `POST` request to the Authorization Server using the `https://fhir.zoommd.com/oauth/token` endpoint.

Query Parameters Include the following required query parameters:

- `client_id` - The ID that is associated with your application.
- `redirect_uri` - The redirect URI that you specified as part of the authorization request.
- `code` - The authorization code returned in the authorization response.
- `code_verifier` - The code verifier value that you generated.

Header Formats

In the header, specify the preferred response formats to return as follows:

- `"Accept", "application/json"`
- `"Content-Type", "application/x-www-form-urlencoded"`

Example Request

```
/**
    * This method exchanges an authorization code, the codeVerifier value you previous generated, for an access token and a refresh token.
    * @param clientId Your application's client id.
    * @param redirectUri The redirect uri that you used as part of the authorization request URL.
    * @param code The authorization code you received from the authorization request.
    * @param codeVerifier The random code verifier value you generated.
    * @return A JSON String containing an access token and a refresh token.
    * @throws IOException
    * @throws URISyntaxException
    */
   public String exchangeCodeAndVerifierForAuthResponse(String clientId, String redirectUri, String code, String codeVerifier) throws IOException, URISyntaxException {
       URI pkceTokenUri = new URIBuilder("https://fhir.zoommd.com/oauth/token")
               .addParameter("client_id", clientId)
```

```
            .addParameter("redirect_uri", redirectUri)
            .addParameter("code", code)
            .addParameter("code_verifier", codeVerifier)
            .addParameter("grant_type", "authorization_code")
            .build();

        HttpClient httpClient = HttpClient.newHttpClient();
        HttpRequest request = HttpRequest.newBuilder(pkceTokenUri)
            .POST(HttpRequest.BodyPublishers.ofString(""))
            .header("Accept", "application/json")
            .header("Content-Type", "application/x-www-form-urlencoded")
            .build();

        HttpResponse<String> httpResponse = httpClient.send(request,
HttpResponse.BodyHandlers.ofString());

        return httpResponse.body();
    }
```

Example Response


```
{
   "token_type": "Bearer",
   "expires_in": 28800,
   "access_token":
"eyJraWQiOiIzSDdLVVpkSDlyNTh6RVVDSzNYMlR1b2o0SjV0eldWUF9FRUxQTDd6X3hjIiwiYWxnIjoiUlMyNTYi
fQ.eyJ2ZXIiOjEsImp0aSI6IkFULkZ6YUUtTVFFb2dKQ1VzZlhSUGtBZldobDllckJndVlYRWphOGFXd1RyRzgiLCJ
pc3MiOiJodHRwczovL2lkZW50aXR5LmNvbnN0YW50Y29udGFjdC5jb20vb2F1dGgyL2F1czFsbTNyeTltRjd4Mk
phMGg4IiwiYXVkIjoiaHR0cHM6Ly9hcGkuY2MuZW1haWwvdjMiLCJpYXQiOjE2NDQ0Mjc3NzMsImV4cCI6MTY0
NDQ1NjU3MywiY2lkIjoiY2M5NGMyZDYtNmU3MC00MjZjLWFmZDEtNzFjOGZhNWY0ODkwIiwidWlkIjoiMDB1
MWlieDhsaGIzYW12SFIwaDgiLCJzY3AiOlsiYWNjb3VudF9yZWFkIiwiZW1haWwiLCJjYW1wYWlnbl9kYXRhIiwic
HJvZmlsZSIsImFkZHJlc3MiLCJjb250YWN0X2RhdGEiLCJvcGVuaWQiXSwic3ViIjoiZ2Vla3NxdWFkIiwicGxhdGZ
vcm1fdXNlcl9pZCI6ImQzYTA2ZTYyLTJmMTQtNGQ5Ni05NDUzLTI2NTkxYWI1M2YwZCJ9.saJxgUYx1_OKUte
H01KV71vzjaxqTKnZ3XfD4QKtsIkjrmcf4tUWEFSzjuF1kdxLvlMoCyr_MwO81ApwYX2UU0YRhkVsfw2VLR4Kp
O89ILul7ZJZyzzsVKak8BrgLhqR1AB7X116VRjsH74vD4D-
1e4uKKUlK4yefsZE14kFgDZNi74_WPn5FBGeicwMsStdG3qms8X-6LqIxuZ5fXgSxK9wDEBe-
xJfJSqAqo1vJKMqw2JT_NvBBLDXqoSg0_q_0IqbxXEMHdX2J_DHGKrY7w1QQGmeBU_BIlewQPUxUGX-
dbRavxncBpItt1tsRLBIiPbPkQ_zDN8_J9lPIxJksA",
   "scope": "account_read offline_access",
   "refresh_token": "HFsqfgtZuGyffnD2qb-AQKWDURWAXM8hlgYIFEwaq4E",
   "id_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiIwMDEyMzQ1ejRXMzhHS3BPMGg3IiwibmFtZSI6Ikpva
G4gRG9lIiwiZW1haWwiOiJqZG9lQGFiYy5jb20iLCJ2ZXIiOjEsImlzcyI6Imh0dHBzOi8vaWRlbnRpdHkuY29uc3R
hbnRjb250YWN0LmNvbS9vYXV0aDIvYXVzengwNm81alJlZlE4RjQwaDciLCJhdWQiOiJmYThhMDZkYy1iMDAw
LTQ3NWMtYWNlZC01NWNlYzVhZWUzMjUiLCJpYXQiOjE2NDQ4NTc4NDcsImV4cCI6MTY0NDg2MTQ0Nywian
RpIjoiSUQuRFhRQ2pxQmd3M28xMnVHdlN5R2ZpQTJ2QjVFcGNvSktsajBqbFhSc2lqayIsImFtciI6WyJwd2QiX
SwiaWRwIjoiMDBvbnBtc2prYmdPc0wydlcwaDciLCJub25jZSI6Ijg2MDgyNTYwMyIsInByZWZlcnJlZF91c2Vyb
mFtZSI6Impkb2UiLCJhdXRoX3RpbWUiOjE2NDQ4NTc4NDQsImF0X2hhc2giOiJuWk5sRjduWGo2bV9UNEFKc
Xl0c1RRIn0.ToO_Ye1YHRx4hWCMBnrT6yksWvkrpHfGmXcHTh7B75A"
}
```


- token_type — This value is always set to Bearer.
- expires_in — The number of seconds remaining until the access_token expires.
- access_token — The new access_token value used to send requests with the ZoomMD API. Access tokens automatically expire 1440 minutes (86,400 seconds). Making an API call with an expired access token returns a 401 unauthorized status code.

- scope — The type of data in ZoomMD FHIR that the user is granting your application permission to use. The offline_access scope is required to get a refresh token.
- refresh_token — Each refresh token corresponds to an access token. Use the refresh token to obtain a new access_token when the corresponding access_token expires.
- id_token — The JWT that includes information about the user such as their name and email address, and if specified, the nonce value you passed in the authorization request. Decode the id_token to verify the nonce value and view user information.

Step 5: Validate the Access Token

For each new access token that you receive, follow best practices for security by validating the signature and claims of the access token.

Load the JSON web-key Set

Use the method that follows to load the ZoomMD FHIR public json web-key set used to validate access tokens.

```
/**
    * This method uses the java11 http client to load ZoomMD FHIR's public json web key set.
    * In real use cases, this value should be cached.
    * @return ZoomMD FHIR's public json web key set for the ZoomMD API.
    */
   public String getPublicJsonWebKeySet() throws IOException {
      HttpClient httpClient = HttpClient.newHttpClient();
      HttpRequest request = HttpRequest.newBuilder(URI.create("https://fhir.zoommd.com/r4/.well-
known/jwk "))
            .GET()
            .header("Accept","application/json")
            .header("Content-Type", "application/x-www-form-urlencoded")
            .build();
      HttpResponse<String> httpResponse = httpClient.send(request,
HttpResponse.BodyHandlers.ofString());

      return httpResponse.body();
   }
```

Verify the Access Token Claims

Use the method that follows to parse the access token, and then validate the signature and claims of the access token. The code examples in this step use the org.jose4j library.

Use the jwt parameter to pass in the access token. Use the jsonWebKeySetJson parameter to pass the JSON Web Key Set (JWKS) value that was returned in the previous step (Load the JSON web-key set to get the access token).

If the access token is not valid or if a required claim is missing or incorrect, an exception occurs.

```
   /**
    * This method uses the jose4j library to verify the claims on the JWT.
    * It throws an exception if it fails to parse the JWT, or the JWT does not include the expected claim.
    * @param jwt String containing a JSON web token.
    * @param jsonWebKeySetJson the set of keys available at https://fhirstaging.zoommd.com/r4/.well-
known/jwk
```

```
    * @throws JoseException
    * @throws InvalidJwtException
    */
   public void verifyJwtToken(String jwt, String jsonWebKeySetJson) throws JoseException,
InvalidJwtException {

      JsonWebKeySet jsonWebKeySet = new JsonWebKeySet(jsonWebKeySetJson);
      VerificationJwkSelector jwkSelector = new VerificationJwkSelector();
      JsonWebSignature jws = new JsonWebSignature();
      jws.setCompactSerialization(jwt);

      JsonWebKey jwk = jwkSelector.select(jws, jsonWebKeySet.getJsonWebKeys());

      JwtConsumer jwtConsumer = new JwtConsumerBuilder()
            .setRequireExpirationTime()
            .setRequireSubject()
            .setExpectedIssuer("https://fhir.zoommd.com/r4")
            .setVerificationKey(jwk.getKey())
            .setJwsAlgorithmConstraints(AlgorithmConstraints.ConstraintType.PERMIT,
AlgorithmIdentifiers.RSA_USING_SHA256)
            .build();

      jwtConsumer.processToClaims(jwt);
   }
```

Check the following claims:

- Compare the cid (client identifier) value to ensure that it matches your application API key
- The platform_user_id claim in the access token uniquely identifies a ZoomMD FHIR user.

```
JwtClaims claims = consumer.processToClaims(token);

    String cid = claims.get("cid")
    if (! apiKey.equals(cid)) {
    throw new InvalidJwtException("Api key in token incorrect");
    }
    String userId = claims.get("platform_user_id")

    if (userId == null) {
    throw new InvalidJwtException("Token is missing platform_user_id");
    }
```

Step 6: Add the Access Token to the Authorization Request

After verifying the access token claims and signature, use the access_token to send requests in the ZoomMD API by adding it to the Authorization request header in the format Authorization: Bearer {your_access_token}. When an access token expires, users must reauthorize your application with ZoomMD FHIR. Access tokens automatically expire after 60 minutes.

Making an API call with an expired access token returns a 401 unauthorized status code.

Step 7 (Optional): Check the Access Token Expiration Timestamp

You can quickly check to see if the access token is expired or not using the method that follows.

Use the jwt parameter to pass in the access token. This method returns True for an expired token if the expiration time is within 5 minutes (300 seconds) of the current time, or False if the token has not expired. If the access token expires, you can exchange it for a new access and refresh token using the *Refresh the Access Token* procedure.

```
 /**
   * This method checks if the current time is greater than or equal to the JWT expiration claim.
   * @param jwt A String JSON web token.
   * @return True if the jwt token is expired. False if the token is not expired.
   */
  public boolean checkIfTokenIsExpired(String jwt) throws InvalidJwtException, MalformedClaimException
{

     JwtConsumer jwtConsumer = new JwtConsumerBuilder()
           .setRequireExpirationTime()
           .setSkipDefaultAudienceValidation()
           .setDisableRequireSignature()
           .setSkipSignatureVerification()
           .build();
      JwtClaims jwtClaims = jwtConsumer.processToClaims(jwt);
      return NumericDate.now().getValue() >= jwtClaims.getExpirationTime().getValue();
  }
```

Example Response

```
{
 "token_type": "Bearer",
 "expires_in": "300",
 "access_token": "Access token given",
 "scope": "patient_read offline_access",
 "refresh_token": "Given refresh token"
}
```

Step 8 (Optional): Refresh the Access Token

To avoid having a user reauthenticate and reauthorize your application with ZoomMD FHIR, use the refresh token to get a new access token and a new refresh token. To refresh the access token, send a POST request to the https://fhir.zoommd.com/oauth/token authorization endpoint.

ZoomMD FHIR rate limits the token endpoint. A 429 response will likely be returned if you attempt to refresh an access token before every ZoomMD API request. ZoomMD FHIR recommends that you only send a refresh token request to get a new access token if your existing access token is expired or about to expire.

Request parameters

- **refresh_token** — *Required*. The refresh token that corresponds to the access token you are trying to refresh.
- **grant_type** — *Required*. The value is refresh_token. This specifies that the purpose of this request is to refresh an access token.
- **redirect_uri** — *Required*. Enter the redirect URI that was used as part of the authorization request URL.

Header Formats

In the header, specify the preferred response formats to return as follows:

- "Accept", "application/json"
- "Content-Type", "application/x-www-form-urlencoded"

Example Request

```
/**
    * This method exchanges a refresh token for a new access token and a new bearer token.
    * @param clientId Your application's client id.
    * @param refreshToken A refresh token.
    * @param redirectUri URI used to redirect users to your application after they complete authorization.
    * @return A JSON object containing a new access token and a new bearer token
    * @throws URISyntaxException
    * @throws IOException
    */
   public String refreshPKCEAccessToken(String clientId, String refreshToken, String redirectUri) throws URISyntaxException, IOException {
      URI pkceTokenRefreshUri = new URIBuilder("https://fhir.zoommd.com/oauth/token")
            .addParameter("client_id", clientId)
            .addParameter("redirect_uri", redirectUri)
            .addParameter("refresh_token", refreshToken)
            .addParameter("grant_type", "refresh_token")
            .build();
      HttpClient httpClient = HttpClient.newHttpClient();
      HttpRequest request = HttpRequest.newBuilder(pkceTokenRefreshUri)
            .POST(HttpRequest.BodyPublishers.ofString(""))
            .header("Accept","application/json")
            .header("Content-Type", "application/x-www-form-urlencoded")
            .build();
      HttpResponse<String> httpResponse = httpClient.send(request, HttpResponse.BodyHandlers.ofString());

      return httpResponse.body();
   }
```

Example Response

```
{
 "token_type": "Bearer",
 "expires_in": "28800",
 "access_token": "Access token given",
 "scope": "patient_read offline_access",
 "refresh_token": "Refresh token given"
}
```

- **token_type** — The value is always set to Bearer.
- **expires_in** - The access_token expiration timestamp, in seconds.
- **access_token** — The response body returns a new access_token value.

refresh_token — The response body returns a new refresh_token value.

Errors And Exceptions

Error Handling

Requests made without an access token, an invalid or expired access token, or for resources not permitted under the user's privileges defined by the authorization will result in an HTTPS 401 status code. An "Unauthorized" API may return a status code for communicating the result of each transaction/API call.

The complete list of status codes typically returned by API are as follows:

Code	Meaning
200	The request was processed appropriately.
400	Invalid parameter.
401	The request did not have a valid authorization token or none was provided. Obtain a valid authorization token and add it to the authorization header.
404	Not found.

Terms & Conditions

Terms and Conditions of Use

Please read these Terms of Use before using the API hosted on https://fhir.zoommd.com/. Your access to use the Service is based on your acceptance with these Terms. These Terms apply to all users of ZoomMD API. If you disagree with any part of the terms, then you may not access the Service.

- **Account Security**: You are responsible for safeguarding the password that you use to access the ZoomMD API and for any activities or actions under your password, whether your password is with our Service or a third-party service. You agree not to disclose your password to any third party. You must notify us immediately upon becoming aware of any breach of security or unauthorized use of your account.

- **Termination of Access and use**: Company may, in its sole discretion, terminate the right to access and use the API of any Eligible Application that has violated these Terms and Conditions.

- **License**: Company hereby grants you a non-transferable, non-exclusive, limited license to use the API subject to the terms and conditions of this Agreement. You may not, either on your own behalf or through any agent or third party decompile, disassemble, reverse engineer, or otherwise attempt to derive source code from the software components of the API, or modify or create derivative works based on the software components of the API.

- **Modifications**: Company reserves the right, at our sole discretion, to modify or replace these Terms at any time. If a revision is made, we will try to provide notice prior to any new terms taking effect. What constitutes a material change will be determined at our sole discretion. By continuing to access or use our Service after those revisions become effective, you agree to be bound by the revised terms. If you do not agree to the new terms, please stop using the Service.

- **Acceptance**: Use of the API signifies your acknowledgement and acceptance of these Terms of Use. If you do not agree to these terms of use, please do not use the API.