TickeTing Angular SDK Documentation v0.1.1

Overview

The TickeTing Angular SDK is an object oriented angular library for communicating with the TickeTing API. It comprises of multiple Factory Services used to retrieve and create Domain Object Models. These models encapsulate system data and are used to perform business logic. In this document we will describe each of the services in turn, and the object models to which they provide access, highlighting how to perform various functions against the API.

Installation

The TickeTing Angular SDK can be installed via npm as follows:

npm install @ticketing/angular

Using the SDK

Once installed the SDK can be injected into any Service or Component as follows:

```
import { TickeTing } from '@ticketing/angular';

@Component({})
class MyComponent{
        constructor(private _ticketing: TickeTing){}
}
```

Test Application

An Angular Test Applicaiton has been created to provide code examples of the SDK in use. The project can be found at: https://github.com/dadlian/ticketing-test-app

Factory Services

Factory Services are the entry points to the TickeTing SDK, and are used to access and manipulate object models from the API. Services are accessed from the main TickeTing service as follows:

```
constructor(private _ticketing: TickeTing){
        this._ticketing.hosts.list() .....
}
```

Account Service

The account service contains functionality used to create and manage accounts. This includes account creation, verification and password reset.

Methods

create - Register a new account with the TickeTing Platform.

Arguments

- account: Account (required) – An object containing the new account data. See the 'Account' model reference for more details.
- password: string (required) – The secret phrase used to secure the new account

Return Value

- newAccount: Promise<Account> - A promise that resolves in the created account, or null if creation failed.

Exceptions

- INVALID_VALUES – One or more of the specified arguments was invalid
- NOT_UNIQUE – An account with the given username or email already exists
- SERVER_ERROR – A server issue prevented creation of the new account

retrieve *- Load an account given its number*

Arguments

- number: string (required) – The unique number of the account being retreived.

Return Value

- newAccount: Promise<Account> - A promise that resolves in the account with the given number, or null if none could be found.

Exceptions

- UNAUTHORISED_ACCESS – The active session does not have sufficient permission to retrieve the requested account.
- SERVER_ERROR – A server issue prevented retrieval of the new account

Session Service

This service is used to manage sessions, which are used for authentication by the TickeTing SDK. A session needs to be created before any privileged SDK actions can be executed, or an UNAUTHORISED_ACCESS exception will be thrown.

Methods

start – Opens a new user session, used to authenticate privileged SDK actions.

Arguments

- identification: string (required) – The username or email address of the TickeTing account for which to start the session.
- password: string (required) – The password specified on the identified account.

Return Value

- newSession: Promise<Session> - A promise that resolves in a new session for the identified user account, or null if a session could not be started.

Exceptions

- UNAUTHORISED_ACCESS – The wrong identificaiton/password combination was used to start the session.
- SERVER_ERROR – A server issue prevented starting a new session

continue – Retrieves an existing user session, given its unique key

Arguments

- key: string (required) – The unique key of the session being retrieved

Return Value

- activeSession: Promise<Session> - A promise that resolves in the active session with the given key, or null if none could be found.

Exceptions

- SERVER_ERROR – A server issue prevented retrieval of the active session

Host Service

This service is used to create and retrieve hosts, which publish and manage events.

Methods

create – Creates a new host profile under which events can be published and managed.

Arguments

- host: Host (required) – An object containing the new host data. See the ‘Host’ model reference for more details.
- administrator: Account (required) – The account to assign as the event host’s default administrator.

Return Value

- newHost: Promise<Host> - A promise that resolves in the created event host, or null if a host could not be created.

Exceptions

- INVALID_VALUES – One or more of the specified host properties was invalid.
- UNAUTHORISED_ACCESS – An active session has not been started.
- NOT_UNIQUE – An event host with the specified name already exists.
- SERVER_ERROR – A server issue prevented created the new event host.

list – Returns a list of all hosts administered by the specified account

Arguments

- administrator: Account (required) – The account which all returned event hosts should be administered by.

Return Value

- hosts: Promise<Array<Host>> - A promise that resolves in a list of hosts administered by the specified account.

Exceptions

- UNAUTHORISED_ACCESS – An active session has not been started, or does not have permission to list hosts.
- SERVER_ERROR – A server issue prevented the retrieval of event hosts.

Event Service

The event service is used to retrieve TickeTing's various event listings.

Methods

submissions – Returns a list of events awaiting approval before being published.

Arguments - None

Return Value

- hosts: Promise<Array<Event>> - A promise that resolves in a list of events submitted for publication approval.

Exceptions

- UNAUTHORISED_ACCESS – The active session does not belong to a TickeTing administrative account, or one has not been started.
- SERVER_ERROR – A server issue prevented the retrieval of event hosts.

Domain Object Models

In addition to the Factory Service, the TickeTing Angular SDK makes use of Domain Object Models to execute business logic. These models are not only used to manipulate data, but define how data must be passed to service methods. In this section we describe the properties and methods of the SDK's models.

Account

The account model defines a TickeTing user account. Accounts are used to identify individual users, and are essential for performing most system actions.

Properties

username: string – The unique username associated with this user account.
email: string – The account's unique email address.
number: string – The unique account identification number.
role: string – The user' account role. Must be one of 'Customer', 'Host' or 'Administrator'.
verified: boolean – True if the account's email address has been verified, false otherwise.
title: string – The nominal title of the user. Must be one of 'Mr','Mrs','Ms'.
firstName: string – The user's forename.
lastName: string – The user's surname.
dateOfBirth: string – The user's date of birth, must be of the format 'YYYY-MM-DD'.
phone: string – The user's primary phone number.
firstAddressLine: string – The first line of the user's address, e,g, Building or House No.
secondAddressLine: string – The second line of the user's address, e.g. Street Name
city: string – The city in which the user primarily resides
state: string – The parish, province or territory in which the user primarily resides
country: string – The country in which the user primarily resides

Methods

verify – Verifies the user account's email address, given the correct verification code.

Arguments

- code: string – The verification code emailed to the user upon account creation

Return Value

- verificationSucceeded: Promise<boolean> - A promise that resolves in true if the account was successfully verified given the verificaiton code, false otherwise.

Exceptions

- NOT_UNIQUE – The user account has already been successfully verified.
- SERVER_ERROR – A server issue prevented verification of the account.

Session

A session allows for authenticated communication between a user and the API. A session must be started before a user can perform actions such as event creation, order placement or ticket redemption. In the absence of an active session, many method calls will result in the UNAUTHORISED_ACCESS exception being thrown.

Properties

started: string – The date and time at which the session was started. Must be of the format 'YYYY-MM-DD hh:mm:ss'.
open: string – True if the session is active, false otherwise.
key: string – The session's unique key.
account: Account – The account for which the session was opened.

Methods

end – Closes the session if it is active. A closed session can no longer be used to authenticate against the server.

Arguments - None

Return Value

- sessionEnded: Promise<boolean> - A promise that resolves in true if the session was closed, false otherwise.

Exceptions - None

Host

An event host represents a person, business, organisation or other entity that can stage events and sell tickets for admission. An event host must be created before events can be submitted for review.

Properties

name: string – The name of the event host.
description: string – A description of the event host.
contact: string – The name of the event host's primary contact person.
email: string – The event host's primary contact email address.
phone: string – The event host's primary contact phone.
website: string – The event host's website URL..
firstAddressLine: string – The first line of the host's primary address
secondAddressLine: string – The second line of the host's primary address
city: string – The city in which the user event host is located.
state: string – The parish, province or territory in which the user event host is located.
country: string – The country in which the user event host is located.
businessNo: string – The event host's business registration number, if applicable.
administrators: Array<string> – A list of account numbers of the event host's administrators.
events: Promise<Array<Event>> - A promise that resolves in a list of events that the host has created.

Methods

createEvent – Creates a new event staged by this host.

Arguments

- event: Event – An object containing the new event data. See the ‘Event’ model reference for more details.

Return Value

- newEvent: Promise<Event> - A promise that resolves in the newly created event, or null if the event could not be created.

Exceptions

- INVALID_VALUES – One or more of the event properties was invalid.
- UNAUTHORISED_ACCESS – The active session does not belong to a TickeTing host account, or one has not been started.
- SERVER_ERROR – A server issue prevented creation of the event.

Event

An event is a staged activity listed on the TickeTing platform for which tickets can be purchased. Events must be listed by Hosts (see above).

Properties

type: string – The type of event being staged. Must be one of ‘Standard’ or ‘Registration’.
title: string – The name of the event.
description: string – A long-form description of the event.
category: string – The top-level event category. e.g. Fete, Concert, etc.
subcategory: string – The second-level event category. e.g. All-Inclusive, Dance, etc.
venue: Location – The place at which the event is being staged.
start: string – The date and time at which the event is scheduled to start. Must be of the format ‘YYYY-MM-DD hh:mm’.
end: string – The date and time at which the event is scheduled to end. Must be of the format ‘YYYY-MM-DD hh:mm’.
public: boolean – True if the event is publically listed, false if it is private.
status: string – The current event status. Must be one of ‘Draft’, ‘Under Review’, ‘Listed’, ‘Cancelled’ or ‘Staged’.
banner: string – The URL or the base64 encoded string, of the event’s banner.

thumbnail: string – The URL or the base64 encoded string, of the event's thumbnail.
disclaimer: string – A long-form disclaimer for the event.
tags: Array<string> – A list of tags describing the event to assist in search.
popularity: number – The popularity score TickeTing has assigned to the event.
sections: Array<Section> – A list of the sections that the event is comprised of.
showings: Promise<Array<Event>> - A promise that resolves in a list of the event's showings or sub-events, if applicable.

Methods

addSection – Adds a new section to the event.

Arguments

- section: Section – An object containing the new section data. See the 'Section' model reference for more details.

Return Value

- newSection: Promise<Section> - A promise that resolves in the newly created section, or null if the section could not be created.

Exceptions

- INVALID_VALUES – One or more of the section properties was invalid.
- UNAUTHORISED_ACCESS – The active session does not belong to the account of an event administrator, or one has not been started.
- NOT_UNIQUE – This event already has a section with the given name.
- SERVER_ERROR – A server issue prevented creation of the section.

addShowing – Adds a new showing to the event.

Arguments

- showing: Event – An object containing the new showing data.

Return Value

- newShowing: Promise<Event> - A promise that resolves in the newly created showing, or null if the showing could not be created.

Exceptions

- INVALID_VALUES – One or more of the showing properties was invalid.
- UNAUTHORISED_ACCESS – The active session does not belong to the account of an event administrator, or one has not been started.
- SERVER_ERROR – A server issue prevented creation of the showing.

submit – Submits the event for TickeTing review

Arguments - None

Return Value

- submitted: Promise<boolean> - A promise that resolves in true if the event was successfully submitted, false otherwise.

Exceptions

- UNAUTHORISED_ACCESS – The active session does not belong to the account of an event administrator, or one has not been started.
- NOT_UNIQUE – The event was already successfully submitted.
- SERVER_ERROR – A server issue prevented creation of the showing.

publish – Publishes the event to TickeTing

Arguments - None

Return Value

- published: Promise<boolean> - A promise that resolves in true if the event was successfully published, false otherwise.

Exceptions

- UNAUTHORISED_ACCESS – The active session does not belong to a TickeTing administrator, or one has not been started.
- NOT_UNIQUE – The event was already successfully published.
- SERVER_ERROR – A server issue prevented creation of the showing.

Location

A location represents a geographical point at which an event can take place.

Properties

name: string – The human-readable name of the location.
longitude: number – The location's longitude as a float.
latitude: number – The location's latitude as a float.

Methods

None

Section

A section is a sub-division or area of an event to which access can be granted. This can be a General or VIP area, special seating, or otherwise reserved area.

Properties

name: string – The name of the section.
description: string – A description of the section and its offerings
basePrice: number – The price of admission to this event section.
salesStart: string – The date and time at which tickets for admission to this section go on sales. Must be of the format 'YYYY-MM-DD hh:mm'. Cannot succeed the event end time.
salesEnd: string – The date and time at which ticket sales for admission to this section end. Must be of the format 'YYYY-MM-DD hh:mm'. Cannot succeed the event end time.
capacity: number – The event section's maximum capacity.
modifiers: Array<Modifier> – A list of price modifiers for this section.

Methods

None

Modifier

A modifier defines how an event section’s admission price varies based on time of purchase, or order size. Each section can have multiple modifiers that increase or decrease the sections’ base price. If multiple modifiers are simultaneously applicable, the one that would result in the lowest base price is chosen.

Properties

name: string – The name of the modifier.
priceDelta: number – The amount by which this modifier alters the section’s base price. Cannot reduce the section’s basePrice below 0.
quantity: number – The number of times this modifier can be applied to a purchase. Cannot exceed the section’s capacity.
availableFrom: string – The date and time from when this modifier is applicable. Must be of the format ‘YYYY-MM-DD hh:mm’. Cannot preceed the section’s salesStart date.
availableTo: string – The date and time until when this modifier is applicable. Must be of the format ‘YYYY-MM-DD hh:mm’. Cannot succeed the section’s salesEnd date.
minOrder: number – The number of tickets included in an order above which this modifier is applicable.
maxOrder: number – The number of tickets included in an order below which this modifier is applicable.

Methods

None