

Customer Accounts API 1.0 CR Final Draft

May 2021 - June 2022

Latest editor's draft:

[#PUBLIC](#) Customer Accounts API 1.0 CR Final Draft

Editors:

Tim Hill ([Open Data Institute](#))

Nick Evans (imin)

Participate:

Comment on this Google Doc directly

[GitHub openactive/customer-accounts-api](#)

[File a bug](#)

Version:

1.0 CR Final Draft

Consulting:

GLL, Everyone Active, MCRactive, Westminster City Council, Gladstone, Legend, PlayFootball, Spawtz, imin, OpenActive W3C Community Group

Abstract

This document specifies an HTTP API for authenticating and managing customer accounts within a booking system, and booking on behalf of these customer accounts.

Status of This Document

This document represents a final draft of a candidate release (CR) specification, building on the existing Open Booking API proposal [#120](#). Contributions to this document are not only welcomed but are actively solicited.

Note: Yellow banners such as this are used throughout this document as notes for the reviewer, to capture key concerns and considerations raised throughout the process, and will be removed when the specification is finalised. GitHub issues referenced in these banners will also be closed.

Note: The numbering in this document will be updated upon finalisation to replace letters with numbers.

Note: When reviewing this final draft please feel free to either comment on this Google Doc directly, or raise issues via [GitHub](#).

If you have been part of the process of creating this document, please ensure that you have agreed to the [W3C Community Contributor License Agreement \(CLA\)](#), by joining the [OpenActive W3C Community Group](#) if you have not already.

Table of Contents

[Abstract](#)

[Status of This Document](#)

[Table of Contents](#)

[A. Introduction](#)

[A1. Scope and requirements](#)

[A1.1. Out of Scope in this version](#)

[A1.2 Out of Scope by design](#)

[A2. Specification dependencies](#)

[A3. Audience](#)

[A4. Conformance](#)

[Design principles](#)

[Example user journey](#)

[Registration flow example](#)

[B. Operations](#)

[B1. Checking whether a Customer Account exists](#)

[B2. Creating a new Customer Account](#)

[B3. Authenticating an existing Customer Account](#)

[B4. Viewing and updating Customer Account details from the Broker](#)

[B5. Associating a Broker-specific barcode to the Customer Account](#)

[B6. Associating an entitlement to the Customer Account](#)

[B7. Display of entitlement pricing and restrictions at the browse stage](#)

[B8. Online Booking through a Customer Account](#)

[B9. Creating and linking Dependent Accounts](#)

[B10. Booking on behalf of others](#)

[C. Model](#)

[C1. CustomerAccount](#)

[C2. Person for customer](#)

[C3. Person for emergencyContact](#)

[C4. PostalAddress](#)

[C5. Barcode](#)

[C6. Entitlement](#)

[C7. EvidenceRequestAction](#)

[C8. RecommendedAction, PrerequisiteAction and UnblockAction](#)

[C9. CustomerAccountUpdateError](#)

[C10. Organization for customer](#)

[C11. OpenBookingError subclasses](#)

[D. Endpoints](#)

[D1. GET /customer-accounts?email={email}](#)

[D2. GET /customer-accounts/me](#)

[D3. PATCH /customer-accounts/me/customer](#)

[D4. PUT /customer-accounts/me/customer](#)

[D5. PUT /customer-accounts/me/access-passes/broker-default](#)

[D6. DELETE /customer-accounts/me/access-passes/broker-default](#)

[D7. POST /customer-accounts/me/entitlements](#)

[D8. DELETE /customer-accounts/me/entitlements
?entitlementType={entitlementTypeId}](#)

[D9. GET https://example.com/openactive/entitlement-list](#)

[D10. GET /customer-accounts-rpde](#)

[E. Open data feeds](#)

[E1. isOpenBookingWithCustomerAccountAllowed within the opportunity](#)

[E2. customerAccountBookingRestriction within the opportunity](#)

[E3. eligibleEntitlementType within the Offer](#)

[E4. LateCancellationPolicy within termsOfService](#)

[F. Open Booking API behaviour](#)

[F1. Support for brokerRole](#)

[F2. customer within the Open Booking API flows for Customer Accounts](#)

[F3. attendee within the Open Booking API flows for Customer Accounts](#)

[F4. error within the Open Booking API flows for Customer Accounts](#)

[F5. Communication and interacting with the booking within the Booking System](#)

[F6. Open Booking API endpoints for Customer Accounts](#)

[F7. Customer Requested Cancellation from within the Booking System for Customer Accounts](#)

[F8. Customer Requested Cancellation from within the Broker for Customer Accounts](#)

[F9. Terms and conditions](#)

[G. OpenID Connect](#)

[G1. OpenID Connect Parameters](#)

[G2. Logout](#)

[G3. Permissible authentication configurations](#)

[G4. Permissible authentication flows](#)

[G5. Consent screen](#)

[G6. Endpoint Scopes](#)

[G7. ID Token claims](#)

A. Introduction

This section is non-normative.

The document is an output of the [OpenActive W3C Community Group](#). As part of the [OpenActive](#) initiative, the community group is developing standards that will promote the publication and use of [open opportunity data](#) in helping people to become more physically active.

The Community Group has already published specifications that define standard data models ([\[Modelling-Opportunity-Data\]](#)), support the publication and harvesting of data in standard formats ([\[RPDE\]](#)), and allow booking of opportunities via third-applications ([\[Open-Booking-API\]](#)). This existing work helps increase the discovery of opportunities and the ease of booking, for people to get more active.

A **Customer Account** in the context of this specification is an account within the Booking System.

This specification builds on the previous work of the Community Group by defining new HTTP API endpoints and behaviour of existing Open Booking API endpoints that can be implemented by Booking Systems that are publishing Opportunity Data and have already implemented the Open Booking API. By allowing third-party applications ("Brokers") to authenticate, manage and make bookings on behalf of Customer Accounts, it becomes possible for more people to participate in events or make use of leisure and sporting facilities using new or existing membership benefits.

The specification defines the flow of HTTP requests and responses between the Booking System (server) and the Broker (client). This includes detailed request and response formats, authentication flow behaviour, and potential error conditions that applications may encounter.

A1. Scope and requirements

The original [#120](#) made steps towards a solution to allow membership discounts to be applied through the Open Booking API. Although [#120](#) worked for the online booking use case, it stopped short of providing a solution that also catered for casual walk-in attendance. To address these gaps, this specification builds upon [#120](#) with the key areas of scope in this section.

All functionality defined in this specification is applicable to a wide range of use cases, including those below:

These use cases summarise the previous OpenActive W3C Community Group calls ([2019-11-20](#), [2020-01-29](#)) on this subject, and subsequent discussions.

- Allow a Customer to use a Broker to make bookings on their Customer Account, to ensure they can make use of their existing memberships.
 - A local authority that wants to have a one-stop-shop for opportunities in their region, including their own discount scheme.
 - A discovery platform that helps inactive people to find opportunities, and then through repeated use upsells an operator's membership (by redirecting to the

operator's "Join" website). The upsold Customer can then continue to make bookings via the Broker using that membership.

- A motivational fitness app that encourages users to get active and provides them native booking using their existing membership.
- Display pricing at the browse stage of the user journey for prices that would likely also be in the Seller's own website (low level of granularity).
- Where agreement between the Broker and Seller exists, allow the Broker to determine which entitlements are applied for a particular Customer Account, to allow the Customer to receive appropriate discounts.
- Allow membership schemes within a region or university to provide a single membership card or student card with a barcode that can be used in any participating Seller.
- Allow a Customer to join a membership scheme that provides e.g. free swimming across a number of Sellers, and then walk into a Seller's venue and be recognised by their entry system.

The core functionality of this specification includes:

1. Checking whether a Customer Account exists
2. Creating a new Customer Account, including accounts for children
3. Authenticating an existing Customer Account
4. Viewing and updating Customer Account details from the Broker
5. Associating a barcode to the Customer Account
6. Associating an entitlement to the Customer Account
7. Display of entitlement pricing and restrictions at the browse stage *
8. Online booking through a Customer Account *
9. Creating and linking Dependent Accounts
10. Booking on behalf of others *

* Note points 7, 8 and 10 require the implementation of the OpenActive data feeds and Open Booking API, respectively, as a prerequisite.

A1.1. Out of Scope in this version

Additional requirements that relate to Customer Accounts are currently out of scope:

- Brokering of memberships via third party join flows

A1.2 Out of Scope by design

By design this specification will not define some types of functionality.

The functionality that is currently defined as out of scope includes:

- **Payment processing for Broker memberships** - as with the Open Booking API, the design assumes that all payment handling related to memberships will be coordinated by the Broker, in a separate payment flow. Brokers are able to use the payment and reconciliation mechanisms that provide the best options for their Sellers and Customers.

- **Administration of paid Seller memberships** - this specification does not manage or manipulate Seller memberships that require direct debit; only memberships that do not require the Seller to collect payment can be added and removed by the Broker.
- **Managing all bookings associated with the Customer Account** - this specification does not provide visibility of all the bookings that have been made within this account; instead channel consistency is respected and expected: only bookings made via the Broker can be managed through the Broker.
- **Provide accurate pricing for all members at the browse stage** - only pricing for a small rationalised subset of discount "entitlements" are supported, as this specification does not attempt to replicate the complexities of membership pricing models in the Broker.

A2. Specification dependencies

Version 1.0 of the Customer Accounts specification depends on the following:

- [RPDE] Version 1.0
- [Modelling-Opportunity-Data] Version 2.x (where $x \geq 0$)
- [Open-Booking-API] Version 1.x (where $x \geq 0$)

Note that the dependencies on the [Modelling-Opportunity-Data] and [Open-Booking-API] specifications are only constrained to the major version release, and that the flexibility provided by supporting minor versions (which do not include breaking changes) enables implementers to take advantage of additional features as the specification evolves.

A3. Audience

The document is primarily intended for the following audiences:

- Software developers who want to implement the API, e.g. as part of a Booking System
- Software developers building client libraries or tools
- Software developers writing applications that will use the API ("Brokers")

The following sections introduce terminology, concepts and requirements that underpin the design of the API. This content might also be useful for a wider audience, e.g. business analysts or product managers looking for a high-level overview of the API functionality.

A4. Conformance

As well as sections marked as non-normative, all authoring guidelines, diagrams, examples, and notes in this specification are non-normative. Everything else in this specification is normative.

The key words **MAY**, **MUST**, **MUST NOT**, **NOT RECOMMENDED**, **NOT REQUIRED**, **OPTIONAL**, **RECOMMENDED**, **REQUIRED**, **SHOULD**, and **SHOULD NOT** in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Note that there is no requirement for implementers of the Open Booking API specification to adopt this specification in order to be declared compliant with the Open Booking API.

This specification makes use of the compact IRI Syntax; please refer to the Compact IRIs section from [JSON-LD].

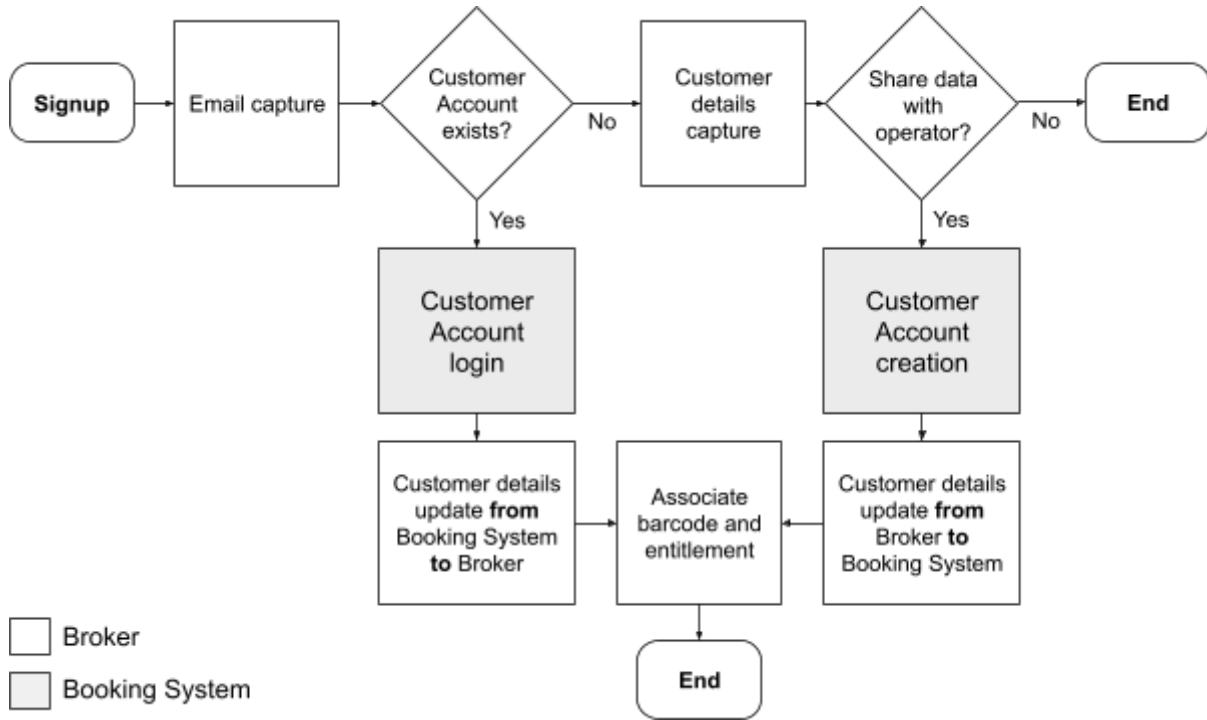
Design principles

- Explicit consent zero-trust based approach: Brokers are only given access to customer data for customers that have explicitly consented to providing them access. Customers may revoke access at any time from individual Brokers.
- Industry standard security using [OpenID Connect + OAuth 2.0](#)
- Utilise the existing booking mechanisms of the OpenActive Open Booking API
- REST-based minimal API calls leveraging JSON-LD

Example user journey

This proposal facilitates the following *illustrative* flow for a membership scheme. The flow itself is not standardised, only an example of how the features within this proposal can be combined in a user experience.

The "Customer Account" referenced in the diagram below is the "Customer Account" within the booking system.



Registration flow example

This section includes examples of the registration flow and settings page in the Broker, where the Customer controls their linked accounts.

Registration

I want to receive 15% discount:

When I book through MCRactive.com

Selecting only this option will share your name, email address, and phone number each time you make a booking

When I book at the front desk of a leisure centre

The personal details in your MCRactive account **i** will be used to create accounts with these leisure centres:



Next

Settings

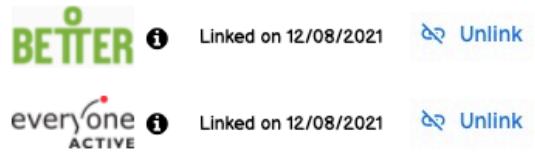
I want to receive 15% discount:

When I book through MCRactive.com

Selecting only this option will share your name, email address, and phone number each time you make a booking

When I book at the front desk of a leisure centre

Your linked leisure centres are listed below:



Save

Registration

I want to receive 15% discount:

When I book through MCRactive.com

Selecting only this option will share your name, email address, and phone number each time you make a booking

When I book at the front desk of a leisure centre

The personal details in your MCRactive account **i** will be used to create accounts with these leisure centres:



- A better.org.uk account provides access to these leisure centres
- Abraham Moss Sports Hall
- Arcadia Library & Leisure Centre
- Ardwick Sports Hall
- Belle Vue Sports Village
- Debdale Outdoor Centre
- Denmark Road Sports Centre
- East Manchester Leisure Centre
- Ardwick Sports Hall
- Belle Vue Sports Village
- Hough End Leisure Centre
- HSBC UK National Cycling Centre

B. Operations

B1. Checking whether a Customer Account exists

Within the Broker sign-up flow it may be useful to determine whether the user has one or more existing Customer Accounts in the Booking System matching a specified email address.

The number of matching Customer Accounts can be useful to display specific prompts to the user to help them understand which account to select.

The response from the API will only contain the number of matching Customer Accounts, and will not contain any personal data relating to the individual accounts. This can be used by the Broker to display appropriate messaging to the users.

Further details can be found in [Section D1](#).

B2. Creating a new Customer Account

[OpenID Connect](#) is used to create a new Customer Account, after the user has already signed up to the Broker and verified their password.

Guidance Note

Booking systems with unique constraints on email address would validate an email address at the account creation screen and prevent a new account being created if it recognised the email address. The booking system would push the user down the 'Existing Account' route, which may or may not need to cater for 'Forgot Password' scenarios.

In addition to being OpenID Connect security best practice in terms of authentication, informing the user that they are "creating an account with <seller>" will implicitly improve other user journeys of the Booking System, as they will have an awareness that they already have an existing account - for example logging into the Booking System directly to book a course or upgrade to a direct-debit membership. It also provides the user with an opportunity to realise that they already have an existing Customer Account under a different email address.

Custom `screen_hint=signup` and `allow_signup=true` OpenID Connect request parameters are also included, which causes the OpenID Connect authentication step to default to the "create account" page, the contents of which are controlled by the Booking System:

Booking System

everyone
ACTIVE

MCRactive is requesting that you create a new account with everyoneactive.com

Sign up to everyoneactive.com

Email

Use MCRactive account details to create a new everyoneactive.com account ⓘ

I have read and agree to the [health questionnaire](#)

[Already have an everyoneactive.com account?](#)

[Everyone Active Terms and Privacy Policy](#)

The email address is passed as the [login_hint](#) OpenID Connect request parameter, so the user does not have to reenter it.

The new account is created with only the provided email address, and the Broker is given [offline access](#) to the newly created account, with an OpenID claim indicating that a new account was created. Customer details may then be updated using "[Updating Customer Account details in the Seller](#)".

"Already have an everyoneactive.com account?" links to "[Authenticating an existing Customer Account](#)", for cases where the user already has an existing Customer Account with a different email address.

This specification discourages free text fields on the create page other than the "Email" field, to ensure a smooth user journey in all cases. The Customer has already entered the data in the Broker.

The account MUST be created in an "uninitialised" state. If the account is not initialised with customer details within e.g. 5 minutes, the Booking System SHOULD delete the "uninitialised" account. The Booking System MUST replace any existing "uninitialised" account with the same email address, to ensure that subsequent attempts to the same account will succeed.

The additional claim `https://openactive.io/isPendingInitialization` MUST be used to indicate whether the user that has been authenticated has had their account initialised yet. If this

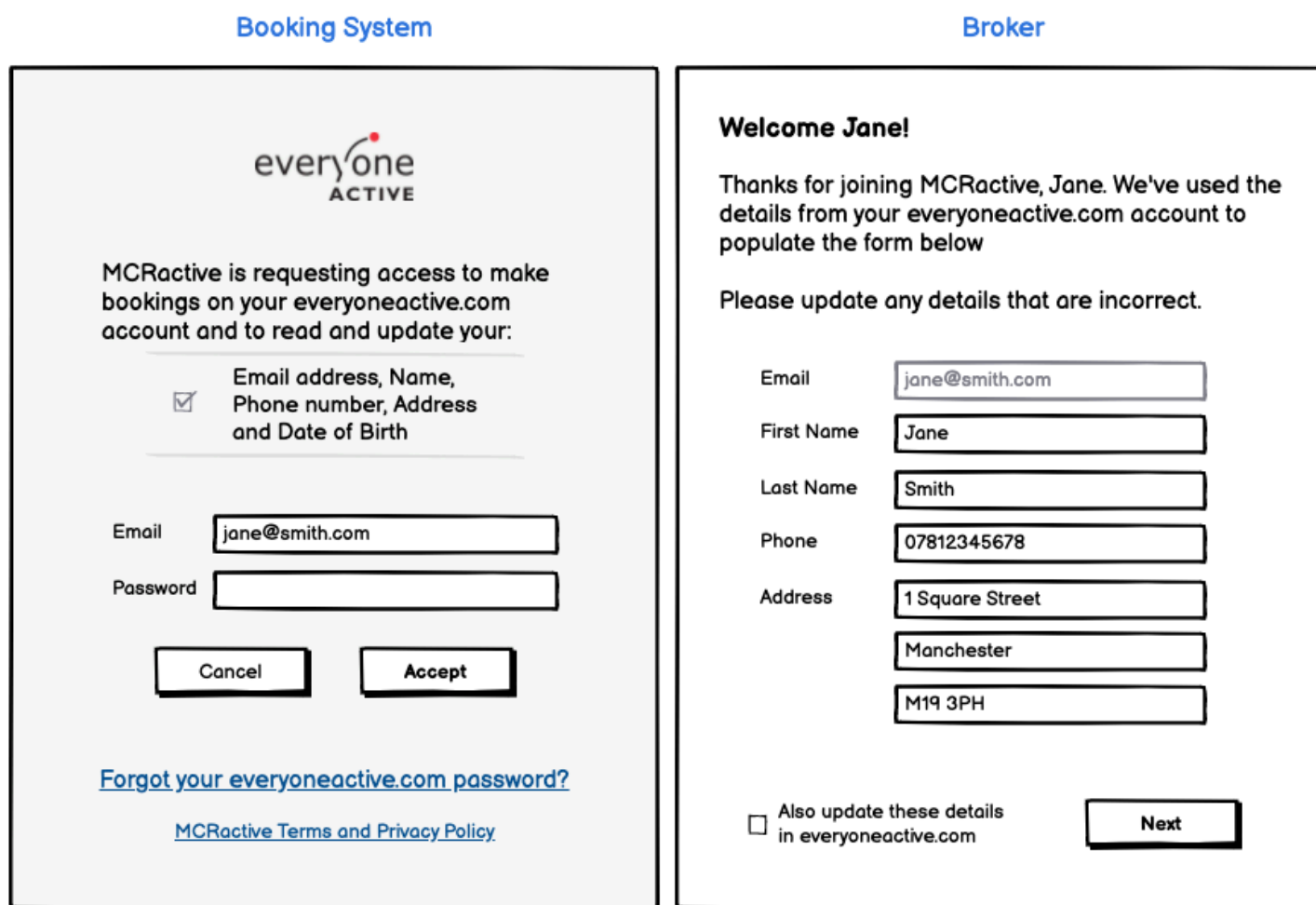
claim is present, the Broker MUST use "[Initialising the Customer Account following account creation](#)" to initialise the account when control is returned to it. If the claim is not present, the Broker MUST treat the account as an existing account that does not require initialisation.

For more information see [Section C](#).

B3. Authenticating an existing Customer Account

[OpenID Connect](#) is used to authenticate an existing Customer Account. This flow may be used early in the Broker signup process to import existing customer details from the Booking System.

The default behaviour of OpenID Connect authentication is a "login" page, controlled by the Booking System:



Note: all data that is transferred between parties MUST be listed on the left hand side of the diagram, and displayed on the right hand side of this diagram. Although the transfer of special category data (as defined by the General Data Protection Regulation) is outside the scope of this proposal, it is important to note that left-hand side interface elements dedicated to special category data MUST be associated with tick boxes to ensure that the transfer of any special category data remains optional..

The booking system MUST NOT include an option to create an account on the login page unless `allow_signup=true` is set, as the Broker might not have already captured details sufficient for account creation, and also might not support account creation.

The email address is passed as the [login_hint](#) OpenID Connect request parameter. Upon successful login the Broker is given [offline access](#) to the account. The OpenID Connect [ID Token](#) provides the Broker with the [customer details](#) (including whether the email has been verified), which may be used by the Broker signup process (*illustrative example above, right*).

Review Note

Due to limitation of customization of the UserInfo endpoint in some platforms (e.g. Azure B2C does not allow custom claims, IdentityServer does not easily allow embedded objects e.g. "address"), and the additional complexity it presents to all parties to have two different models in use within the same specification, this data is provided by a standardised `/customer-accounts/me` endpoint. This ensures that a standard model can be used for both GET and PATCH of the same data, and any scope-based restrictions would then affect this.

The Booking System MAY implement the UserInfo endpoint, and MUST ensure that the `id_token` contains a minimum set of claims if the "openid" and "profile" scopes are included:

- sub
- given_name
- family_name
- email
- email_verified
- phone_number

Further details are provided to the Booking System by the Broker using "[Updating Customer Account details in the Seller](#)".

"Forgot your everyoneactive.com password?" links to the usual mechanism within the booking system, which may include a "magic link" or one-time code, instead of forcing a password reset. It is recommended that this link opens a new tab, and includes the name of the account (e.g. "everyoneactive.com") to reduce support calls.

In keeping with the OpenID Connect specification, the refresh token MAY not expire until it is revoked, which simplifies implementation for all parties.

For more information see [Section G](#).

B4. Viewing and updating Customer Account details from the Broker

Following existing account authentication or new account creation, the Broker may update the customer details in the Booking System. The screens below are both *illustrative* examples of how this might be displayed to the user by the Broker.

Details in an update form could be pre-populated from the Seller's existing data (left) or the Broker's existing data (right), depending on the needs of the Broker.

The update is only available using the authentication token of the Customer, obtained via one of the OpenID Connect flows above, for security and to ensure explicit consent.

Broker

Welcome Jane!

Thanks for joining MCRactive, Jane. We've used the details from your everyoneactive.com account to populate the form below

Please update any details that are incorrect.

Email	<input type="text" value="jane@smith.com"/>
First Name	<input type="text" value="Jane"/>
Last Name	<input type="text" value="Smith"/>
Phone	<input type="text" value="07812345678"/>
Address	<input type="text" value="1 Square Street"/>
	<input type="text" value="Manchester"/>
	<input type="text" value="M19 3PH"/>

Also update these details in everyoneactive.com

Broker

Update Details

We've used the details from your MCRactive account to populate the form below

These details will be shared with everyoneactive.com

First Name	<input type="text" value="Jane"/>
Last Name	<input type="text" value="Smith"/>
Phone	<input type="text" value="07812345678"/>
Address	<input type="text" value="1 Square Street"/>
	<input type="text" value="Manchester"/>
	<input type="text" value="M19 3PH"/>

Customer account details also include the following:

- The membership number of the Customer
- Any outstanding actions that the Customer needs to perform in order to make bookings (e.g. paying outstanding debts).

B5. Associating a Broker-specific barcode to the Customer Account

The Broker-specific barcode MUST use a pattern (e.g. prefix, suffix) specific to the Broker (agreed out-of-band). The Broker is given full ownership of all barcodes within its designated namespace, and can reassign them from other Customer Accounts within the booking system to ensure that they are associated with the correct Customer.

The specific type and format of the Broker-specific barcode is agreed out-of-band between the Broker and the Seller, and could be a QR code.

Note the expectation is that setting the Broker-specific barcode SHOULD NOT overwrite the primary barcode on the Customer Account, but instead be recognised as an alternative barcode for the same Customer Account by the Booking System.

The Broker-specific barcode may also be assigned by the Seller directly in the Booking System, and the Customer Accounts updates feed ([Section D10](#)) SHOULD be checked periodically by the Broker and the Customer Accounts GET endpoint ([Section D2](#)) compared to the Broker's expectations. The Broker MAY choose to adopt the new barcode as its own primary barcode for the Customer if a new barcode is detected.

Further details can be found in [Section D5](#).

Considerations for guidance

It is recommended that the range of barcodes that the Broker uses should be different between physical cards and virtual cards - for example "BROKERV000001" and "BROKERP000001". This ensures that the Broker can easily distinguish between a virtual Barcode that has been set on the linked account in error (and hence can be overwritten) and a physical Barcode that the Customer may choose to associate with their Broker account.

B6. Associating an entitlement to the Customer Account

Entitlements may be added and removed from a Customer Account. These do not have a monthly membership fee associated with them from the Booking System side, and simply give the customer discounts relevant to membership schemes.

The Broker may renew/extend an entitlement using the same mechanism that they use to add it.

Entitlements are allocated by the Broker, and hence the Broker MUST maintain business logic to determine which entitlement to apply to each Seller within each Booking System based on the customer's eligibility.

Note there could be a monthly membership fee associated with an entitlement, and collected from the Broker side, however registration of this payment is outside the scope of this specification, as such a membership fee is likely to enable Entitlements for a number of activities

across a number of bookings systems (and so the payment would not be directly related to a single seller).

~~It is recommended that the Broker updates the entitlement every month automatically to keep it current, by using a 2 month expiry window. Note that **this will not require user action every month**, and that renewal is managed entirely by the Broker (e.g. for MCRactive it will be an annual process). Hence, if the customer chooses to disconnect from the Broker from within the Booking System, then the entitlement will naturally expire within 2 months. Note this is only a recommendation, and Broker and Seller may also agree to use e.g. a standard 12 month expiry window.~~

Further details can be found in [Section D7](#) and [Section D8](#).

B7. Display of entitlement pricing and restrictions at the browse stage

Membership schemes publish their entitlement levels as open data in [SKOS](#) format. Sellers that are part of the membership scheme may then choose to provide specific pricing for entitlement levels within that membership scheme.

Such entitlement pricing ("rationalised discount pricing") may be included in the open data feeds to facilitate displaying pricing at the activity search and browse stage. To avoid an explosion in open data volume, membership schemes are required to rationalise the number of available entitlement levels to facilitate this.

It is recommended that Booking Systems provide an administration interface that allows Sellers to assign entitlement levels from membership schemes to Offers within their feed (i.e. their internal membership price levels).

To allow entitlement levels to be represented accurately in the open data feed, each entitlement level MUST have exactly one "entitlement type". If multiple internal "subscriptions" are used to create an "entitlement level", these MUST be mapped to a single "entitlement type", and not be exposed outside of the Booking System.

In scenarios where the Customer has opted for sharing only the minimum data required to use the service, the creation of a Customer Account may not be appropriate. Hence Offers that contain entitlement pricing have an @id that can be used to book via guest checkout in the usual way.

For Customer Account bookings the price is calculated based on the Custom Account eligibility, and not on a selected Offer. Offers are marked as "indicative" at the browse stage where the Customer Account has additional entitlements in addition to the entitlement added by the membership scheme. The Broker may also use the "indicative" indicator to signpost the Customer to add the item to their basket to determine its exact price.

In order to avoid disappointment from scenarios where the Customer authenticates only to be told they are not permitted to book the activity with their membership,

`customerAccountBookingRestriction` is displayed at the browse stage to indicate e.g. "Gym induction required" or "Gold members only". This does not require exposing the complexity of the underlying membership structure via open data, and is designed such that the Seller can describe the restriction in plain text.

Further details can be found in [Section E3](#).

B8. Online Booking through a Customer Account

The Customer Account is specified as part of the Open Booking API flow, both for the customer and the attendees as required. A specific Offer is not included in the selection for Customer Account attendees, as the price is determined by the Customer Account.

Guests of the Customer Account are permitted (via additional OrderItems), however these guests are not entitled to any discounts/entitlements that are associated with the Customer Account.

Further details can be found in [Section F](#).

B9. Creating and linking Dependent Accounts

GitHub Discussion

A GitHub discussion relates to this section:

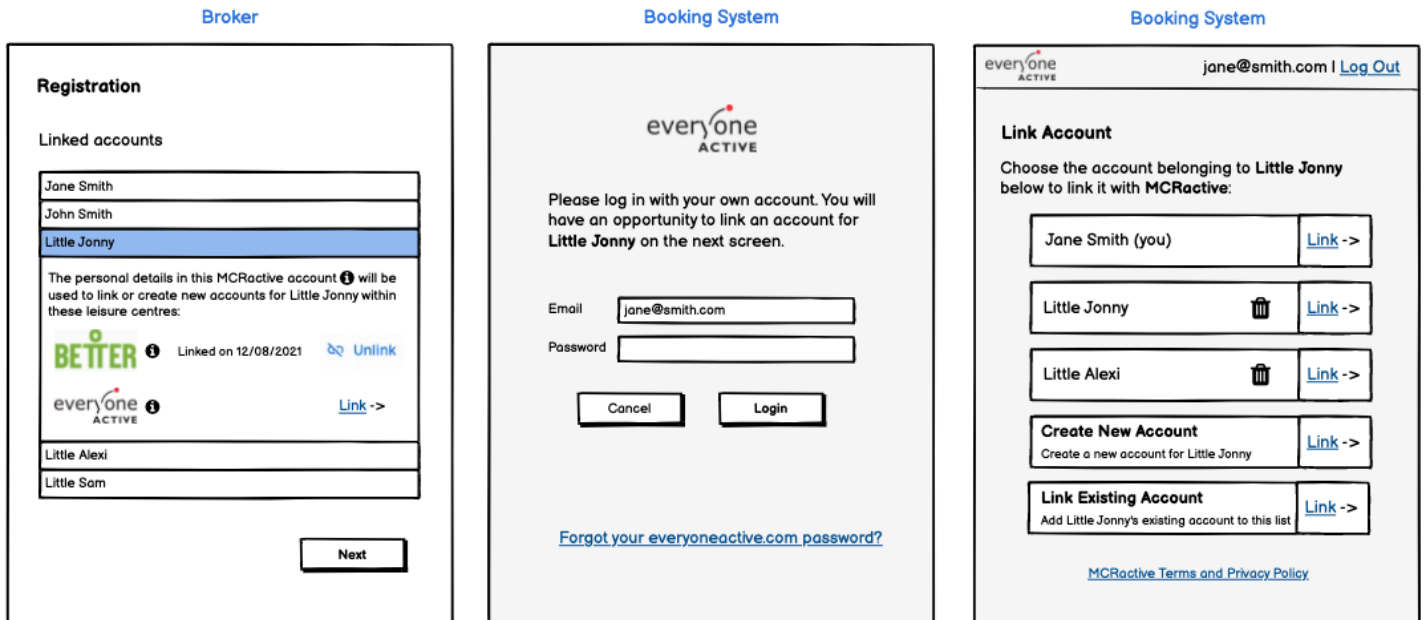
<https://github.com/openactive/customer-accounts/issues/5>

Dependent Accounts are Customer Accounts that are accessed and maintained by an associated Customer Account within the booking system. They facilitate use cases such as Customers who are too young, who are under the care of another, or who are simply disinterested in having their own account.

Booking System models for Dependent Accounts vary from simple "parent-child" relationships to more complex group-based administration. To facilitate the broadest possible range of "Dependent Account" models within the Booking System and to simplify implementation, this specification does not attempt to support the synchronisation of such relationships or groups between Broker and Booking System.

Instead, Dependent Accounts are treated exactly as Customer Accounts throughout the API, except for in the OpenID Connect flow.

When the `openactive_flow_type=customer_delegation` parameter is used in the OpenID Connect flow, the flow facilitates the selection of an existing Dependent Account from a list, and the creation of a new one, alongside the administration of the Booking System's own Dependent Account model.



The API defined in this specification does not currently provide any mechanism for a Broker to access a list of related or dependent accounts within the Booking System’s model.

Once authenticated, Dependent Accounts are simply accessed and updated via the endpoints specified in Section D; from the Broker's perspective they appear as standard Customer Accounts, with their own set of tokens.

Changes to the management relationships of a Dependent Account within the Booking System SHOULD NOT impact existing Broker connections. The Booking System MUST provide a means to display which Brokers a Dependent Account is connected to, such that if the Dependent Account was to be managed by a new Customer Account, the new Customer Account may view a list of currently connected Brokers and choose to disconnect them.

The Booking System SHOULD also provide a mechanism within the OpenID Connect flow to remove erroneously created Dependent Accounts found in the list, to encourage the Customer to accurately maintain the Booking System’s records.

Further details of the OpenID Connect flow can be found in [Section G4](#).

B10. Booking on behalf of others

GitHub Discussion

A GitHub discussion relates to this section:

<https://github.com/openactive/customer-accounts/issues/5>

To facilitate use cases such as “booking on behalf of a friend” and “booking on behalf of a Dependent Account”, it is possible for one Customer Account to book on behalf of another.

As permissions and relationships between Customer Accounts are not synchronised between the Broker and Booking System within this specification, the Broker is entirely responsible for ensuring that explicit consent is provided by both parties when one “books on behalf of” another.

Additionally, Customer Accounts (including Dependent Accounts) MUST grant permission to a particular Broker, to allow them to be used as the subject of any "on-behalf-of" booking by that Broker, using the “openactive-openbooking-on-behalf-of” scope.

Booking on-behalf-of follows the same approach as [Online Booking through a Customer Account](#), using an @id reference to the relevant Customer Account for the attendee. Any Customer Account may be specified as the attendee, provided it has granted permission for the “openactive-openbooking-on-behalf-of” scope to the Broker.

C1, C2 and B request extract

```
"attendee": {
  "@type": "Person",
  "hasAccount": "https://eg.com/customer-accounts/fdc14503-275e-46d3-9922-45b986c9f9aa"
},
```

To provide additional security for personal data, attendee details for Customer Accounts MUST NOT be included in the response from C1/C2/B, and MUST be dereferenced by the Broker if required using an access token for that attendee. The customer details MUST be included in the response from C1/C2/B, as the customer is always based on the authenticating access token.

For privacy, error messages for failed on-behalf-of bookings that require resolution via a change to the state of the target Customer Account MUST NOT include details of issues relating to that Customer Account (for example, a debt or ban on the account). Such errors at C1/C2/B MUST be generic using the OutstandingActionOnCustomerAccountError OrderItem error (“A booking could not be made on behalf of this account at this time”), and the resolution MUST be described in an “outstandingAction” available on the target’s Customer Account via their own access tokens.

Note an Organization cannot be specified as an attendee, and hence a CustomerAccount of type Organization cannot be the target of an on-behalf-of booking.

Further details can be found in [Section F3](#) and [Section F4](#).

C. Model

This section includes details of the model used as part of the proposal.

Note to the reader: This section is useful to reference in conjunction with Section D, rather than reading it stand-alone.

C1. CustomerAccount

CustomerAccount models the account that belongs to a customer. It is a subclass of schema:Thing.

GitHub Discussion

A GitHub discussion relates to this section:

<https://github.com/openactive/customer-accounts-api/issues/7>

Property	Status	Type	Notes
@type	required	schema:Text	CustomerAccount
@id	required	schema:URL	A URI providing a unique identifier for the resource. This MUST match the <code>https://openactive.io/customerAccountId</code> custom claim within the Open ID Connect <code>id_token</code> .
schema:identifier	required	schema:Text	The identifier of the Customer Account used by the Booking System. This MUST match the "sub" claim within the Open ID Connect <code>id_token</code> .
oa:accountNumber	optional	schema:Text	A unique customer-facing identifier for the Customer Account.
schema:customer	required	schema:Person or schema:Organization	The Person or Organization to whom this Customer Account belongs
oa:accessPass	optional	Array of schema:Barcode	The barcode, QR code, magnetic stripe, or swipe card associated with this Customer Account, within their own namespaces.
oa:hasHiddenEntitlements	required	schema:Boolean	Whether there are any additional entitlements (other than those listed in entitlement) or other types of discounts are associated with the Customer Account that will influence pricing, and therefore whether the pricing for the entitlement in the feed SHOULD be treated as

			indicative.
<u>oa:outstandingAction</u>	optional	Array of <u>schema:Action</u>	Outstanding actions on this Customer Account, such as the resolution of outstanding debts or membership renewal. These may prevent the Customer from making bookings.
<u>oa:entitlement</u>	optional	Array of <u>oa:Entitlement</u>	The current valid and active entitlements associated with this customer. Note that expired or inactive entitlements are not included in this list.

C2. Person for customer

The Person class is extended to include additional properties relevant to memberships

Property	Status	Type	Notes
<u>@type</u>	required	<u>schema:Text</u>	Person
<u>schema:identifier</u>	optional	<u>schema:Text</u>	The identifier of the Customer used by the Broker.
<u>oa:hasAccount</u>	optional	<u>oa:CustomerAccount</u>	Reference to the CustomerAccount associated with this Customer, for use within the Open Booking API flow.
<u>schema:email</u>	optional	<u>schema:Text</u>	Email address used to uniquely identify the Customer.
<u>schema:givenName</u>	optional	<u>schema:Text</u>	Name of the Customer.
<u>schema:familyName</u>	optional	<u>schema:Text</u>	Name of the Customer.
<u>schema:telephone</u>	optional	<u>schema:Text</u>	Telephone number of the Customer.
<u>schema:birthDate</u>	optional	<u>schema:Date</u>	Date of birth in <u>ISO 8601</u> format
<u>schema:gender</u>	optional	<u>schema:GenderType</u> or <u>schema:Text</u>	The gender of the Customer. May be either <u>https://schema.org/Male</u> , <u>https://schema.org/Female</u> , or a free text string, in line with the <u>schema.org definition</u> ("as with the gender of individuals, we do not try to enumerate all possibilities"). A booking system MAY map all free text strings in the request to the value "Other", in which case the value "Other" MUST be included in the response.
<u>schema:address</u>	optional	<u>schema:PostalAddress</u>	Address of the Customer.
<u>schema:honorificPrefix</u>	optional	<u>schema:Text</u>	An honorific prefix preceding a

			Person's name such as Dr/Mrs/Mr. This specification does not provide a controlled vocabulary of values.
oa:emergencyContact	optional	schema:Person	The emergency contact for the Customer

Use of `gender` is preferred to `honorificPrefix` by this specification, as it is considered a more contemporary approach to data capture.

Note that this specification does not currently support the Booking System or Seller specifying "required" fields, or support the Booking System requiring consistency between values of `gender` and `honorificPrefix`. Any minimum field requirements MUST be agreed between the Broker and the Seller out-of-band.

The rationale for this is that minimum field requirements enforced at an API level only serve to force work-arounds of "dummy values" when they are in excess of the varying business requirements of the Seller (often the case, as following GDPR personal data captured should always be the minimum), hence an out-of-band agreement on minimum requirements is a more effective approach overall for all parties. This also reduces technical complexity as there is no need to list "required" fields or deal with edge cases where they are not provided. It might be that customers logging into the booking system UI directly are prompted to complete missing fields to continue, which is the expected behaviour for cases where the agreed minimum is less than the system's own imposed minimum. Booking systems are encouraged to use a placeholder character value such as "-" if their database does not support null values for required fields.

C3. Person for emergencyContact

The Person class is used to represent an emergency contact.

Property	Status	Type	Notes
@type	required	schema:Text	Person
schema:name	required	schema:Text	Full name of the emergency contact.
schema:telephone	required	schema:Text	Telephone number of the emergency contact.

C4. PostalAddress

The PostalAddress class matches that defined in the Open Booking API.

Property	Status	Type	Notes
@type	required	schema:Text	PostalAddress
schema:streetAddress	optional	schema:Text	Street address
schema:addressLocality	optional	schema:Text	Town or other area
schema:addressRegion	optional	schema:Text	Region

schema:postalCode	optional	schema:Text	Postcode
schema:addressCountry	optional	schema:Text	ISO 3166-1 alpha-2 country code.

C5. Barcode

The Barcode class includes both the namespace and text of the barcode.

Property	Status	Type	Notes
@type	required	schema:Text	Barcode
@id	required	schema:URL	A URI providing a unique identifier for the resource
schema:identifier	required	schema:Text	The namespace identifier of the Barcode
schema:text	optional	schema:Text	Barcode number

Note the barcode type (e.g. "Code 128") is not specified, and MUST be handled out-of-band.

C6. Entitlement

The [oa:Entitlement](#) class subclasses [schema:Permit](#).

Property	Status	Type	Notes
@type	required	schema:Text	Entitlement
schema:validFrom	optional	schema:DateTime	The date and time the entitlement becomes valid. This MUST be in the past. Date and time allows for short-lived entitlement use cases.
schema:validUntil	required	schema:DateTime	The date and time that the entitlement is no longer valid. This MUST be in the future. Date and time allows for short-lived entitlement use cases.
oa:entitlementType	required	skos:Concept	For the request, this is the value of the @id of the Concept being referenced. For the response, this the full Concept object including the @id and prefLabel . See further Section D9 .
oa:evidenceRequestAction	optional	oa:EvidenceRequestAction	Any evidence request associated with the entitlement.

Note all Entitlements created or viewed by this API are currently active (i.e. not in the past or future).

C7. EvidenceRequestAction

This action is specifically related to entitlements that require evidence.

Property	Status	Type	Notes
@type	required	schema:Text	EvidenceRequestAction
schema:description	required	schema:Text	Description of the request, for display to the user.
schema:target	recommended	Array of schema:EntryPoint	<p>An EntryPoint that contains only the urlTemplate property MUST reference the URL where evidence can be provided. Other EntryPoints may be provided to include platform specific bindings as outlined here and here.</p> <p>Note this is not required for circumstances where the evidence request cannot be completed digitally.</p> <p>This property MUST be omitted when actionStatus is set to https://schema.org/CompletedActionStatus.</p>
schema:actionStatus	optional	schema:ActionStatusType	<p>This may be https://schema.org/ActiveActionStatus or https://schema.org/CompletedActionStatus, or omitted if the evidence request is not being tracked.</p> <p>If actionStatus is not provided, it is assumed that this request MUST be treated as fire-and-forget, as the booking system does not support status tracking (and e.g. the link should therefore be sent via email immediately).</p>

Note that if an [EvidenceRequestAction](#) is not tracked (i.e. its status is unknown beyond its association with the entitlement), [actionStatus](#) MUST be omitted entirely.

C8. RecommendedAction, PrerequisiteAction and UnblockAction

These actions are required to resolve the Customer Account back to a healthy state. They are primarily intended to advise the Customer on issues with their account that will prevent some or all bookings from taking place.

`UnblockAction` is reserved for actions that are required to allow the Customer to make *any* booking (e.g. paying outstanding debts, an "office message", a lapsed membership, etc). The Broker SHOULD make these known to the Customer.

`PrerequisiteAction` is reserved for actions that are required to make *specific* bookings (e.g. attending an in-person gym induction, or viewing an online gym induction). The Broker SHOULD make these known to the Customer.

`RecommendationAction` is reserved for actions that are informational which are generally useful for the upkeep of the account (e.g. attending an in-person gym induction, or viewing an online gym induction). The Broker SHOULD make these known to the Customer.

Property	Status	Type	Notes
@type	required	schema:Text	RecommendationAction , PrerequisiteAction or UnblockAction
schema:description	optional	schema:Text	Description of the request, for display to the user.
schema:target	recommended	Array of schema:EntryPoint	An EntryPoint that contains only the urlTemplate property MUST reference the URL where the issue can be resolved. Other EntryPoints may be provided to include platform specific bindings as outlined here and here . Note this is not required for circumstances where the resolution cannot be completed digitally.

Note that [actionStatus](#) is not supported for [RecommendationAction](#), [PrerequisiteAction](#) or [UnblockAction](#).

C9. CustomerAccountUpdateError

This error contains a child `errors` relating specific properties

Property	Status	Type	Notes
@type	required	schema:Text	CustomerAccountUpdateError
schema:name	required	schema:Text	A short, human-readable summary of the problem type, that MUST communicate "Customer Account could not be updated".
schema:description	required	schema:Text	A human-readable explanation specific to this occurrence of the problem, providing specific information about why the error occurred in this particular case.
schema:error	optional	Array of	Child errors relating to the overall

		oa:OpenBookingError	error. The description is expected to be displayed to the user.
--	--	-------------------------------------	--

C10. Organization for customer

The `Organization` class is extended to include additional properties relevant to memberships

GitHub Discussion

A GitHub discussion relates to this section:

<https://github.com/openactive/open-booking-api/issues/229>

Property	Status	Type	Notes
@type	required	schema:Text	Person
schema:identifier	optional	schema:Text	The identifier of the Customer used by the Broker.
oa:hasAccount	optional	oa:CustomerAccount	Reference to the CustomerAccount associated with this Customer, for use within the Open Booking API flow.
schema:email	optional	schema:Text	Email address used to uniquely identify the Customer.
schema:name	optional	schema:Text	Name of the Customer.
schema:telephone	optional	schema:Text	Telephone number of the Customer.
schema:address	optional	schema:PostalAddress	Address of the Customer.
oa:primaryContact	optional	schema:Person	The primary contact for the Customer

Note that this specification does not currently support the Booking System or Seller specifying "required" fields. Any minimum field requirements MUST be agreed between the Broker and the Seller out-of-band.

The rationale for this described in [Section C2](#).

C11. OpenBookingError subclasses

The table below contains all the additional subclasses of `OpenBookingError` defined within this specification

Error Type (@type)	Status Code (oa:statusCode)	Use Case (schema:name)
CustomerAccountUpdateError	400	Customer Account could not be updated
EmailAddressAlreadyExistsError	400	The supplied email address is already associated with another account within the Booking System

D. Endpoints

This section includes details of the proposed endpoints.

Note to the reader: Detail regarding specific properties used in these examples can be found in Section C.

All endpoints MUST only be accessed using HTTPS with an appropriate Authorization header. All Booking System endpoints described below are secured using OpenID Connect and OAuth 2.0.

D1. GET /customer-accounts?email={email}

Checks whether any Customer Accounts with the given email address..

This endpoint is accessible without a user logging in, via the Client Credentials flow.

Brokers using this endpoint as part of a sign-up flow MUST be careful to prevent [Account Enumeration](#) attacks, by using a Web Application Firewall or other similar mitigation.

Authorization
Type: Bearer token obtained via OpenID Connect Client Credentials flow Required Scope: openactive-customeraccount-query
Request
GET /customer-accounts?email=jane@example.com
Response
HTTP/1.1 200 OK { "@context": "https://openactive.io/", "@type": "ItemList", "numberOfItems": 2 }

D2. GET /customer-accounts/me

Returns all details associated with the Customer Account.

The OpenID Connect scopes `openactive-customeraccount-read` and `openactive-customeraccount-modify` grant access to all personal data specified as part of the Person class in Section C2 of this specification. Hence these endpoints are not affected by the [standard scopes](#) included in the OpenID Connect specification.

Note that this version of the specification does not include any provision for the sharing of [special category data](#) (e.g. racial or ethnic origin), which would require separate explicit consent via an additional OpenID Connect scope that is not defined here.

Note that the `accessPass` barcodes MUST have a namespace-specific prefix (e.g. "MCR"), where one is applicable. All Customer Account barcodes MUST be included in the response.

Also note that a namespace-specific `Barcode` may be assigned by the Seller via the Booking System and appear in the response without the Broker having added it (e.g via the physical provision of a broker-specific membership card at front-of-house); hence the Broker SHOULD expect to check the `accessPass` property for updates.

For systems that support only a single barcode per Customer Account, the Broker may wish to check for an existing `accessPass` property and notify the user before it is overwritten by the PUT call detailed in D5.

The `hasHiddenEntitlements` property is used to indicate whether additional entitlements or other types of discounts are associated with the customer account, and therefore whether the pricing for the entitlement in the feed MUST be treated as indicative.

In order to reduce the size of the response to only the most relevant information, the `entitlement` array includes only current entitlements (and hence excludes expired or entitlements).

The Broker SHOULD NOT poll this endpoint for updates, and SHOULD instead use the Customer Account updates feed specified in [Section D10](#).

Authorization
Type: Bearer token obtained via OpenID Connect Authorization Code flow Required Scope: <code>openactive-customeraccount-read</code>
Request
GET <code>/customer-accounts/me</code>
Response
HTTP/1.1 200 OK { "@context": "https://openactive.io/", "@type": "CustomerAccount", "@id": "https://eg.com/customer-accounts/fdc14503-275e-46d3-9922-45b986c9f9aa", "identifier": "fdc14503-275e-46d3-9922-45b986c9f9aa", "accountNumber": "CA00000123", "customer": { "@type": "Person", "email": "alexjones@example.com", "telephone": "020 811 8055", "givenName": "Alex", "familyName": "Jones", "birthDate": "1970-01-01", "gender": "https://schema.org/Female",

```

"address": {
  "@type": "PostalAddress",
  "streetAddress": "Raynes Park High School, 46A West Barnes Lane",
  "addressLocality": "New Malden",
  "addressRegion": "London",
  "postalCode": "NW5 3DU",
  "addressCountry": "GB"
},
"emergencyContact": {
  "@type": "Person",
  "name": "Ralph Jones",
  "telephone": "020 811 8055"
}
},
"accessPass": [
  {
    "@type": "Barcode",
    "text": "MCR012345620",
    "identifier": "MCR"
  }
],
"hasHiddenEntitlements": false,
"entitlement": [
  {
    "@type": "Entitlement",
    "evidenceRequestAction": {
      "@type": "EvidenceRequestAction",
      "description": "Please provide better.org.uk with valid documents for your
Concessionary membership. Document types accepted include current entitlement letter or
Bank statement showing payment for Universal Credit or Entitlement for Carers Allowance.
Until valid documents are provided and verified your membership will not be active and you
will be unable to enjoy the full benefits of your membership. You will need to reset your
better.org.uk password first if you do not already have one. Once you have uploaded your
documents and these have been verified, you will be able to enjoy the full benefits of
your membership.",
      "target": [
        {
          "@type": "EntryPoint",
          "urlTemplate":
"https://better.legendonlineservices.co.uk/enterprise/account/login"
        }
      ],
      "actionStatus": "https://schema.org/CompletedActionStatus"
    },
    "validFrom": "2021-08-26T11:40:00+01:00",
    "validUntil": "2022-09-25T10:45:33+00:00",
    "entitlementType": {
      "@type": "Concept",
      "@id":
"https://localhost:5001/openactive/entitlement-list#621c8dc0-ce78-4551-88e2-93812b1d9f07",
      "prefLabel": "Better Adult Conc MCR Active CA"
    }
  }
],
"outstandingAction": [
  {
    "@type": "UnblockAction",
    "description": "There is unpaid debt on your account. You MUST log in to your
account and pay this debt in order to continue to make bookings.",
    "target": [
      {
        "@type": "EntryPoint",
        "urlTemplate":
"https://better.legendonlineservices.co.uk/enterprise/account/login"
      }
    ]
  }
]

```

```

    }
  ]
},
{
  "@type": "UnblockAction",
  "description": "Your better.org.uk pay monthly membership has expired. Please log in
to your better.org.uk to renew your membership.",
  "target": [
    {
      "@type": "EntryPoint",
      "urlTemplate":
"https://better.legendonlineservices.co.uk/enterprise/account/login"
    }
  ]
},
{
  "@type": "PrerequisiteAction",
  "description": "You have not yet completed your gym induction, so you will not be
able to book gym activities. Please book a gym induction."
}
]
}

HTTP/1.1 403 Forbidden

{
  "@context": "https://openactive.io/",
  "@type": "CustomerAccountUninitializedError",
  "name": "This Customer Account has not been initialised",
  "description": "Please initialise the Customer Account before attempting this operation"
}

```

D3. PATCH /customer-accounts/me/customer

Updates the personal data related to the Customer in the Customer Account.

Review Note

Note that this endpoint no longer provides a best-efforts update, as per discussion on the call 2 December 2021 it is easier to reason about an endpoint that performs an atomic update.

This endpoint provides an atomic partial update capability, hence the following status codes:

- **200 OK** is returned when all requested updates have succeeded, including the full updated `Person` or `Organization` object. The response **MUST** include values for all properties for the `Person` or `Organization` if they exist, even if they were not included in the request.
- **400 Bad Request** is returned when the update of any individual property fails. The `error` array contains all field errors encountered during the update, such that if all the erroring properties were omitted from a subsequent request, then this request **MUST** succeed.
- **403 Forbidden** containing a single `OpenBookingError` subclass is used in the case of an error where updating contact details is simply not permitted, such as when the Customer has a monthly direct debit membership (and therefore they **MUST** update their details directly in the booking system), or when the Seller's configuration prevents all Customers from updating their own details (e.g. to prevent member fraud).

Note that this specification does not currently support the Booking System or Seller specifying "required" fields through the API, and this call is intended as a *partial* update. This is especially important given that the update may be coming from a system that may not hold values for all properties. Therefore this call MUST succeed even if no fields at all are supplied, and even if the resulting record does not meet the booking system's minimum requirements within its own UI. Any minimum field requirements MUST be agreed between the Broker and the Seller out-of-band. See Section C2 for further rationale.

Where updating a particular property is not supported by the implementation, an appropriate error is included in the `error` array. The `instance` property of the child error relates to properties of the `Person` or `Organization`, rather than nested properties.

Properties that are not supported at all by the implementation also return a `PropertyUpdateNotSupportedError`. For Brokers the extraneous properties can then simply be removed and the request retried.

Authorization
Type: Bearer token obtained via OpenID Connect Authorization Code flow Required Scope: <code>openactive-customeraccount-modify</code>
Request
<pre>PATCH /customer-accounts/me/customer { // Includes all properties that are attempted to be updated "@type": "Person", "email": "alexjones@example.com", "telephone": "020 811 8055", "givenName": "Alex", "familyName": "Jones", "birthDate": "1970-01-01", "gender": "https://schema.org/Female", "address": { "@type": "PostalAddress", "streetAddress": "Raynes Park High School, 46A West Barnes Lane", "addressLocality": "New Malden", "addressRegion": "London", "postalCode": "NW5 3DU", "addressCountry": "GB" }, "emergencyContact": { "@type": "Person", "name": "Ralph Jones", "telephone": "020 811 8055" } } PATCH /customer-accounts/me/customer { // Includes all properties that are attempted to be updated "@type": "Organization", "name": "Company Limited", "address": { "@type": "PostalAddress", "streetAddress": "Raynes Park High School, 46A West Barnes Lane", "addressLocality": "New Malden", "addressRegion": "London", "postalCode": "NW5 3DU", "addressCountry": "GB" } }</pre>

```
},
"primaryContact": {
  "@type": "Person",
  "name": "Ralph Jones",
  "email": "alexjones@example.com",
  "telephone": "020 811 8055"
}
}
```

Response

HTTP/1.1 400 Bad Request

```
{
  "@context": "https://openactive.io/",
  "@type": "CustomerAccountUpdateError",
  "name": "Customer Account could not be updated"
  "description": "There was an issue with updating properties within the Customer
Account."
  "error": [
    {
      "@type": "EmailAddressAlreadyExistsError",
      "name": "The supplied email address is already associated with another account
within the Booking System",
      "instance": "https://schema.org/email",
      "description": "This email address is already associated with another account"
    },
    {
      "@type": "PropertyUpdateNotSupportedError",
      "name": "This system does not support updates to this property",
      "instance": "https://schema.org/email",
      "description": "Updating email address outside of the booking system is not
supported"
    },
    {
      "@type": "PropertyInvalidError",
      "name": "The value of the property supplied was not valid",
      "instance": "https://schema.org/telephone",
      "description": "The phone number supplied was not valid"
    },
    {
      "@type": "AccessDeniedError",
      "name": "This Broker does not have permission to perform this operation",
      "instance": "https://schema.org/birthDate",
      "description": "Permission was not granted to update the date of birth"
    }
  ]
}
```

HTTP/1.1 403 Forbidden

```
{
  "@context": "https://openactive.io/",
  "@type": "AccessDeniedError",
  "name": "This Broker does not have permission to perform this operation",
  "description": "You have a pay-monthly membership or other non-MCRActive membership with
better.org.uk, so it is not possible to update your contact details here. Please log into
better.org.uk to update your details there."
}
```

HTTP/1.1 403 Forbidden

```
{
  "@context": "https://openactive.io/",
```

```
"@type": "CustomerAccountUninitializedError",
"name": "This Customer Account has not been initialised",
"description": "Please initialise the Customer Account before attempting this operation"
}
```

D4. PUT /customer-accounts/me/customer

Initialises the Customer Account following account creation.

This endpoint exists separately to D3 to prevent unintentional overwriting of customer data, as the D4 call MUST only be used once, when a new Customer Account is first created.

Guidance Note

Some booking systems may detect duplicate account details based on the values of specific fields, and would prevent a duplicate account being created. The `CustomerAccountDuplicateDetectedError` allows the Broker to direct the user to log in to their existing account by directing them back to the login screen in [Section B3](#), or alternatively allow them to enter different personal details.

This endpoint functions as a total overwrite of the customer data (except for email address), and functions exactly as D3 (to simplify implementation), with these exceptions:

1. It MUST only be called on an uninitialised Customer Account
2. Email address cannot be updated by an initialisation call (as it was determined in the GUI of the account creation step), and the call will fail if this is attempted with a 403 `EmailAddressCannotBeInitializedError`
3. A 409 error is used to indicate that the initialisation failed if the customer account has previously been initialised, with a `CustomerAccountAlreadyInitialisedError` error
4. A 409 error is used to indicate that the initialisation failed because a deduplication check has determined that a Customer Account for this Customer already exists, with a `CustomerAccountDuplicateDetectedError` error. The "description" of this error is expected to be displayed to the Customer.
5. A 400 error is used to indicate that the initialisation failed for any other reason, with a `CustomerAccountInitializionError` error

Note that this specification does not currently support the Booking System or Seller specifying "required" fields through the API, therefore this call MUST succeed if no fields at all are supplied, and even if the resulting record does not meet the booking system's minimum requirements within its own UI. Any minimum field requirements MUST be agreed between the Broker and the Seller out-of-band. See Section C2 for further rationale.

Properties that are not supported at all by the implementation also return a `PropertyUpdateNotSupportedError`. For Brokers the extraneous properties can then simply be removed and the request retried.

Authorization

Type: Bearer token obtained via OpenID Connect Authorization Code flow
Required Scope: openactive-customeraccount-create

Request

```
PUT /customer-accounts/me/customer
{
  "@type": "Person",
  "email": "alexjones@example.com",
  "telephone": "020 811 8055",
  "givenName": "Alex",
  "familyName": "Jones",
  "birthDate": "1970-01-01",
  "gender": "https://schema.org/Female",
  "address": {
    "@type": "PostalAddress",
    "streetAddress": "Raynes Park High School, 46A West Barnes Lane",
    "addressLocality": "New Malden",
    "addressRegion": "London",
    "postalCode": "NW5 3DU",
    "addressCountry": "GB"
  },
  "emergencyContact": {
    "@type": "Person",
    "name": "Ralph Jones",
    "telephone": "020 811 8055"
  }
}

PUT /customer-accounts/me/customer
{ // Includes all properties that are attempted to be updated
  "@type": "Organization",
  "name": "Company Limited",
  "address": {
    "@type": "PostalAddress",
    "streetAddress": "Raynes Park High School, 46A West Barnes Lane",
    "addressLocality": "New Malden",
    "addressRegion": "London",
    "postalCode": "NW5 3DU",
    "addressCountry": "GB"
  },
  "primaryContact": {
    "@type": "Person",
    "name": "Ralph Jones",
    "email": "alexjones@example.com",
    "telephone": "020 811 8055"
  }
}
```

Response

```
HTTP/1.1 400 Bad Request

{
  "@context": "https://openactive.io/",
  "@type": "CustomerAccountUpdateError",
  "description": "There was an issue with updating properties within the Customer Account."
  "error": [
    {
      "@type": "PropertyInvalidError",
      "instance": "https://schema.org/telephone",

```

```

    "description": "The phone number supplied was not valid"
  }
]
]

HTTP/1.1 403 Forbidden
{
  "@context": "https://openactive.io/",
  "@type": "EmailAddressCannotBeInitializedError",
  "name": "Email cannot be updated using an initialization call",
  "description": "Email cannot be updated using an initialization call."
}

HTTP/1.1 409 Conflict
{
  "@context": "https://openactive.io/",
  "@type": "CustomerAccountAlreadyInitialisedError",
  "name": "Customer Account has already been initialised",
  "description": "Customer account has already been initialised."
}

HTTP/1.1 409 Conflict
{
  "@context": "https://openactive.io/",
  "@type": "CustomerAccountDuplicateDetectedError",
  "name": "The Customer Account details provided have been detected as a duplicate of an existing account",
  "description": "The Customer Account has been detected as a duplicate. Please try authenticating with your customer account directly."
}

HTTP/1.1 400 Bad Request
{
  "@context": "https://openactive.io/",
  "@type": "CustomerAccountInitializionError",
  "name": "Customer Account initialisation failed",
  "description": "Customer account initialisation failed due to a database error."
}

```

D5. PUT /customer-accounts/me/access-passes/broker-default

Updates the barcode related to the Customer Account within the specified namespace.

Review Note

It was decided on the 9 December 2021 call that it is the responsibility of the Seller to ensure that a clash does not occur in the namespace-specific patterns used by different Brokers, and hence this specification does not require a specific pattern be used or validated by the Booking System. The advantage of this approach is that it does not assume any particular format for the barcodes in use, as different access control systems have different formats and requirements.

GitHub Discussion

A GitHub discussion relates to this section:

Each Broker's OpenID Connect Client ID is assigned a notional default "namespace", which is used to store the barcode within the Booking System. This "namespace" MAY be configurable by the Seller. This specification defines the alias "broker-default" that MUST be used to reference this "namespace" within the path of the request. No other values are permissible within the path in this version of the specification.

The barcode `text` value MUST have a namespace-specific pattern (e.g. prefix or suffix of "MCR"). The Broker is given ownership of all barcodes within the "namespace", so they will not clash with barcodes in other namespaces (which may be controlled by other Brokers or applications).

The Booking System SHOULD implement access control to prevent Brokers updating barcodes in namespaces for which they do not have permission.

The Booking System MUST include the actual namespace `identifier` in the response of this request (rather than the alias) and in the `GET /customer-accounts/me` endpoint, to assist with debugging.

To simplify implementation, the API call below is a forced update, hence:

- If the barcode is not currently associated with the Customer Account, it MUST replace all other barcodes associated with the Customer Account within the namespace.
- If the barcode is already associated with another Customer Account, it MUST be removed from that account and associated with the current one.
- The call is idempotent, so will succeed if the barcode is already associated.

In systems that support only a single barcode per Customer Account, where the existing barcode is in a different namespace it is simply replaced by the Broker-specific barcode. The operational implications of this MUST be agreed out-of-band, and this is generally not an issue for operators who have moved to digital membership cards - as these can be easily updated.

Note that guest booking barcodes (`accessPass` on the `OrderItem`) are provided for guest bookings, and are not related to the Customer Account barcode described above.

Authorization
Type: Bearer token obtained via OpenID Connect Authorization Code flow Required Scope: openactive-customeraccount-modify
Request
PUT /customer-accounts/me/access-passes/broker-default { "@context": "https://openactive.io/", "@type": "Barcode", "text": "MCR0123456789" }
Responses

```

HTTP/1.1 201 Created
{
  "@context": "https://openactive.io/",
  "@type": "Barcode",
  "identifier": "MCR", // Namespace identifier
  "text": "MCR0123456789"
}

HTTP/1.1 409 Conflict
{
  "@context": "https://openactive.io/",
  "@type": "BarcodeExistsOutsideOfNamespaceError",
  "name": "The specified barcode already exists in another barcode namespace."
}

HTTP/1.1 403 Forbidden
{
  "@context": "https://openactive.io/",
  "@type": "AccessDeniedError",
  "name": "Access to this resource is not permitted",
  "description": "Access to this barcode namespace is not permitted for this broker."
  // Note that this should only occur in the case of misconfiguration
}

HTTP/1.1 403 Forbidden
{
  "@context": "https://openactive.io/",
  "@type": "CustomerAccountUninitializedError",
  "name": "This Customer Account has not been initialised",
  "description": "Please initialise the Customer Account before attempting this operation"
}

```

D6. DELETE /customer-accounts/me/access-passes/broker-default

Removes the broker-specific barcode related to the Customer in the Customer Account.

This SHOULD be called by the Broker before the Customer disconnects their Customer Account.

Authorization
Type: Bearer token obtained via OpenID Connect Authorization Code flow Required Scope: openactive-customeraccount-modify
Request
DELETE /customer-accounts/me/access-passes/broker-default
Response
HTTP/1.1 204 No Content HTTP/1.1 403 Forbidden { "@context": "https://openactive.io/", "@type": "CustomerAccountUninitializedError", "name": "This Customer Account has not been initialised", "description": "Please initialise the Customer Account before attempting this operation"

```
}
```

D7. POST /customer-accounts/me/entitlements

Adds or extends an entitlement on the Customer Account.

If an entitlement of the same type already exists, that entitlement is extended rather than added.

In the case that an entitlement cannot be applied (for example, entitlement conflicts with one that is already applied to the account), a 409 Conflict is returned. The Booking System SHOULD allow all entitlements to be applied where possible, and use the most favorable of all available prices when offering pricing to the Customer.

Authorization

Type: Bearer token obtained via OpenID Connect Authorization Code flow
Required Scope: openactive-customeraccount-modify

Request

```
POST /customer-accounts/me/entitlements

{
  "@context": "https://openactive.io/",
  "@type": "Entitlement",
  "validFrom": "2021-08-26T11:40:00+01:00",
  "validUntil": "2022-09-25T10:45:33+00:00",
  "entitlementType":
  "https://data.mcractive.com/openactive/entitlement-list#5e78bcbe-36db-425a-9064-bf96d09cc351"
}
```

Response

```
HTTP/1.1 201 Created

{
  "@context": "https://openactive.io/",
  "@type": "Entitlement",
  "validFrom": "2021-08-26T11:40:00+01:00",
  "validUntil": "2022-09-25T10:45:33+00:00",
  "evidenceRequest": {
    "@type": "EvidenceRequestAction",
    "target": [
      {
        "@type": "EntryPoint",
        "urlTemplate": "https://legend.example.com/upload-form/1234/magic-link/abc"
      }
    ],
    "description": "Please provide better.org.uk with valid documents for your  
Concessionary membership. Document types accepted include current entitlement letter or  
Bank statement showing payment for Universal Credit or Entitlement for Carers Allowance.  
Until valid documents are provided and verified your membership will not be active and you  
will be unable to enjoy the full benefits of your membership. You will need to reset your  
better.org.uk password first if you do not already have one. Once you have uploaded your  
documents and these have been verified, you will be able to enjoy the full benefits of  
your membership."
  }
}
```



```
    "actionStatus": "https://schema.org/CompletedActionStatus"
  },
  "entitlementType": {
    "@type": "Concept",
    "@id":
"https://data.mcractive.com/openactive/entitlement-list#5e78bcbe-36db-425a-9064-bf96d09cc351",
    "prefLabel": "MCRactive Adult Resident",
    "inScheme": "https://data.mcractive.com/openactive/entitlement-list"
  }
}
```

HTTP/1.1 409 Conflict

```
{
  "@context": "https://openactive.io/",
  "@type": "EntitlementConflictError",
  "name": "The entitlement cannot be applied due to other entitlements already associated
with the Customer.",
  "description": "This customer already has a paid monthly membership"
}
```

HTTP/1.1 409 Conflict

```
{
  "@context": "https://openactive.io/",
  "@type": "EntitlementNotAppropriateError",
  "name": "The entitlement cannot be applied as it is not appropriate for the Customer."
  "description": "Customer date of birth does not match the entitlement."
}
```

HTTP/1.1 409 Conflict

```
{
  "@context": "https://openactive.io/",
  "@type": "EntitlementExpiryInvalidError",
  "name": "Expiry date MUST be in the future"
}
```

HTTP/1.1 403 Forbidden

```
{
  "@context": "https://openactive.io/",
  "@type": "CustomerAccountUninitializedError",
  "name": "This Customer Account has not been initialised",
  "description": "Please initialise the Customer Account before attempting this operation"
}
```

D8. DELETE /customer-accounts/me/entitlements ?entitlementType={entitlementTypeId}

Removes an entitlement from the Customer Account.

Only entitlements that are visible via GET /customer-accounts/me may be removed using this endpoint.

All broker-related entitlements SHOULD be removed using this call before the Customer disconnects their Customer Account.

This endpoint returns success if the entitlement has already been removed, and is therefore idempotent

Authorization
Type: Bearer token obtained via OpenID Connect Authorization Code flow Required Scope: openactive-customeraccount-modify
Request
DELETE /customer-accounts/me/entitlements?entitlementType=https://data.mcractive.com/openactive/entitlement-list#5e78bcbe-36db-425a-9064-bf96d09cc351
Response
HTTP/1.1 204 No Content HTTP/1.1 403 Forbidden { "@context": "https://openactive.io/", "@type": "CustomerAccountUninitializedError", "name": "This Customer Account has not been initialised", "description": "Please initialise the Customer Account before attempting this operation" }

D9. GET <https://example.com/openactive/entitlement-list>

List of all entitlement types in the memberships scheme as open data, in the form of a SKOS vocabulary.

This endpoint is implemented by the owner of the membership scheme (e.g. MCRactive). It is consumed by the Booking System for use when populating entitlement type options within its own administration interface.

The `@id` values in the entitlement list SHOULD remain unchanged wherever possible.

An entitlement types list MAY be hierarchical. A Broker MUST match a broader entitlement type associated with an Offer to any narrower entitlement type that is associated with a Customer within the Broker, if no Offer with an exactly matching `eligibleEntitlementType` is available. To illustrate this using the example entitlement list below: a Customer with an entitlement type of "Adult Pay & Play AcmeCity Non-Resident" would match an Offer with `eligibleEntitlementType` that included "Adult Pay & Play". This allows Sellers to simplify their entitlement configuration for simple cases, and optimise data exchange volume.

Guidance Note

Separate guidance should cover:

- Dealing with identifiers changing, or dealing with updates.
- Where and how the SKOS vocab ID could be exchanged between parties

Future versions of this specification may include a simple catalogue of entitlement lists (such that a Seller may be presented with a dropdown list of applicable entitlements within the Booking System administration interface). This is useful for booking systems with a high volume of Sellers.

Authorization

Type: None (open data)

Request

GET /entitlements/entitlements.jsonld

Response

HTTP/1.1 200 OK

```
{
  "@context": "https://openactive.io/",
  "@id": "https://data.example.com/entitlements/entitlements.jsonld",
  "@type": "ConceptScheme",
  "title": "Entitlement Types for AcmeCity",
  "license": "https://creativecommons.org/licenses/by/4.0/",
  "concept": [
    {
      "@type": "Concept",
      "@id": "https://data.example.com/entitlements#7baf9a0a-02a5-4c3c-93a7-4bb3fff9efc5",
      "prefLabel": "Adult Pay & Play",
      "narrower": [
        {
          "@type": "Concept",
          "@id": "https://data.example.com/entitlements#41730e9d-e07b-4737-a0b8-5bd12fe6be79",
          "prefLabel": "Adult Pay & Play AcmeCity Non-Resident"
        },
        {
          "@type": "Concept",
          "@id": "https://data.example.com/entitlements#db7b1d0e-effd-4083-9469-701a683ef434",
          "prefLabel": "Adult Concession Pay & Play AcmeCity Non-Resident"
        },
        {
          "@type": "Concept",
          "@id": "https://data.example.com/entitlements#a7d5bd07-45b2-4402-9bc3-3fda9b16c6de",
          "prefLabel": "Adult Concession Pay & Play AcmeCity Non-Resident (Age 60+)"
        },
        {
          "@type": "Concept",
          "@id": "https://data.example.com/entitlements#4212b6cf-4b1e-45b6-ace9-b037d628c955",
          "prefLabel": "Adult Concession Pay & Play AcmeCity Resident (Age 60+)"
        },
        {
          "@type": "Concept",
          "@id": "https://data.example.com/entitlements#041c56ff-a897-4ae3-a870-35324ffc8a65",
          "prefLabel": "Adult Pay & Play AcmeCity Resident"
        }
      ]
    },
    {
      "@type": "Concept",
      "@id": "https://data.example.com/entitlements#cfe4d7c6-8528-4731-a854-6c1b9127ae1d",
      "prefLabel": "Junior Pay & Play",
      "narrower": [

```

```

    "@type": "Concept",
    "@id": "https://data.example.com/entitlements#3f772b2e-6fd1-48fd-b0dc-a13d373f10b7",
    "prefLabel": "Junior Pay & Play AcmeCity Non-Resident"
  },
  {
    "@type": "Concept",
    "@id": "https://data.example.com/entitlements#71cccadd-ff7a-46dd-a714-bd3a10723845",
    "prefLabel": "Junior Concession Pay & Play AcmeCity Non-Resident"
  },
  {
    "@type": "Concept",
    "@id": "https://data.example.com/entitlements#63a196d0-90a3-4bf8-b561-e99fdd258e27",
    "prefLabel": "Junior Pay & Play AcmeCity Resident (Age 16-17)"
  },
  {
    "@type": "Concept",
    "@id": "https://data.example.com/entitlements#106c6991-9c7a-4864-b81b-86148c758bd0",
    "prefLabel": "Junior Pay & Play AcmeCity Resident"
  },
  {
    "@type": "Concept",
    "@id": "https://data.example.com/entitlements#fec2f1d2-47eb-4d30-8a59-c3a6da4d357e",
    "prefLabel": "Junior Concession Pay & Play AcmeCity Resident"
  }
]
}
]
}

```

D10. GET /customer-accounts-rpde

Review Note

This section was proposed on the 9 December 2021 call - introduced as an additional feed for Customer Account updates to reduce the load on the Booking System. It was agreed on the call on 6 January 2022 to require implementation of this and discourage polling GET /customer-accounts/me to achieve the same result.

Gets a feed of updates for Customer Accounts that are related to the Broker .

An update is defined by any material change to the contents of the GET /customer-accounts/me endpoint for a specific Customer Account.

The feed only includes updates for Customer Accounts to which the Broker has been given permission.

Although the Broker could poll the GET /customer-accounts/me endpoint to get updates, this feed-based approach reduces load on the Booking System.

The feed contains only the @id and identifier of the Customer Account that is being updated; for security, updates themselves MUST be retrieved using the GET /customer-accounts/me endpoint.

Implementation note: The client ID is available via the authentication, and e.g. the Open ID Connect grants table can be used to filter the list of Customer Accounts included in this endpoint to only those that have granted permission to that Client ID.

Guidance Note

Note this is "pull" rather than "push", which puts the impetus on the Broker to inform the user of such an update and manage consent around it - e.g. the next time the customer logs into the Broker. It also removes the need for the booking system to inform the user of which Brokers might be updated when customer details are changed.

Authorization

Type: Bearer token obtained via OpenID Connect Client Credentials flow
Required Scope: openactive-customeraccount-updates

Request

GET
/customer-accounts-rpde?afterTimestamp=1521565719&afterId=e11429ea-467f-4270-ab62-e47368996fe8

Response

HTTP/1.1 200 OK

```
{
  "next":
  "https://example.com/api/customer-accounts-rpde?afterTimestamp=1521565719&afterId=e11429ea-467f-4270-ab62-e47368996fe8",
  "items": [
    {
      "state": "updated",
      "kind": "CustomerAccount",
      "id": "fdc14503-275e-46d3-9922-45b986c9f9aa",
      "modified": 1521565719,
      "data": {
        "@context": "https://openactive.io/",
        "@type": "CustomerAccount",
        "@id": "https://id.eg.com/customer-accounts/fdc14503-275e-46d3-9922-45b986c9f9aa",
        "identifier": "fdc14503-275e-46d3-9922-45b986c9f9aa"
      }
    }
  ]
}
```

E. Open data feeds

This section includes details of how opportunities bookable via a Customer Account are described within the open data feeds.

E1. `isOpenBookingWithCustomerAccountAllowed` within the opportunity

The opportunity is marked in the open data with `isOpenBookingWithCustomerAccountAllowed` to indicate that a Customer Account may be used to book that opportunity, noting that it is sometimes the case that not all opportunities in open data are available to book online. If `isOpenBookingWithCustomerAccountAllowed` is not supplied, its value is assumed to be `false`.

The ability to book via a Customer Account is unrelated to `openBookingInAdvance` and `openBookingPrepayment` within the `Offers` in the open data, as these `Offers` indicate capabilities with respect to guest booking.

This data MUST be included in both the open data, and in the responses from C1, C2 and B.

This supports the user journeys within the Broker experience:

1. When clicking "book" on an opportunity, the user may be prompted to authenticate with their Customer Account (in the case that Open Booking is not permitted).
2. Following C1 a user may be prompted to authenticate with their Customer Account

Open data extract

```
{
  "@type": "SessionSeries"
  ...
  "isOpenBookingWithCustomerAccountAllowed": false
}
```

E2. `customerAccountBookingRestriction` within the opportunity

Due to the sheer range and complexity of pay-monthly entitlements configurable in a Booking System, this specification does not support the codification of restricted opportunities (i.e. it is not possible to determine whether an opportunity is only bookable using a specific entitlements, if it is not available to all members).

To ensure clarity is provided to Customers at the browse stage, the opportunity is marked in the open data with a free text property `customerAccountBookingRestriction`. The Broker MUST display this text to the Customer at the browse stage, before they proceed to authenticate the Customer Account to book an opportunity, and the Customer may then use this information to determine whether they are qualified to continue booking the specific session.

Open data extract

```
{
  "@type": "SessionSeries"
  ...
  "customerAccountBookingRestriction": [
    "Gold members only",
    "Gym induction required"
  ]
}
```

If a Customer attempts to proceed to book an opportunity they are not permitted to attend, they will receive an error as per Section F4.

E3. eligibleEntitlementType within the Offer

Offers in open data can be marked as requiring an entitlement type via `eligibleEntitlementType`. Note that the same Offer may be applicable to multiple entitlement types, and the Customer MUST have **at least one** matching entitlement type to qualify for the Offer.

It is important to note that Offers without `eligibleEntitlementType` are also provided by the Seller to facilitate guest bookings by those Customers who are not part of the membership scheme.

As entitlements are generally related to a membership scheme, they are expected to only apply to (and be included in open data for) locations of the Seller where that membership scheme operates.

All Offers MUST be bookable by guest checkout, where they are used as part of the booking flow (and hence support for Customer Accounts is optional for the Broker). For Customer Account account bookings the Offers are not used as part of the booking flow (see Section F), and hence the "bookability" of an opportunity by a Customer Account is described in Section E1 and E2.

Pricing displayed to the Customer by the Broker MUST use the Offer with the lowest price which contains an `eligibleEntitlementType` that matches the Customer Account. Where a matching `eligibleEntitlementType` is not found, the appropriate default Offer (considering the Offer's `ageRestriction` where appropriate) MUST be displayed.

Prices MUST be marked as "indicative" in the Broker at the browse stage where either:

- a) There is more than one entitlement on the Customer Account
- b) The entitlement on the Customer Account is not part of the Broker's membership scheme
- c) `"hasHiddenEntitlements": true` is set in the CustomerAccount
- d) An Offer for the Customer Account entitlement is not found in the open data feed (and hence the default Offer is displayed).

Note that the `OfferOverride` subclass of `Offer` does not include `eligibleEntitlementType`.

Including entitlement pricing in open data feeds

```
"offers": [
```

```

{
  "@type": "Offer",
  "@id": "https://example.com/api/identifiers/facility-uses/1489/slots/14882#/offers/0",
  "identifier": "OX-AD",
  "name": "Adult",
  "price": 3.3,
  "priceCurrency": "GBP"
},
{
  "@type": "Offer",
  "@id": "https://example.com/api/identifiers/facility-uses/1489/slots/14882#/offers/1",
  "identifier": "E/OX-AD",
  "name": "Adult",
  "price": 3.3,
  "priceCurrency": "GBP",
  "eligibleEntitlementType": [
    {
      "id": "https://data.mcractive.com/openactive/entitlement-list#12345",
      "type": "Concept",
      "prefLabel": "MCRactive Adult Non-resident",
      "inScheme": "https://data.mcractive.com/openactive/entitlement-list"
    }
  ]
},
{
  "@type": "Offer",
  "@id": "https://example.com/api/identifiers/facility-uses/1489/slots/14882#/offers/2",
  "identifier": "E/MCR2",
  "name": "MCRactive Junior Resident",
  "price": 2.1,
  "priceCurrency": "GBP",
  "ageRestriction": {
    "@type": "QuantitativeValue",
    "minValue": 50
  },
  "eligibleEntitlementType": [
    {
      "id": "https://data.mcractive.com/openactive/entitlement-list#5678",
      "type": "Concept",
      "prefLabel": "MCRactive Junior Resident",
      "inScheme": "https://data.mcractive.com/openactive/entitlement-list"
    }
  ]
}
],

```

E4. LateCancellationPolicy within termsOfService

GitHub Discussion

A GitHub discussion relates to this section:

<https://github.com/openactive/customer-accounts/issues/4>

Review Note

This section was added based on feedback from Everyone Active on 10 January 2022

It is possible for a customer to request the cancellation of a Customer Account booking outside of the cancellation window. Before such a cancellation is requested, the customer MUST be notified of any late cancellation policy that may apply.

To ensure clarity is provided to Customers before they proceed with the cancellation, the policy is summarised in the free text property `abstract` within a `LateCancellationPolicy`. The Broker MUST display this text to the Customer before they proceed to request a cancellation outside of

the cancellation window, and the Customer may then use this information to determine whether they wish to proceed with the cancellation.

Open data extract

```
"organizer": {
  "@type": "Organization"
  ...
  "termsOfService": [
    {
      "@type": "LateCancellationPolicy",
      "name": "Late Cancellation Policy",
      "url": "https://example.com/late-cancellation-policy",
      "abstract": "Late cancellations may incur a penalty if the space is not relet"
    }
  ]
}
```

If a Customer is not permitted to make a late cancellation, they MUST receive an error from the Cancellation request with the reason clearly articulated.

F. Open Booking API behaviour

This section includes details of how to make bookings for a Customer Account via the Open Booking API flow.

F1. Support for brokerRole

Customer Account bookings may only be made in either `AgentBroker` or `NoBroker` modes. There is no scenario where `ResellerBroker` is viable as the Customer Account exists in the Booking System and MUST be used to make the booking.

C1, C2 and B will always return the following error message if `brokerRole` is `ResellerBroker`:

Authorization
Type: Bearer token obtained via Authorization Code flow Required Scope: <code>openactive-openbooking-customeraccount</code>
C1, C2 and B request extract
<code>"brokerRole": "https://openactive.io/ResellerBroker"</code>
C1, C2 and B response
HTTP/1.1 409 Conflict { "@context": "https://openactive.io/", "@type": "UnsupportedBrokerRole", "name": "The specified Broker Role is not supported", "description": "ResellerBroker cannot be used for Customer Account bookings" }

F2. customer within the Open Booking API flows for Customer Accounts

When the Authorization header contains a Bearer token for a Customer Account, C1, C2 and B behave differently.

Bookings are made by using the `hasAccount` property, referencing the `@id` of the `CustomerAccount`, which can be obtained from the `https://openactive.io/customerAccountId` custom claim within the Open ID Connect `id_token`. Note this MUST be made available in the `id_token` to allow bookings for Brokers that do not have permission to use the `openactive-customeraccount-read` scope and simply want to allow the Customer to access their discounts.

Review Note

Although [OpenID scopes](#) could be used to give permission to access specific properties within the `Person`, this additional complexity is unlikely to be warranted, and so has not been included.

The `Person` is expanded in the response to only include the following key properties: `email`, `telephone`, `givenName`, `familyName`, `emergencyContact`. These are important for Brokers that do not have permission to use the `openactive-customeraccount-read` to be able to accurately represent the booking that has been made.

Review Note

Note to remove the need for the Broker to know whether the Customer Account type should be a `Person` or an `Organization`, we simply don't include `"customer"` in the requests and infer it from the authentication. This requires less work for the Broker (note they will still need the `customerAccountId` if they're specifying attendees, attendees MUST be a `Person`).

The advantage of this approach is that it reduces the number of error conditions. The disadvantage is that it removes an additional verification of the Broker's correct implementation, however such a double-check may be excessive.

C1 will always return the following error message:

Authorization

Type: Bearer token obtained via Authorization Code flow
Required Scope: `openactive-openbooking-customeraccount`

C1 response

HTTP/1.1 409 Conflict

```
{
  "@context": "https://openactive.io/",
  "@type": "UnexpectedCustomerAuthenticationError",
  "name": "The attempt request is invalid when a Customer Account Bearer token is provided in the Authorization header",
  "description": "C1 cannot be called when using a Customer-specific access token."
}
```

For C2 and B, the request MUST include `hasAccount` within the `customer` that MUST match the Bearer token

Authorization

Type: Bearer token obtained via Client Credentials flow

C2 and B request extract

(without `"customer"`)

C2 and B request response

HTTP/1.1 409 Conflict

```
{
  "@context": "https://openactive.io/",
  "@type": "CustomerAuthenticationExpectedError",
  "name": "The attempted request is invalid when a Client Credentials flow Bearer token is provided in the Authorization header"
}
```

Authorization
Type: Bearer token obtained via Authorization Code flow Required Scope: openactive-openbooking-customeraccount
C2 and B request extract
<pre>"customer": { "@type": "Person", ... }</pre>
C2 and B request response
<pre>HTTP/1.1 409 Conflict { "@context": "https://openactive.io/", "@type": "CustomerAuthenticationUnexpectedError", "name": "Customer must not be specified when a Customer Account Bearer token is provided in the Authorization header" }</pre>

Authorization
Type: Bearer token obtained via Authorization Code flow Required Scope: openactive-openbooking-customeraccount
C2 and B request extract
(without "customer")
C2 and B response extract
<pre>"customer": { "@type": "Person", "hasAccount": { "@type": "CustomerAccount", "@id": "https://eg.com/customer-accounts/fdc14503-275e-46d3-9922-45b986c9f9aa", "identifier": "fdc14503-275e-46d3-9922-45b986c9f9aa", "accountNumber": "CA00000123" }, "email": "alexjones@example.com", "telephone": "020 811 8055", "givenName": "Alex", "familyName": "Jones", "emergencyContact": { "@type": "Person", "name": "Ralph Jones", "telephone": "020 811 8055" } }</pre>

F3. attendee within the Open Booking API flows for Customer Accounts

Any OrderItem **without** a hasAccount property within the attendee:

- Is considered to be a **guest attendee** booked by the authenticated Customer Account (or a `guest customer` if specified)
- MUST include an `acceptedOffer` in the request and response to indicate the appropriate price

Any `OrderItem` **with** a `hasAccount` property within the `attendee` referencing the `@id` of a `CustomerAccount`:

- Is considered to be booked under the `CustomerAccount` specified by the `attendee`, and this `CustomerAccount` MUST have granted permission for the "openactive-openbooking-on-behalf-of" scope to the Broker if it is not the same as the `CustomerAccount` associated with the provided access token.
- MUST NOT include an `acceptedOffer` in the request `OrderItem`
- MUST include an `acceptedOffer` in the response `OrderItem` that takes into account any applicable discount pricing based on entitlements of that `CustomerAccount`. The response `acceptedOffer` does not include an `@id`, but MUST include `allowCustomerCancellationFullRefund` and `latestCancellationBeforeStartDate` and other properties of the `offer` if supported by the booking system.
- MUST NOT include personal data in the `attendee` within the response. This is for security of personal data when booking on-behalf-of, and to reduce data duplication in the response if the authenticated `CustomerAccount` is booking for themselves (as the `customer` in the response will already include the personal data for the authenticated `CustomerAccount`).

Authorization
Type: Bearer token obtained via Authorization Code flow Required Scope: openactive-openbooking-customeraccount
C2 and B request extract
<pre>"orderedItem": [{ "@type": "OrderItem", "position": 0, "attendee": { "@type": "Person", "hasAccount": "https://eg.com/customer-accounts/fdc14503-275e-46d3-9922-45b986c9f9aa" }, "orderedItem": "https://example.com/events/452/subEvents/132" }]</pre>
C2 and B response extract
<pre>"orderedItem": [{ "@type": "OrderItem", "position": 0, "attendee": { "@type": "Person", "hasAccount": { "@type": "CustomerAccount", "@id": "https://eg.com/customer-accounts/fdc14503-275e-46d3-9922-45b986c9f9aa", "identifier": "fdc14503-275e-46d3-9922-45b986c9f9aa" } } }]</pre>

```

    },
    "acceptedOffer": {
      "@type": "Offer",
      "price": 10,
      "priceCurrency": "GBP",
      "validFromBeforeStartDate": "P6D",
      "allowCustomerCancellationFullRefund": true,
      "latestCancellationBeforeStartDate": "P1D"
    },
    "orderedItem": {
      "@type": "ScheduledSession",
      "@id": "https://example.com/events/452/subEvents/132",
      "identifier": 123,
      "eventStatus": "https://schema.org/EventScheduled",
      "maximumAttendeeCapacity": 30,
      "remainingAttendeeCapacity": 20,
      "startDate": "2018-10-30T11:00:00Z",
      ...
    }
  }
]

```

F4. error within the Open Booking API flows for Customer Accounts

For opportunities that are not permitted to be booked by a specific `CustomerAccount` (where such restrictions SHOULD be indicated by `customerAccountBookingRestriction` at the browse stage), C1 and C2 MUST return a specific error, with a customer-facing description, for example:

C2 and B response extract

```

"error": [
  {
    "@type": "BookingNotPermittedByCustomerAccountError",
    "name": "This Customer Account is not permitted to make this booking at this time",
    "description": "This opportunity is not permitted to be booked using your membership"
  }
]

```

For scenarios where a booking is not permitted due to the current state of the `CustomerAccount`, such issues MUST be indicated by `outstandingAction` within the `CustomerAccount`. This allows the Broker to display the `outstandingActions` to the Customer, to aid their resolution. Where an `outstandingAction` blocks a booking for the authenticated `CustomerAccount`, C1 and C2 MUST return a specific error, with a customer-facing description, for example:

C2 and B response extract

```

"error": [

```

```
{
  "@type": "OutstandingActionOnCustomerAccountError",
  "name": "An outstanding action on this account is preventing booking of this
opportunity",
  "description": "A gym induction is required to make the booking"
}
```

When the `CustomerAccount` specified by the attendee is different to the `CustomerAccount` related to the authentication token (i.e. an "on-behalf-of" booking), and has an `outstandingAction` the description specified in the above error MUST be generic to ensure the privacy of the attendee.

C2 and B response extract

```
"error": [
  {
    "@type": "OutstandingActionOnCustomerAccountError",
    "name": "An outstanding action on this account is preventing booking of this
opportunity",
    "description": "A booking could not be made on behalf of this account at this time"
  }
]
```

When the `CustomerAccount` related to the authentication token has an `outstandingAction`, the booking MAY fail (depending on the business rules of the Seller), with an `OutstandingActionOnCustomerAccountError` response as shown below:

C2 and B response

```
{
  "@type": "OutstandingActionOnCustomerAccountError",
  "name": "An outstanding action on this account is preventing booking of this opportunity",
  "description": "A gym induction is required to make the booking"
}
```

F5. Communication and interacting with the booking within the Booking System

This section is copied from the summary of the [previous discussion in #120](#), with terms updated to match the rest of this document.

The Broker communicates all updates to the Customer for bookings made via the Broker, even if amendments are made from within the Booking System. Bookings made via a Broker can only be updated in two specific ways.

This proposal attempts to satisfy all stakeholders minimally: minimising the implementation overhead while also ensuring the Customer experience is consistent.

To meet Customer expectations, all bookings made through the Open Booking API via a Customer Account SHOULD be visible to the Customer within their account within the Booking System.

Additionally, the Booking System MAY allow the Customer to perform the following operations from within the Booking System on bookings made originally through the Open Booking API via a Customer Account:

- Customer can cancel within the Booking System, which triggers "[Customer requested cancellation](#)" as per the current Open Booking API specification, and hence a full refund.
- Customer can reschedule within the Booking System, which triggers "[Replacement](#)" as per the current Open Booking API specification.

In both cases, as per the rules in the Open Booking API specification, the Booking System MUST NOT send notifications to the customer directly, and the Broker MUST process any refund due and send immediate notifications to the Customer. This makes the communication consistent within the scope of each booking: if a Customer booked through the Change4Life app, all further updates would come through the same app, which keeps the app in a consistent state.

The Booking System MUST NOT allow the user to perform any other actions on the booking from within their user experience, as these are not supported by the Open Booking API, and therefore will not be supported by the Broker, and would create an inconsistent state. The user interface of the booking system SHOULD reflect this, and reference the name property of the broker for that booking, such that it communicates "Booked via Broker X".

This should be in keeping with the Customer's expectations, and the communication from the Broker should be expected if bookings are labelled in the interface.

F6. Open Booking API endpoints for Customer Accounts

All endpoints that require the scope `openactive-openbooking-customeraccount` also require a Bearer token obtained via Authorization Code flow for the specific `CustomerAccount` that was used to create the Order associated with the requested UUID.

Hence if a Customer revokes access to their Customer Account, the Broker is no longer able to access or make changes to its bookings.

To ensure refunds can still be processed in the event of access being revoked, such Orders MUST continue to be visible in the Orders feed.

Authorization
Type: Bearer token NOT obtained via Authorization Code flow Required Scope: <code>openactive-openbooking-customeraccount</code>
Response


```
HTTP/1.1 403 Forbidden
```

```
{
  "@context": "https://openactive.io/",
  "@type": "InvalidCustomerAccountAuthenticationError",
  "name": "This request requires the appropriate Customer Account Bearer token to be provided in the Authorization header",
  "description": "This Order requires a Customer-specific access token to be used."
}
```

F7. Customer Requested Cancellation from within the Booking System for Customer Accounts

Review Note

This section was added based on feedback from Everyone Active on 30 January 2022

The Customer may cancel a booking made on their Customer Account via the Booking System directly, if this functionality is supported within the Booking System. Such cancelled `OrderItems` will have `orderItemStatus` set to `https://openactive.io/CustomerCancelled` in the Orders feed, in order that refunds may be processed directly .

F8. Customer Requested Cancellation from within the Broker for Customer Accounts

Review Note

This section was added based on feedback from Everyone Active on 30 January 2022

A customer may request the cancellation of any item booked via their Customer Account at any time before the `endDate` of the opportunity. Before a cancellation request is made outside of the cancellation window the customer MUST first be notified of any late cancellation policy that may apply - see Section E4.

F9. Terms and conditions

GitHub Discussion

A GitHub discussion relates to this section:

<https://github.com/openactive/open-booking-api/issues/231>

The `termsOfService` of the Seller and Booking System that are included in the feed are intended to be displayed during a Broker checkout flow, and are not applicable to [Creating a New](#)

[Customer Account](#). Terms and conditions relating to Creating a New Customer Account are displayed as part of the Booking System's OpenID Connect flow (see [Section B2](#)).

The Broker MUST NOT require explicit consent for `termsOfService` of the Seller and Booking System for bookings made via a Customer Account, regardless of the value of `requiresExplicitConsent`, as the Customer has already accepted the Seller and Booking System terms when they created the Customer Account, as part of the flow outlined in [Section B2](#).

For Customer Account bookings, the Broker SHOULD indicate within any checkout flow that the booking is being made within the authenticated Customer Account, and that `termsOfService` will apply.

For guest bookings, as stated in the [Open Booking API specification](#), consent for `termsOfService` is not stored in the booking system, and instead MUST be captured, where required by `requiresExplicitConsent`, in order for the booking to be made. `dateModified` is provided so that the same Customer is only prompted for explicit consent if they have been updated since the last time they booked.

Open data extract - within Organization

```
"termsOfService": [
  {
    "@type": "Terms",
    "name": "Health Questionnaire",
    "url": "https://www.everyoneactive.com/legal-policies/health-commitment-statement/",
    "dateModified": "2020-02-16T20:31:13Z",
    "requiresExplicitConsent": true
  },
  {
    "@type": "Terms",
    "name": "Terms and Conditions",
    "url": "https://www.everyoneactive.com/legal-policies/terms-and-conditions-online/",
    "dateModified": "2020-02-16T20:31:13Z",
    "requiresExplicitConsent": true
  },
  {
    "@type": "Terms",
    "name": "Privacy Policy",
    "url": "https://www.everyoneactive.com/legal-policies/privacy/",
    "dateModified": "2020-02-16T20:31:13Z",
    "requiresExplicitConsent": true
  }
]
```

Considerations for guidance

Example rendering of terms requiring explicit consent:

I have read and agree to the [health questionnaire](#)

Yes No

I have read and agree to the [terms and conditions](#) and [privacy policy](#)

Yes No

The following should be added to guidance associated with this specification:

Activity-specific prerequisite videos and waivers

Additional Details capture MAY be used for activities that require inductions or waivers, when booking as a guest or with a Customer Account that has not yet watched the induction or accepted the waiver.

C1 extract

```
{
  "@type": "OrderItem",
  ...
  "orderItemIntakeForm": [
    {
      "@type": "BooleanFormFieldSpecification",
      "@id": "https://example.com/photoconsent",
      "name": "Online Gym Induction",
      "description": "I confirm that I have watched and understood how to use the equipment and will
only use equipment I have been shown how to use",
      "url": "https://www.everyoneactive.com/open-booking-gym-induction",
    }
  ]
}
```

I confirm that I have watched and understood how to use the equipment and will only use equipment I have been shown how to use.

SUBMIT

G. OpenID Connect

This section includes details of the Open ID Connect authentication used within this specification.

G1. OpenID Connect Parameters

This section describes the various OpenID Connect Parameters that are defined within this specification

`openactive_flow_type`

- `seller` - the existing multi-seller authentication flow defined in the Open Booking API
- `customer` - the Customer Account creation/link flow defined within this specification
- `customer_delegation` - the Dependent Account flow defined within this specification

`allow_signup`

- `true` - Allow new account creation within the `customer` or `customer_delegation` flow
- `false` (default) - Only login / link existing account is permitted within the `customer` / `customer_delegate` flow

`screen_hint` (as defined by [Auth0](#))

- `signup` - Default to the signup screen that is enabled via `allow_signup`
- Other values are left intentionally undefined

`login_hint`

- Optional. The email address or username that SHOULD be pre-populated within the flow.

`delegation_login_hint`

- Optional. The email address or username that SHOULD be pre-populated for a dependent account within the flow

`delegation_name_hint`

- Required for `openactive_flow_type=customer_delegation`. The name used for signposting throughout the dependent account flow.

`seller_id`

- The `@id` of the Seller for which permission is being requested. Required for `openactive_flow_type=customer` and `openactive_flow_type=customer_delegation` within multi-seller systems that have per-seller Customer Accounts
("authenticationBasis":
"https://openactive.io/MultipleSellerAuthentication").

G2. Logout

Implementations MUST support https://openid.net/specs/openid-connect-rpinitiated-1_0.html/ / https://openid.net/specs/openid-connect-session-1_0.html#RedirectionAfterLogout which allows a seamless logout of the Booking System to occur.

G3. Permissible authentication configurations

The table below summarises the permissible configurations for OpenID Connect, which are specified by `authenticationBasis` within the dataset site.

Access Token Origin	SingleSellerAuthentication	BookingSystemAuthentication	MultipleSellerAuthentication
Client Credentials Flow	Orders feed Guest booking	Orders feed	Orders feed
Authorization Code Flow <code>openactive_flow_type=seller</code>	N/A	Guest booking	Guest booking
Authorization Code Flow <code>openactive_flow_type=customer*</code> Without "sellerId" parameter	CustomerAccount booking	CustomerAccount booking	
Authorization Code Flow <code>openactive_flow_type=customer*</code> With "sellerId" parameter			CustomerAccount booking

Systems with a single Customer Account and a single seller

Dataset contains "authenticationBasis":
"`https://openactive.io/SingleSellerAuthentication`"

In this scenario the Customer has a single e.g. "seller.com" account that they use to make bookings for that seller within the Booking System.

The UI of the Broker can be informed by the [publisher](#) property within the dataset site.

The Customer Account authentication tokens are retrieved for the Customer Account of the Seller via an OpenID Connect Flow in either `openactive_flow_type=customer` or `openactive_flow_type=customer_delegation` case.

For guest `customer` bookings, the Broker MUST use the access tokens retrieved via the client credentials flow.

Systems with a single booking system Customer Account for multiple sellers

Dataset contains "authenticationBasis":
"https://openactive.io/BookingSystemAuthentication"

In this scenario the Customer has a single e.g. "bookingsystem.com" account that they use to make bookings for all Sellers within the Booking System.

The UI of the Broker can be informed by the [bookingService](#) property within the dataset site.

The Customer Account authentication tokens are retrieved for the Customer Account of the Booking System via an OpenID Connect Flow in either `openactive_flow_type=customer` or `openactive_flow_type=customer_delegation` case. This same token is used to book for the Customer Account with any Seller.

However, the Customer is only permitted to book via the Customer Account if the Broker has been given permission to the Seller via a separate seller-specific OpenID Connect authorization flow using `openactive_flow_type=seller`. This seller-specific token MUST be used for guest bookings.

For guest `customer` bookings, the Broker MUST use the access tokens retrieved via the multi-seller authentication flow (`openactive_flow_type=seller`) for this Seller.

Review Note

The following section was removed, as it could be easily implemented in the future, however perhaps creates unnecessary complexity for initial implementation:

As a Customer can use the same Customer Account within the Booking System to book with multiple Sellers, they MUST give permission to the Broker separately for each Seller. This likely means that the Customer will stay logged in on the Booking System, and simply need to visit the "Consent" screen multiple times (or the "Consent" screen may list Sellers with which they have an account, similar to the way GitHub allows users to select permissions for multiple Organizations in its own OpenID Connect flow).

Systems with per-seller Customer Account for multiple sellers

Dataset contains "authenticationBasis":
"https://openactive.io/MultipleSellerAuthentication"

In this less common scenario the Customer creates a separate "seller.com" account within the Booking System for each Seller they wish to book through.

An additional parameter "sellerId" and `openactive_flow_type=customer` or `openactive_flow_type=customer_delegation` is specified on the `/authorize` endpoint of OpenID Connect. This MUST be equal to the Seller @id, and the consent page displayed MUST be specific to the Seller, Customer Account *and* Broker.

The UI of the Broker can be informed by the `organizer` property within the open data feeds, where the Seller details are visible.

This page is only displayed successfully when the Broker has already been granted permission by the Seller to accept bookings (via the seller-specific OpenID Connect authorization flow using `openactive_flow_type=seller`), otherwise the page MUST display an error message.

For guest `customer` bookings, the Broker MUST use the access tokens retrieved via the multi-seller authentication flow (`openactive_flow_type=seller`) for this Seller.

G4. Permissible authentication flows

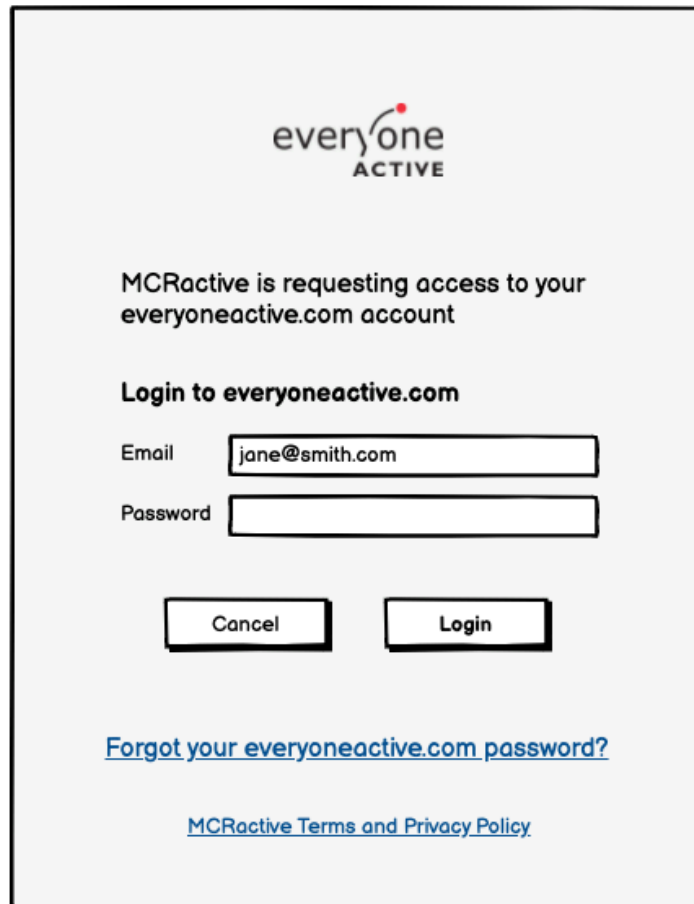
`openactive_flow_type=seller` - the existing multi-seller authentication flow

This flow is already defined [within the Open Booking API](#).

`openactive_flow_type=customer` - the Customer Account creation/link flow

Linking existing accounts is available via the default flow (or "allow_signup=false"):


Booking System



The screenshot shows a login form for everyoneactive.com. At the top is the logo for everyone ACTIVE. Below it, a message states: "MCRactive is requesting access to your everyoneactive.com account". The form title is "Login to everyoneactive.com". There are two input fields: "Email" with the value "jane@smith.com" and "Password" which is empty. Below the fields are two buttons: "Cancel" and "Login". At the bottom, there are two links: "[Forgot your everyoneactive.com password?](#)" and "[MCRactive Terms and Privacy Policy](#)".

Creating new accounts is possible via the parameters "allow_signup=true" **with** "screen_hint=signup":

Booking System



MCRActive is requesting that you create a new account with everyoneactive.com

Sign up to everyoneactive.com

Email

Use MCRActive account details to create a new everyoneactive.com account ⓘ

I have read and agree to the [health questionnaire](#)

[Already have an everyoneactive.com account?](#)

[Everyone Active Terms and Privacy Policy](#)

Note the "Already have an everyoneactive.com account?" link above opens the login screen from the default flow.

Finally the parameters "allow_signup=true" **without** "screen_hint=signup", simply adds a "Sign up for a new account" link to the login screen of the default flow, which links to the screen above:

Booking System

everyone
ACTIVE

MCRactive is requesting access to your everyoneactive.com account

Login to everyoneactive.com

Email

Password

[Reset your everyoneactive.com password](#)

[Sign up for a new everyoneactive.com account](#)

[MCRactive Terms and Privacy Policy](#)

The above may be used for the scenario when the Customer has the option to login OR create a new account, but their email address has been found to match an existing account, so they are encouraged to login with this one (note they are not forced to, as e.g. perhaps another family member has already created a Booking System account with their email address, so they may choose to create a new account here instead).

openactive_flow_type=customer_delegation - the Dependent Account flow

This flow allows the linking of existing primary and dependent accounts, and the creation of new dependent accounts. It is intended to be used by a Broker after a primary account has already been linked/created using `openactive_flow_type=customer`, to allow the linking of other related accounts.

The flow allows the Customer to manage their dependent accounts once they have authenticated, and then select one to link to the Broker. It also allows the Customer to choose to link themselves.

It is only possible to create a dependent account via this flow, rather than creating a new primary account as in `openactive_flow_type=customer`. This removes potential ambiguity from the user journey. This also encourages best practice of the user only authenticating as themselves rather than requiring them to share passwords with others (which allows multi-factor authentication to

be supported in future if required), while also providing compatibility with legacy accounts that require the current user to use another user's credentials.

Notes relating to the illustrative flow below:

- The "Create New Account" option is only visible if "allow_signup=true"
- The email address field on the "Create New Account" screen is optional. For systems that have a unique constraint on the email address used for each account, the email field in the "Create New Account" screen can simply be omitted.
- It is recommended that the Booking System persists the authenticated session in a cookie so that the current user does not need to log in to the Booking System again, which means the second screen below will be skipped in most cases.
- The Booking System SHOULD provide a mechanism within the OpenID Connect flow to remove erroneously created Dependent Accounts found in the list (as illustrated by the trash icons), to encourage the Customer to accurately maintain the Booking System's records.

The image displays three sequential screenshots of a user interface flow:

- Broker Registration:** A screen titled "Registration" under the "Broker" header. It shows a list of "Linked accounts" with entries for Jane Smith, John Smith, Little Jonny (highlighted), Little Alexi, and Little Sam. Below the list, there is a section for "BETTER everyone ACTIVE" with a date "Linked on 12/08/2021" and an "Unlink" button. A "Next" button is at the bottom.
- Booking System Login:** A screen titled "Booking System" with the "everyone ACTIVE" logo. It prompts the user to log in with their own account to link an account for "Little Jonny". It includes fields for "Email" (jane@smith.com) and "Password", along with "Cancel" and "Login" buttons. A link for "Forgot your everyoneactive.com password?" is at the bottom.
- Booking System Link Account:** A screen titled "Booking System" with the "everyone ACTIVE" logo and the user's email "jane@smith.com" and "Log Out" link. It prompts the user to "Choose the account belonging to Little Jonny below to link it with MCRactive:". It lists accounts: Jane Smith (you), Little Jonny (with a trash icon), Little Alexi (with a trash icon), Create New Account, and Link Existing Account. Each entry has a "Link ->" button. A link for "MCRactive Terms and Privacy Policy" is at the bottom.

Booking System | Create New Account

everyone ACTIVE jane@smith.com | [Log Out](#)

everyone ACTIVE

This will create a new account for **Little Jonny** within the account **Jane Smith** using the following email address:

Email

Use MCRactive account details for **Little Jonny** to create a new everyoneactive.com account ⓘ

I have read and agree to the [health questionnaire](#) on behalf of this user

[Everyone Active Terms and Privacy Policy](#)

Booking System | Link Existing Account

everyone ACTIVE jane@smith.com | [Log Out](#)

everyone ACTIVE

If **Little Jonny** has an existing Everyone Active account, please log in using that email address and password to link this account to **Jane Smith**.

Email

Password

[Forgot the password for Little Jonny?](#)

For compatibility with Booking Systems that require an email address for each account, an additional parameter `delegation_email_hint` is provided, which can optionally be used by the Booking System to inform the user of the email address that is being used to create the account or authenticate with an existing account. For such systems this may default to the authenticated user's email address, if the `delegation_email_hint` is not provided.

Additionally a parameter of `delegation_name_hint` allows the UI of the Booking System to display the name of the Broker account to be linked, to help signpost the user appropriately.

When a user is created via this flow, the email address is optionally set via initialisation, and the email field is not required.

The Booking System MUST provide a means to display which Brokers a Dependent Account is connected to, such that if the Dependent Account was to be managed by a new Customer Account, the new Customer Account may view a list of currently connected Brokers and choose to disconnect them. Changes to the management relationships of a Dependent Account within the Booking System SHOULD NOT impact existing Broker connections.

G5. Consent screen

After the user completes all of the Booking System screens within any of the flows specified in [G4](#), they MUST be prompted with a consent screen to provide explicit consent for the Broker to access their account, unless they have previously provided consent and ticked "Remember My Decision".

Such consent screens are often generated automatically by OpenID Connect libraries.

An example of such a screen is below:

BETTER Not jane@smith.com? ▾

MCRactive is requesting your permission to access the following features of your better.org.uk account:

- Access the unique identifier of your better.org.uk account
- Make bookings using your account
- Allow others to book on your behalf
- Get updates for cancellations
- Access and update your email address, name, phone number, gender, address and date of birth
- Manage discounts you are entitled to
- Add a barcode to your account
- Offline Access - Allow the above even when you are not logged in

Remember My Decision

YES, ALLOW **NO, DO NOT ALLOW**

G6. Endpoint Scopes

This specification defines and utilises the following OpenID Connect scopes, which allow for granular control over a Broker's use of the Customer Accounts API.

Scope	Description
<code>openactive-customeraccount-claims</code>	Claims relating to the Customer Account
<code>openactive-customeraccount-create</code>	Operations that create a Customer Account.

<code>openactive-customeraccount-query</code>	Operations that query existing Customer Accounts, without requiring Customer Account authorisation.
<code>openactive-customeraccount-read</code>	Operations that read an existing Customer Account, using Customer Account authorisation.
<code>openactive-customeraccount-modify</code>	Operations that modify an existing Customer Account, using Customer Account authorisation.
<code>openactive-customeraccount-updates</code>	Operations that retrieve updates for existing Customer Accounts, without requiring Customer Account authorisation.
<code>openactive-openbooking-customeraccount</code>	Operations defined in the Open Booking API, executed with an authenticated Customer Account.
<code>openactive-openbooking-on-behalf-of</code>	Operations defined in the Open Booking API, where any attendee specified is a Customer Account other than the authenticated Customer Account.

For cases where the required scope has not been obtained, an `AccessDeniedError` MUST be returned:

Response
<pre>HTTP/1.1 403 Forbidden { "@context": "https://openactive.io/", "@type": "AccessDeniedError", "name": "This Broker does not have permission to perform this operation", "description": "The provided access token does not include the required scope openactive-openbooking-customeraccount." }</pre>

G7. ID Token claims

The ID Token is designed to be read by the Broker to give them information about the Customer Account that has just been authenticated. This allows the Broker to store the Access Token and Refresh Token against the correct `customerAccountId` and `sellerId` in their database, so that they can use these when booking the Seller's opportunities on behalf of the Customer Account.

The `openactive-customeraccount-claims` scope MUST include the Claims listed below in the ID Token.

The following custom claims are for use by the Broker, and must conform to the custom claim names specified below. The custom claim names are collision-resistant in accordance with [the OIDC specification](#).

Custom claim	Description	Exactly matches
--------------	-------------	-----------------

https://openactive.io/customerAccountId	The JSON-LD ID of the Customer Account.	@id of CustomerAccount
https://openactive.io/isPendingInitialization	Whether the authenticated Customer Account has been initialised	N/A