EDX

Event Data Exchange

Version 0.8

Working Draft 17 November 2009

This version:
http://www.trybooking.com/html/downloads/EDX/edx.pdf
Latest Editor's draft:
http://www.trybooking.com/html/downloads/EDX/edx.odt (openoffice.org)
Editors:
Grant Dunoon, Bookwana Pty. Ltd, grant@trybooking.com

Table of Contents

Abstract

This specification defines the Event Data Exchange (EDX), version 1.0.

Introduction

The EDX core components technical specification presents a methodology for developing a common set of semantic building blocks to represent the general types of Event and Ticketing data in use today. It also provides for the creation of new business vocabularies and restructuring of existing business vocabularies.

This EDX document describes and specifies a new approach to the well-understood problem of the lack of information interoperability between applications in the Event Ticketing arena. Traditionally, standards for the exchange of business data have been focused on static message definitions or proprietary formats that have not enabled a sufficient degree of interoperability or flexibility. EDX addresses the need for a more flexible and interoperable way of standardising Event and Ticketing semantics.

Revisions

Revised by:	Date	Version
Grant Dunoon – Bookwana Pty. Ltd.	14 October 2009	0.7
Grant Dunoon – Bookwana Pty. Ltd.	17 November 2009	0.8

Licence

Typographic Conventions

This section is non-normative.

This section defines the following typographic conventions used by this specification:

- Defined terms appear as this sample defined term. Such terms are referenced as sample defined term, providing a link back to the term definition.
- Words that denote a conformance clause or testable assertion use keywords from [RFC2119]: must, must not, required, should, should not, recommended, may and optional.
- Words in italics denote a formal definition given in an external specification.

The following are a few samples of the usage of the typographic conventions in this document:

This is an example. Examples are used to explain concepts or demonstrate how to use a feature. Examples are non-normative.

ISSUE: This is an issue. It implies something that Working Group is trying to fix. Eventually, all these will disappear from the spec. Issues are non-normative.

Note: This is a note, it usually contains useful supplementary information in a non-normative form.

Definitions

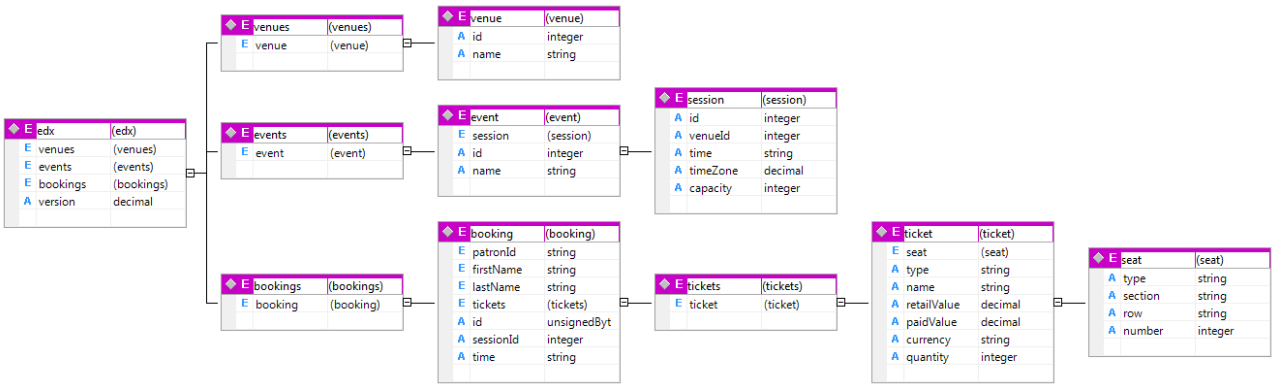
Attachment	See end of this document.
Callback message	A message transmission returned by some operations some time after the operation was invoked (asynchronously).
Client	System requesting the EDX
Document	The XML file of the EDX.
Document hash	A condensed representation of a document intended to protect document integrity, calculated according to the SHA 256 algorithm.
Messages	A message is an XML document that is a well-formed XML data structure with a single root element that is transmitted between computers and is valid as defined by one of the defined message structure schemas in this document.
Server	System sending the EDX
Synchronous response	A message transmission returned immediately (synchronously) as the result of an operation. Every operation has a synchronous response.

Changes

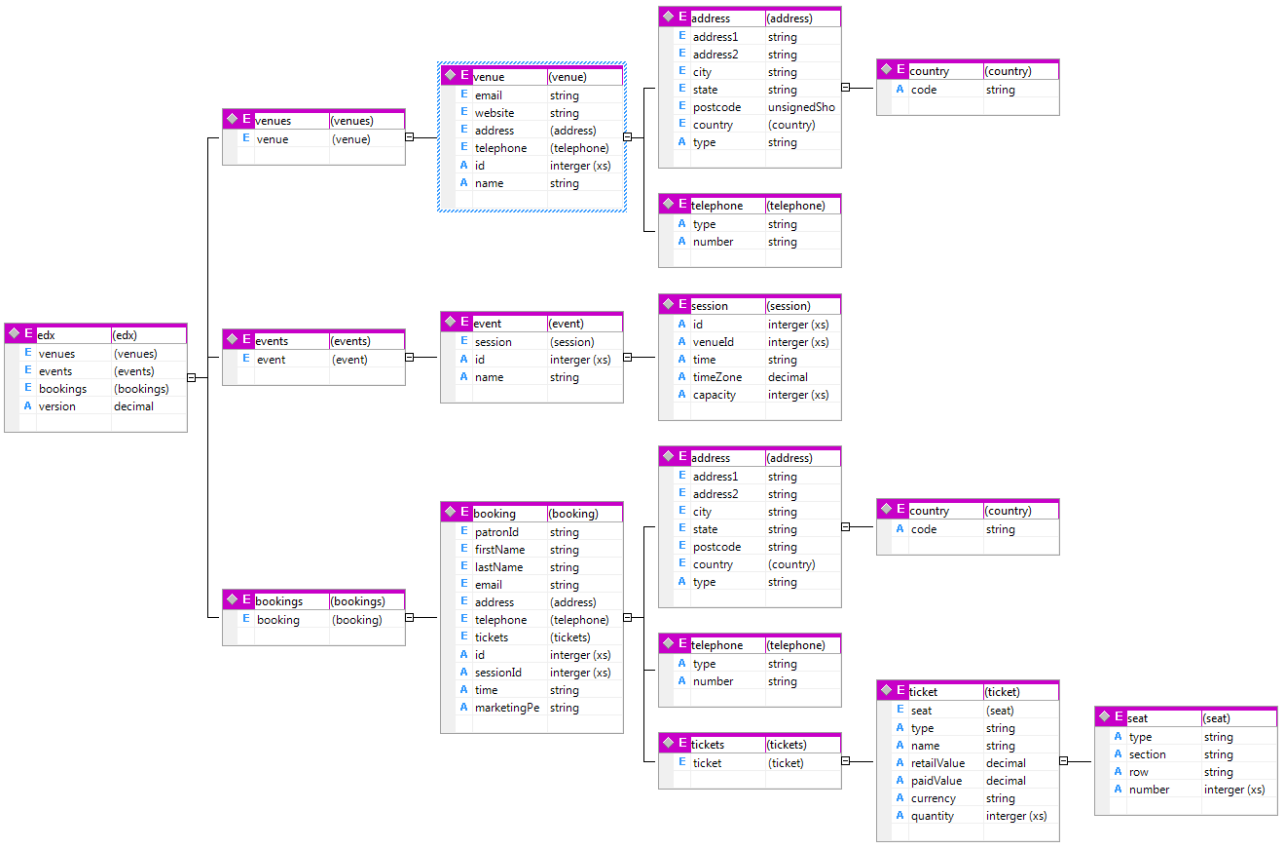
Version	Notes
0.8	Added a message tag <message> to the root node. Added permission based marketing attribute (marketingPermission) to Booking. Added the time of booking attribute (time) to Booking Added time zone attribute (timeZone) to Session Added currency attribute (currency) to Ticket New Appendix D - recommended API Notes, updated all date & time fields to be UTC Updated definitions Updated UML Data Structure Updated XSD
0.7	New document.

UML Data Structure

Required fields:



All fields :



Basic Data Structure

The data message is made of three nodes off the root node: event, venue and booking. Each node is repeated for the data to be retrieved. An empty "<edx>" message will constitute no data for the period requested.

Tag	Attributes	Type	Description	Required
edx	version	node	integer	Yes
venues		node	See Venues Data Structure	Yes*
events		node	See Events Data Structure	Yes*
bookings		node	See Bookings Data Structure	Yes*
message		node	See Message Data Structure	No

*If data is present in any one of venues, events or bookings then all are required.
The message tag is used to indicate a problem.

Example:

```
<edx version="1.0">
        <venues />
        <events />
        <bookings />
        <message />
</edx>
```

Message Data Structure

The Message tag is away for the Server to pass messages back to the client. The message tag will mainly be use when the server can't fulfil the request eg bad password, or Account Id.

Tag	Attributes	Type	Description	Required
message		node	Contains human readable message as to the problem.	Yes
message	status	integer	0 = normal 1 = bad Account/username or password 2 = Server error 3 = No data for requested date(s)	No

Example 1:

<message status=”1”>
AcountID or Password is incorrect
</message>

Example 2:

<message status=”3”>
No data for requested period.
</message>

Venues Data Structure

The Venue node describes the venue for which the Event is being held at. The Address node is optional.

Tag	Attributes	Type	Description	Required
venues		node	Container for venue nodes	Yes
venue	id	integer		Yes
venue	name	char	Name of the Venue.	Yes
email		char		No
website		Char	Website address	No
address		node	See Address Data Structure	No
telephone		node	See Telephone Data Structure	No

Example:

```
<venues>
        <venue id=”123” name=”Town Hall”>
                <email>info@example.com</email>
                <website>http://www.example.com</website>
                <address />
                <telephone />
        </venue>
</venues>
```

Event Data Structure

The event node describes the Event for which Bookings are made to. The session node holds each time the event is run.

Tag	Attributes	Type	Description	Required
events		node	Container for event nodes	Yes
event	id	Integer		Yes
event	name	Char	Name of the Event	Yes
session		node	See Session Data Structure	Yes

Example 1:

```
<events>
	<event id=”234” name=”My Big Day Out Event” >
		<session />
	</event>
</events>
```

Example 2 with session data:

```
<events>
	<event id=”234” name=”My Big Day Out Event” >
		<session id=”456” venueId=”123” time=”2009-10-02 16:00:00” timeZone=”10.0” capacity=”200” />
		<session id=”460” venueId=”123” time=”2009-10-02 19:30:00” timeZone=”10.0” capacity=”200” />
		<session id=”499” venueId=”123” time=”2009-10-03 19:30:00” timeZone=”10.0” capacity=”200” />
	</event>
</events>
```

Session Data Structure

The session node describes the Event sessions for which Bookings are made to. The session node holds each time the event is run.

Tag	Attributes	Type	Description	Required
session		node	Holds session times for the event.	Yes
session	id	Integer		Yes
session	venueId	Integer	Id of the venue	Yes
session	time	Datetime	The UTC session time for the event.	Yes
session	timeZone	float	Time difference from UTC eg: 10.5 or -8.0	Yes
session	capacity	Integer	Maximum number people that can attend this eventually	No

Example 1:

<session id=”456” venueId=”123” time=”2009-10-01 19:30:00” timeZone=”10.0” capacity=”200” />

Booking Data Structure

The Booking node describes the patron details for the Event attended.

Tag	Attributes	Type	Description	Required
bookings		node	Container for venue nodes	Yes
booking	id	Integer	Unique Id for this booking	Yes
booking	patronId	Integer /Char	Can be alphanumerical identify. The Id should identify an individual The Id can be a char and could be an email address.	Yes
booking	sessionId	Integer	The Id of the Event Session	Yes
booking	time	DateTime	UTC date time of when the booking was made.	Yes
booking	marketingPermission	Yes/No	Contains “Yes” or “No” to indicate permission to send marketing material.	No
firstName		Char	First name of person making the booking	Yes
lastName		Char	Last name of person making the booking	Yes
email		Char	Email address	No
address		node	See Address Data Structure	No
telephone		node	See Telephone Data Structure	No
tickets		node	See Ticket Data Structure	Yes

Example 1:

```
<bookings>
        < booking id=”123” sessionId=”456” time=”2009-10-25 14:30:00”
marketingPermission=”Yes”>
                <patronId>558942</patronId>
                <firstName>Grant</firstName>
                <lastName>Dunoon</lastName>
                <email>grant@dunoon.com.au</email>
                <address />
                <telephone />
                <tickets />
        </booking>
</bookings>
```

Example 2 (using the email as the patron Id):

```
<bookings>
        < booking id=”123” sessionId=”456” time=”2009-10-25 14:30:00”>
                <patronId>grant@dunoon.com.au</patronId>
                <firstName>Grant</firstName>
                <lastName>Dunoon</lastName>
                <email>grant@dunoon.com.au</email>
                <address />
                <telephone />
                <tickets />
        </booking>
</bookings>
```

Address Data Structure

The Address node describes the patrons address detail. Address is optional however if used must be complete.

Tag	Attributes	Type	Description	Required
address		node	Container for and street address.	Yes
address	type	Char	Types: street, PO, unknown	No
address1		Char	Line 1 of the address	Yes
address2		Char	Line 2 of the address	No
city		Char		Yes
state		Char		Yes
postcode		Char		Yes
country		Char		Yes
country	code	Char	2 letter country code, internet DNS zone: eg: Australia = au, USA=us	No

Example 1:

```
<address type=”street”>
        <address1>Suite 602</address1>
        <address2>1 Princess Street</address2>
        <city>Kew</city>
        <state>Victoria</state>
        <postcode>3101</postcode>
        <country code=”au”>Australia</country>
</address>
```

Example 2:

```
<address type=”PO”>
        <address1>P. O. Box 123</address1>
        <city>Kew</city>
        <state>Victoria</state>
        <postcode>3101</postcode>
        <country code=”au”>Australia</country>
</address>
```

Telephone Data Structure

The telephone node describes the telephone number. The Telephone node is optional.

Tag	Attributes	Type	Description	Required
telephone		node	Container node	Yes
telephone	type	char	Values: mobile, home, work	No
telephone	number	char	Telephone number, may contain country access code (+), space etc.	Yes

Example 1:

<telephone type=”mobile” number=”+613402993808” />

Example 2:

<telephone type=”home” number=”9999 5555” />

Example 3:

<telephone type=”work” number=”(03)90123460” />

Tickets Data Structure

The Tickets node describes the ticketing for which the Event is being held.

Tag	Attributes	Type	Description	Required
tickets		node	Container for ticket nodes	Yes
ticket				
ticket	type	Char	Value: seat, general	Yes
ticket	name	Char	The name you have called this ticket eg: Adult, Child, Single etc.	No
ticket	retailValue	Money	The normal retail value of the ticket	Yes
ticket	paidValue	Money	The value the patron was charged. Free of charge tickets will have a paidValue=”0”	Yes
ticket	currency	Char	3 letter currency code. Australia=AUD	Yes
ticket	quantity	Integer	Number of tickets for this ticket type eg 1 for Adult, 4 for Family ticket.	Yes
seat		node	Node holding the seat details	No*

* required if a ticket type is seat.

Example 1:

```
<tickets>
        <ticket type=”seat” name=”Adult” retailValue=”100.00” paidValue=”80.00” currency =”AUD” quantity=”1” >
                <seat />
        </ticket>
</tickets>
```

Example 2:

```
<tickets>
        <ticket type=”general” retailValue=”100.00” paidValue=”80.00” currency =”AUD” quantity=”1”  />
</tickets>
```

Example 3 (with seat nodes):

```
<tickets>
    <ticket type=”seat” name=”Family” retailValue=”200.00” paidValue=”200.00” currency =”AUD” quantity=”4” >
        <seat type=”seat” section=”stalls” row=”A” number=”23” />
        <seat type=”seat” section=”stalls” row=”A” number=”24” />
        <seat type=”seat” section=”stalls” row=”A” number=”25” />
        <seat type=”seat” section=”stalls” row=”A” number=”26” />
    </ticket>
</tickets>
```

Seat Data Structure

The telephone node describes the telephone number. The Telephone node is optional.

Tag	Attributes	Type	Description	Required
seat		node	Container the seat	Yes
seat	type	char	Values: seat, wheelchair	Yes
seat	section	char	Section name for which the seat is in	No
seat	row	char	The row the seat is in	Yes
seat	number	char	The seat number	Yes

Example 1:

<seat type=”seat” section=”stalls” row=”A” number=”23” />

Example 2:

<seat type=”wheelchair” section=”stalls” row=”A” number=”1” />

Acknowledgements

Grant Dunoon, Bookwana Pty. Ltd.

Leanne Gunnulson
Client Service Officer, ADVICE VICTORIA
(Audience data and Visitor Information Collection Enterprise)
Australia Council for the Arts

References

[XML] Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation 6 October 2000 http://www.w3.org/TR/2000/REC-xml-20001006

Appendix A

Example Output Required Fields:

Note: This example only shows required fields.

```
<?xml version="1.0" encoding="UTF-8"?>
<edx version="1.0">
 <venues>
  <venue id="123" name="Town Hall" />
 </venues>
 <events>
  <event id="234" name="My Big Day Out Event" >
   <session id="456" venueId="123" time="2009-10-01 19:30:00" timeZone="10.0" capacity="200" />
  </event>
 </events>
 <bookings>
  < booking id="123" sessionId="456" time="2009-09-01 08:30:00">
   <patronId>558942</patronId>
   <firstName>Grant</firstName>
   <lastName>Dunoon</lastName>
   <tickets>
    <ticket type="seat" name="Family" retailValue="200.00" paidValue="200.00" currency="AUD" quantity="4" >
     <seat type="seat" section="stalls" row="A" number="23" />
     <seat type="seat" section="stalls" row="A" number="24" />
     <seat type="seat" section="stalls" row="A" number="25" />
     <seat type="seat" section="stalls" row="A" number="26" />
    </ticket>
   </tickets>
   </booking>
  </bookings>
</edx>
```

Appendix B

Example Output in Full:

Note: This example only all fields.

```
<?xml version="1.0" encoding="utf-8"?>
<edx version="1.0">
 <venues>
  <venue id="123" name="Town Hall" >
   <email>info@example.com</email>
   <website>http://www.example.com</website>
   <address type="street">
    <address1>1 Main Road</address1>
    <city>Kew</city>
    <state>Victoria</state>
    <postcode>3101</postcode>
    <country code="au">Australia</country>
   </address>
   <telephone type="work" number="(03)9999 5555" />
  </venue>
 </venues>
 <events>
  <event id="234" name="My Big Day Out Event" >
   <session id="456" venueId="123" time="2009-10-01 19:30:00" timeZone="10.0" capacity="200" />
  </event>
 </events>
 <bookings>
  <booking id="123" sessionId="456"  time="2009-09-01 08:30:00" marketingPermission="Yes">
   <patronId>grant@dunoon.com.au</patronId>
   <firstName>Grant</firstName>
   <lastName>Dunoon</lastName>
   <email>grant@dunoon.com.au</email>
   <address type="street">
    <address1>Suite 602</address1>
    <address2>1 Princess Street</address2>
    <city>Kew</city>
    <state>Victoria</state>
    <postcode>3101</postcode>
    <country code="au">Australia</country>
   </address>
   <telephone type="mobile" number="+613402993808" />
   <tickets>
    <ticket type="seat" name="Family" retailValue="200.00" paidValue="200.00" currency="AUD" quantity="4" >
     <seat type="seat" section="stalls" row="A" number="23" />
     <seat type="seat" section="stalls" row="A" number="24" />
     <seat type="seat" section="stalls" row="A" number="25" />
     <seat type="seat" section="stalls" row="A" number="26" />
    </ticket>
   </tickets>
  </booking>
 </bookings>
</edx>
```

Appendix C

XSD:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="edx">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="venues">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="venue">
        <xs:complexType>
         <xs:sequence>
          <xs:element name="email" type="xs:string" />
          <xs:element name="website" type="xs:string" />
          <xs:element name="address">
           <xs:complexType>
            <xs:sequence>
             <xs:element name="address1" type="xs:string" />
             <xs:element name="address2" type="xs:string" />
             <xs:element name="city" type="xs:string" />
             <xs:element name="state" type="xs:string" />
             <xs:element name="postcode" type="xs:unsignedShort" />
             <xs:element name="country">
              <xs:complexType>
               <xs:simpleContent>
                <xs:extension base="xs:string">
                 <xs:attribute name="code" type="xs:string" use="required" />
                </xs:extension>
               </xs:simpleContent>
              </xs:complexType>
             </xs:element>
            </xs:sequence>
            <xs:attribute name="type" type="xs:string" use="required" />
           </xs:complexType>
          </xs:element>
          <xs:element name="telephone">
           <xs:complexType>
            <xs:attribute name="type" type="xs:string" use="required" />
            <xs:attribute name="number" type="xs:string" use="required" />
           </xs:complexType>
          </xs:element>
         </xs:sequence>
         <xs:attribute name="id" type="xs:interger" use="required" />
         <xs:attribute name="name" type="xs:string" use="required" />
        </xs:complexType>
       </xs:element>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
```

```
<xs:element name="events">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="event">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="session">
       <xs:complexType>
        <xs:attribute name="id" type="xs:interger" use="required" />
        <xs:attribute name="venueId" type="xs:interger" use="required" />
        <xs:attribute name="time" type="xs:string" use="required" />
        <xs:attribute name="timeZone" type="xs:decimal" use="required" />
        <xs:attribute name="capacity" type="xs:interger" use="required" />
       </xs:complexType>
      </xs:element>
     </xs:sequence>
     <xs:attribute name="id" type="xs:interger" use="required" />
     <xs:attribute name="name" type="xs:string" use="required" />
    </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name="bookings">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="booking">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="patronId" type="xs:string" />
      <xs:element name="firstName" type="xs:string" />
      <xs:element name="lastName" type="xs:string" />
      <xs:element name="email" type="xs:string" />
      <xs:element name="address">
       <xs:complexType>
        <xs:sequence>
         <xs:element name="address1" type="xs:string" />
         <xs:element name="address2" type="xs:string" />
         <xs:element name="city" type="xs:string" />
         <xs:element name="state" type="xs:string" />
         <xs:element name="postcode" type="xs:string" />
         <xs:element name="country">
          <xs:complexType>
           <xs:simpleContent>
            <xs:extension base="xs:string">
             <xs:attribute name="code" type="xs:string" use="required" />
            </xs:extension>
           </xs:simpleContent>
          </xs:complexType>
         </xs:element>
        </xs:sequence>
        <xs:attribute name="type" type="xs:string" use="required" />
       </xs:complexType>
      </xs:element>
      <xs:element name="telephone">
       <xs:complexType>
        <xs:attribute name="type" type="xs:string" use="required" />
```

```
            <xs:attribute name="number" type="xs:string" use="required" />
          </xs:complexType>
        </xs:element>
        <xs:element name="tickets">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ticket">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element maxOccurs="unbounded" name="seat">
                      <xs:complexType>
                        <xs:attribute name="type" type="xs:string" use="required" />
                        <xs:attribute name="section" type="xs:string" use="required" />
                        <xs:attribute name="row" type="xs:string" use="required" />
                        <xs:attribute name="number" type="xs:interger" use="required" />
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                  <xs:attribute name="type" type="xs:string" use="required" />
                  <xs:attribute name="name" type="xs:string" use="required" />
                  <xs:attribute name="retailValue" type="xs:decimal" use="required" />
                  <xs:attribute name="paidValue" type="xs:decimal" use="required" />
                  <xs:attribute name="currency" type="xs:string" use="required" />
                  <xs:attribute name="quantity" type="xs:interger" use="required" />
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="id" type="xs:interger" use="required" />
      <xs:attribute name="sessionId" type="xs:interger" use="required" />
      <xs:attribute name="time" type="xs:string" use="required" />
      <xs:attribute name="marketingPermission" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
 </xs:sequence>
 <xs:attribute name="version" type="xs:decimal" use="required" />
 </xs:complexType>
 </xs:element>
</xs:schema>
```

Appendix D

Application Programming Interface (API)

This appendix is non-normative and describes a recommendation for an API.

Because the EDX contains personal information (address, email etc) it is recommended that all communication take place in SSL.

To reduce network and database load it is recommended that only a single day of data be returned per requested. The advantages of this method is if the client needs to sync a months worth of data, a progress bar could reflected each day as a percentage of the overall progress.

The request should POST the data fields and not place them in the querystring. This is to avoid the server logging the qureystring in clear text and exposing a potential security risk.

Request Fields

Field	Type	Notes
AccountID	Char	
Password	Char	
Date	Date	UTC date. YYYY-MM-DD

Example