Powered by Maritz Global Events

Real Time Services (RTS)

Contents

Overview

Real Time Services (RTS) is a suite of web services hosted by Maritz Global Events which enables our clients and partners to access live data on-demand. The data will be available during all phases of the show, including onsite. Using RTS, the consumer has the option to pull information back in XML-formatted strings or create a CSV file available for download on the Maritz Global Events client FTP site.

Description

WSDL

The WSDL of the RTS Web Service is accessible. It can be viewed at this secure URL:

https://ews.exl.mge360.com/rts/export.asmx

**NOTE ** There may be some methods included in this WSDL that are deprecated, not-recommended or for internal use only. For any questions regarding availability of methods not found in this document, please contact your Maritz Global Events contact.

Data Protection

RTS supports Data Protection regulations giving users in specific countries the option to opt out of sharing their personally identifiable information (PII). In these cases, RTS will still return a record for these individuals however all of their PII will be removed. Data objects that are subject to this policy are marked with a *.

Data Objects

Registrant*

Each user that registers for an event has a unique Registrant record. This record contains vital information pertaining to the user including contact information, answers to demographic questions, purchases and reservations.

Registrant

Includes contact information, demographic responses and purchases.

```
<RegistrantList CurrentPage="0" PageSize="100" ShowCode="AFH557">
        <Registrant>
                <RegistrantID>1080</RegistrantID>
                <RegState>COM</RegState>
                <NickName/>
                <FirstName>test1</FirstName>
```

* Please Note: This data object is subject to our Data Protection policy. For registrants who have opted out, all of their PII will be removed and the FirstName and LastName nodes will have their information replaced with 'OPTOUT'. This also includes any `Freeform` type Demographic responses.

```
<Middle/>
<LastName>tester1</LastName>
<RegTypeCode>RD</RegTypeCode>
<QCGroupNumber>WEB</QCGroupNumber>
<ShowItemList>STUFFBS,BS,REG,AS,CS,AS,BS,</ShowItemList>
<Prefix/>
<Suffix/>
<Title/>
<ExCompanyID>0</ExCompanyID>
<ExhibitorName/>
<CompanyCode/>
<Company>test</Company>
<Company2/>
<Address>12345 test lane</Address>
<Address2/>
<Address3/>
<City>Frederick</City>
<StateCode>MD</StateCode>
<ZipCode>21704</ZipCode>
<CountryCode/>
<Phone/>
<PhoneExtension/>
<Fax/>
<Email>tester1@test.com</Email>
<MemberNumber/>
<MemberType>N</MemberType>
<IsCancelled>0</IsCancelled>
<IsPended>false</IsPended>
<GuestID>1080</GuestID>
<UpdateDate>2014-09-03T10:01:42-04:00</UpdateDate>
<InsertDate>2014-09-02T14:42:39-04:00</InsertDate>
<MemberCategory/>
<Phone2/>
<Phone2Extension/>
<VerifyTypeCode/>
<NPINumber/>
<EngageAccessCode>kqzty</EngageAccessCode>
<Degree/>
<AddressTypeCode/>
<HsnState>INP</HsnState>
<Demographics>
        <sysRowStamp Type="Freeform">AAAAAAAMhA8=</sysRowStamp>
        <sysEventID Type="Freeform">-1</sysEventID>
        <sysNoteID Type="Freeform">-1</sysNoteID>
        <SingleSt Type="Lookup">A</SingleSt>
        <MultiSt Type="Lookup">1,2</MultiSt>
```

```
            <TextSt Type="Freeform">TextSt1</TextSt>
            <SingleDy Type="Lookup">BY</SingleDy>
            <MultiDy Type="Lookup">1Y,2Y</MultiDy>
            <TextDy Type="Freeform">hello</TextDy>
            <Days Type="Lookup"/>
            <DiscCode Type="Freeform">ten</DiscCode>
            <test Type="Lookup"/>
        </Demographics>
        <Purchases>
            <Purchase InsertDate="2014-09-02T14:42:40-04:00"
                      IsVoided="0"
                      RegTranState="COM"
                      RegTransactionID="2225"
                      ShowItemCode="STUFFBS"
                      ShowItemTreeID="8"
                      TotalAmount="0.0000"
                      UnitQuantity="1"
                      UpdateDate="2014-09-02T14:42:40-04:00"/>
            <Purchase InsertDate="2014-09-02T14:42:40-04:00"
                      IsVoided="0"
                      RegTranState="COM"
                      RegTransactionID="2226"
                      ShowItemCode="BS"
                      ShowItemTreeID="1"
                      TotalAmount="20.0000"
                      UnitQuantity="1"
                      UpdateDate="2014-09-02T14:42:40-04:00"/>
            <Purchase InsertDate="2014-09-02T14:42:40-04:00"
                      IsVoided="0"
                      RegTranState="COM"
                      RegTransactionID="2227"
                      ShowItemCode="REG"
                      ShowItemTreeID="2"
                      TotalAmount="0.0000"
                      UnitQuantity="1"
                      UpdateDate="2014-09-02T14:42:40-04:00"/>
            <Purchase InsertDate="2014-09-02T14:42:40-04:00"
                      IsVoided="0"
                      RegTranState="COM"
                      RegTransactionID="2228"
                      ShowItemCode="AS"
                      ShowItemTreeID="1"
                      TotalAmount="140.0000"
                      UnitQuantity="7"
                      UpdateDate="2014-09-02T14:42:40-04:00"/>
            <Purchase InsertDate="2014-09-02T14:42:40-04:00"
```

```
                                IsVoided="0"
                                RegTranState="COM"
                                RegTransactionID="2229"
                                ShowItemCode="CS"
                                ShowItemTreeID="1"
                                TotalAmount="80.0000"
                                UnitQuantity="4"
                                UpdateDate="2014-09-02T14:42:40-04:00"/>
                        <Purchase InsertDate="2014-09-03T10:01:41-04:00"
                                IsVoided="0"
                                RegTranState="COM"
                                RegTransactionID="2232"
                                ShowItemCode="AS"
                                ShowItemTreeID="1"
                                TotalAmount="100.0000"
                                UnitQuantity="5"
                                UpdateDate="2014-09-03T10:01:41-04:00"/>
                        <Purchase InsertDate="2014-09-03T10:01:41-04:00"
                                IsVoided="0"
                                RegTranState="COM"
                                RegTransactionID="2233"
                                ShowItemCode="BS"
                                ShowItemTreeID="1"
                                TotalAmount="80.0000"
                                UnitQuantity="4"
                                UpdateDate="2014-09-03T10:01:41-04:00"/>
                </Purchases>
                <RegistrantFlag>
                        <CustomCode>AgreeToPrivacyPolicy</CustomCode>
                        <FlagValue>true</FlagValue>
                        <UpdateDate>2018-04-17T03:58:25-04:00</UpdateDate>
                </RegistrantFlag>
                <RegistrantFlag>
                        <CustomCode>AgreeToTermsOfService</CustomCode>
                        <FlagValue>true</FlagValue>
                        <UpdateDate>2018-04-17T03:58:25-04:00</UpdateDate>
                </RegistrantFlag>
                <RegistrantFlag>
                        <CustomCode>RequiredToSeePrivacyPolicy</CustomCode>
                        <FlagValue>true</FlagValue>
                        <UpdateDate>2018-04-17T03:58:25-04:00</UpdateDate>
                </RegistrantFlag>
        </Registrant>
</RegistrantList>
```

Node Definitions

- RegistrantID – Unique number identifying the registrant record
- RegState – The state of the user's registration. Can be one of:
    - COM (Complete)
    - INP (In Progress)
    - CXL (Cancelled)
- NickName, FirstName, Middle, LastName, Prefix, Suffix, Company, Address, City, StateCode, Phone, PhoneExtension, Fax, Email, ZipCode, CountryCode – Contact information fields
- ShowItemList – Provides a list of the codes associated with sessions/showitems purchased by the user for this particular event
- ExCompanyID – Links registrant to the correct exhibiting company if they are an exhibitor
- MemberNumber – Associates registrant with a member record if they were loaded as a member
- IsCancelled – Shows whether the registrant has been cancelled (1) or not (0)
- IsPended – Determines if registrant is currently pended.
- GuestID – Used for guests, this represents the RegistrantID of the main registrant that booked this person. If the registrant is not a guest, this will be the same as their RegistrantID
- Demographics – Contains Demographic responses
    - Type determines the category that the demographic falls under.(Ex: Freeform would be a response typed by user, while Lookup a response that was selected)
- Purchases – Provides a list of the user's purchases including their registration, show items/sessions, guests, and packages.
    - InsertDate, UpdateDate – represent the date/time the purchase was made or updated
    - TotalAmount, UnitQuantity – Self explanatory
    - ShowItemCode – The unique code that identifies the item that was purchased
    - ShowItemTreeID – The identifying number of the branch the show item/session belongs to
    - IsVoided – Determines whether this purchase has been voided (1=true, 0=false)
    - RegTransactionID – Identifying number for the purchase's transaction

  - RegTranState – Represents whether the transaction was completed (COM), still in progress (INP), or cancelled (CXL)
- RegistrantFlag – These nodes are listed inside of the Registrant node denoting if a registrant has agreed to a policy or not.
  - Code – The Unique code that identifies a unique type of Registrant Flag
  - FlagValue – This field indicates if the user has agreed to the flag or not
  - By default, there are three different flags
    - AgreeToTermsOfService – Terms of using EventXL
    - RequiredToSeePrivacyPolicy – Registrant given the option to Opt Out of data sharing
    - AgreeToPrivacyPolicy – Registrant Privacy Policy

RegistrantWithHousing

Similar to the above Registrant, this includes contact information, demographic responses and purchases, but also contains reservation information.

```
<RegistrantList CurrentPage="1" PageSize="100" ShowCode="AFH557">
        <Registrant>
                <RegistrantID>1021</RegistrantID>
                <BookingID>1022</BookingID>
                <RegState>COM</RegState>
                <HSNState>COM</HSNState>
                ...
                <Demographics>
                        ...
                </Demographics>
                <Purchases>
                        ...
                </Purchases>
                <Reservations>
                        <Reservation ArrivalDate="2015-04-26T00:00:00-04:00"
                                     BillingComments=""
                                     BookingDate="2014-10-02T09:01:55-04:00"
                                     CancellationDate="2014-10-18T12:03:01.87-04:00"
                                     DepartureDate="2015-04-29T00:00:00-04:00"
                                     HotelCategoryDescription=""
                                     HotelCode="12240"
                                     HotelConfirmationNumber=""
                                     HotelDeposit="0.0000"
                                     HotelName="Insert Name Here"
                                     InventoryPartCode=""
                                     OccupantTypeCode="LEADOCC"
                                     ReservationID="1001"
```

```
                                    ReservationInsertDate="2014-10-02T09:01:55-04:00"
                                    ReservationStatus="X"
                                    ReservationUpdateDate="2014-10-18T12:02:19-04:00"
                                    SpecialRequestComments=""
                                    WebComments=""/>
            </Reservations>
            <Comments/>
        </Registrant>
</RegistrantList>
```

Node Definitions

- HSNState – Represents the state of the housing on this record. Can be one of:
  - COM (Complete)
  - INP (In Progress)
  - CXL (Cancelled)
- Reservations – Provides a list of reservations made under this record
  - ReservationID – A unique number identifying each room reservation
  - ReservationUpdateDate, ReservationInsertDate – self explanatory
  - SpecialRequestComments, BillingComments, WebComments – Special Request comments could include preferences such as location of room or smoking/non-smoking
  - HotelDeposit – Self explanatory
  - CancellationDate, BookingDate – Dates the reservation was cancelled (if it was cancelled) and booked, respectively
  - OccupantTypeCode – Is the occupant the lead occupant (LEADOCC) of the room or a guest
  - ArrivalDate, DepartureDate, HotelName – Self explanatory

Member

Registrant membership information is held in the Member object. This object contains the identifying MemberNumber that is included on each member registrant, contact information that is loaded into each registrant upon their first login, and additional fields that represent show level configurations such as MemberType and RegTypeCode.

MemberList

```
<MemberList CurrentPage="1" PageSize="100" ShowCode="SHO151">
        <Member>
            <MemberNumber>CA6</MemberNumber>
            <LastName>test</LastName>
            <FirstName>test</FirstName>
            <Middle/>
```

```
            <Prefix/>
            <Suffix/>
            <Title/>
            <Company>Test Company</Company>
            <Company2/>
            <Address>5202 Presidents Court</Address>
            <Address2/>
            <City>Frederick</City>
            <StateCode>MD</StateCode>
            <ZipCode>21703</ZipCode>
            <CountryCode>
            <Phone>1111111111</Phone>
            <PhoneExtension/>
            <Fax/>
            <Email>test@test.com</Email>
            <RegTypeCode/>
            <RegSourceCode/>
            <MemberType>MEM</MemberType>
            <ExID>CA6</ExID>
            <InsertDate>2014-09-30T22:41:32-04:00</InsertDate>
            <UpdateDate>2014-09-30T22:41:32-04:00</UpdateDate>
            <MemberCategory/>
            <NPINumber/>
        </Member>
</MemberList>
```

Node Definitions

- MemberNumber – The unique identifying number associated with each member object. This field is added to the corresponding registrant record if the member record is used. Depending on application, this field can be formatted as all digits or contain characters. Example: “AAA-000”, “1000”, “MEMBER” are all valid member numbers
- LastName, FirstName, Middle, Prefix, Suffix, Title, Company, Company2, Address, Address2, City, StateCode, ZipCode, CountryCode, Phone, PhoneExtension, Fax, Email – Contact information fields. There are instances where fields such as first name and last name will be blank. An example would be for company level membership, where the member record could be used by several registrants.
- RegTypeCode – The code representing this member’s registration type. Associates them with a Registration Status object.
- MemberType – Some shows have multiple membership statuses that drive fees and business rules. This field associates each member object with their corresponding membership status.

- MemberCategory – A field that can be used to hold custom information about each member record. As an example, for shows with both individual and company membership, this field can be used to determine which category each record falls under
- NPINumber – National Provider Identifier number

Exhibiting Companies

Companies that plan to Exhibit at an event are loaded into our system as ExCompany objects. These objects hold the companies' contact and booth information, as well as a few other identifying fields that allow them to be linked to registrants. When registrants are created under an Exhibiting Company, they are given the ExCompanyID that is associated with it. This relationship is used not only to keep track of the amount of registrants attending an event under each company, but to calculate booth allotments as well.

ExCompanyList

```
<ExCompanyList CurrentPage="1" PageSize="100" ShowCode="SHO151">
        <ExCompany>
                <ExCompanyID>1002</ExCompanyID>
                <ExCompanyBoothGroupID>0</ExCompanyBoothGroupID>
                <BoothName>Test Company</BoothName>
                <Company>Test Company</Company>
                <Company2/>
                <CompanyCode/>
                <Address>5202 Presidents Court</Address>
                <Address2/>
                <City>Frederick</City>
                <StateCode>MD</StateCode>
                <Zipcode>21703</Zipcode>
                <CountryCode/>
                <Phone/>
                <Fax/>
                <Email>test@test.com</Email>
                <BoothNumber>313</BoothNumber>
                <BoothSize>test</BoothSize>
                <Website/>
                <Note/>
                <InsertDate>2015-04-30T09:45:20-04:00</InsertDate>
                <UpdateDate>2015-05-07T14:33:09-04:00</UpdateDate>
                <IsActive>true</IsActive>
                <Username/>
                <Password>test</Password>
                <ProductList/>
        </ExCompany>
```

```
</ExCompanyList>
```

Node Definitions

- ExCompanyID – The unique identification number associated with this company. This is also passed into each registrant that is registered under a company and is used to calculate allotments
- BoothName – The name given to this company's booth. This is often the same as the company name and is also what appears to the user in many EventXL applications
- Company, Address, Address2, City, StateCode, ZipCode, CountryCode, Phone, Fax, Email – Contact fields
- BoothNumber – Identifying number for the booth associated with this Exhibiting Company
- BoothSize – The size of this company's booth
- ProductList – A list of products that will be sold by this company at the event (Deprecated)

Demographic

Demographics represent custom fields that contain additional event specific information about each registrant. These are objects that are not included with the base registrant fields such as contact information, registration types, or membership information.

DemographicLookupListWithFreeform

```
<DemographicsLookupListWithFreeform ShowCode="AFH557">
        <DemographicsLookup ColumnDescription="" FieldName="Days" Type="Lookup">
                <DemographicResponse Description="Monday"
                                     IsOther="false"
                                     PickCode="M"/>
                <DemographicResponse Description="Tuesday"
                                     IsOther="false"
                                     PickCode="T"/>
                <DemographicResponse Description="Wednesday"
                                     IsOther="false"
                                     PickCode="W"/>
        </DemographicsLookup>
        <DemographicsLookup ColumnDescription="" FieldName="DiscCode"
                            Type="Freeform"/>
        <DemographicsLookup ColumnDescription="" FieldName="MultiDy"
                            Type="Lookup">
                <DemographicResponse Description="1Y"
                                     IsOther="false"
                                     PickCode="1Y"/>
                <DemographicResponse Description="2Y"
```

```
                                    IsOther="false"
                                    PickCode="2Y"/>
            <DemographicResponse Description="3Y"
                                    IsOther="false"
                                    PickCode="3Y"/>
    </DemographicsLookup>
    <DemographicsLookup ColumnDescription="" FieldName="MultiSt" Type="Lookup">
            <DemographicResponse Description="1"
                                    IsOther="false"
                                    PickCode="1"/>
            <DemographicResponse Description="2"
                                    IsOther="false"
                                    PickCode="2"/>
            <DemographicResponse Description="3"
                                    IsOther="false"
                                    PickCode="3"/>
    </DemographicsLookup>
    <DemographicsLookup ColumnDescription="" FieldName="SingleDy"
    Type="Lookup">
            <DemographicResponse Description="AY"
                                    IsOther="false"
                                    PickCode="AY"/>
            <DemographicResponse Description="BY"
                                    IsOther="false"
                                    PickCode="BY"/>
            <DemographicResponse Description="CY"
                                    IsOther="false"
                                    PickCode="CY"/>
    </DemographicsLookup>
    <DemographicsLookup ColumnDescription="" FieldName="SingleSt"
    Type="Lookup">
            <DemographicResponse Description="A"
                                    IsOther="false"
                                    PickCode="A"/>
            <DemographicResponse Description="B"
                                    IsOther="false"
                                    PickCode="B"/>
            <DemographicResponse Description="C"
                                    IsOther="false"
                                    PickCode="C"/>
    </DemographicsLookup>
    <DemographicsLookup ColumnDescription="" FieldName="test" Type="Lookup"/>
    <DemographicsLookup ColumnDescription="" FieldName="TextDy"
                            Type="Freeform"/>
    <DemographicsLookup ColumnDescription="" FieldName="TextSt"
                            Type="Freeform"/>
```

```
</DemographicsLookupListWithFreeform>
```

Node Definitions

- DemographicsLookup – Represents a custom demographics field that contains show specific information
  - FieldName – The unique name of the custom field that this demographic represents
  - Type – The form in which the demographic is displayed to the user
    - Freeform – Shown as an open text field for the registrant to enter their information
    - Lookup – Options are presented for the user to select their response
- DemographicResponse – For non-freeform fields, these are the options that are presented to the user
  - Description – The value that is seen by the user when selecting their response to the demographic
  - IsOther – Some demographics require additional information depending on their response – if the value of this field is true, this response has another demographic that contains additional information for this response
  - PickCode – The value that is entered into the registrant. This is what is seen in the demographic fields in the Registrant data pulls.

Status

Statuses help drive the registration process. Fees for registration and show items are derived from them. When registering to attend an event, a registrant is given a "Registration" status and in many cases a member type that matches a "Membership" status.

```
<StatusList CurrentPage="1" PageSize="100" ShowCode="AFH557">
    <Status InsertDate="2014-08-08T16:05:10-04:00" IsActive="true" StatusCode="A"
        StatusDescription="Advance" StatusID="1004" StatusMasterCode="Date"
        UpdateDate="2014-08-08T16:05:10-04:00"/>
    <Status InsertDate="2014-08-14T14:52:02-04:00" IsActive="true" StatusCode="M"
        StatusDescription="Member" StatusID="1006" StatusMasterCode="Membership"
        UpdateDate="2014-08-14T14:52:02-04:00"/>
    <Status InsertDate="2014-08-08T16:05:55-04:00" IsActive="true" StatusCode="O"
        StatusDescription="Onsite" StatusID="1005" StatusMasterCode="Date"
        UpdateDate="2014-08-08T16:05:55-04:00"/>
    <Status InsertDate="2014-08-08T16:03:01-04:00" IsActive="true" StatusCode="R1"
        StatusDescription="R1" StatusID="1001" StatusMasterCode="Registration"
        UpdateDate="2014-08-08T16:03:01-04:00"/>
</StatusList>
```

Node Definitions

- InsertDate – The date that this status was entered into the system
- IsActive – Determines whether this status is currently being used in the registration process
- StatusCode – A code that represents the status, for registration and membership status' this is what is seen in the RegType and MemberType fields
- StatusDescription – A short description of the status and what it represents
- StatusID – Unique identifying number for this status
- StatusMasterCode – Shows what category this status falls under. Common types are Date, Registration, or Membership.
- UpdateDate – The date that this object was updated in the system

Sessions

Sessions represent items that can be purchased by registrants. They are items that are sold before, during, or after the show such as registration, guests, workshops, donations, and materials.

SessionsList

```
<SessionList CurrentPage="1" PageSize="100" ShowCode="SHO151">
        <Session>
                <ShowItemID>2088</ShowItemID>
                <ShowItemCode>C08</ShowItemCode>
                <Description>8.  Risk Management</Description>
                <LimitQty>77</LimitQty>
                <ReservedQty>0</ReservedQty>
                <UsedQty>0</UsedQty>
                <StartDate>2015-05-02T10:00:00-04:00</StartDate>
                <EndDate>2015-05-02T11:00:00-04:00</EndDate>
                <SoldOutDate>1900-01-01T00:00:00-05:00</SoldOutDate>
                <ShowItemSynopsis/>
                <ExID/>
                <InsertDate>2014-10-08T10:23:52-04:00</InsertDate>
                <UpdateDate>2015-04-01T11:58:13-04:00</UpdateDate>
                <IsActive>true</IsActive>
                <ActiveStartDate>1900-01-01T00:00:00-05:00</ActiveStartDate>
                <ActiveEndDate>1900-01-01T00:00:00-05:00</ActiveEndDate>
                <MaxQtyPerReg>1</MaxQtyPerReg>
                <SessionCategoryDescription/>
                <Title/>
                <HTMLDescription>The facts about managing your risk, short and long term.
        Discussion of strategies to insure against loss of income, life, or ability to
        practices.</HTMLDescription>
```

```
            <Location>Phoenix Convention Center North - 227A</Location>
            <ShowItemTypeCode>SES</ShowItemTypeCode>
            <ShowFacilityID>0</ShowFacilityID>
            <ShowItemTreeId>1006</ShowItemTreeId>
            <ParentId>1</ParentId>
            <ShowItemTreeCode>LAB3</ShowItemTreeCode>
            <ShowItemTreeDescription>Learning Labs</ShowItemTreeDescription>
            <Speakers/>
            <SessionTracks>
                  <Track ShowItemTrackCode="T2" ShowItemTrackDescription="Exit
      Strategy/Selling Track" ShowItemTrackID="1066"/>
            </SessionTracks>
      </Session>
</SessionList>
```

Node Definitions

- ShowItemID – Unique identifying number for session/showitem
- ShowItemCode – This is the value that will be seen in registrant records under the ShowItemList field. For the Registration session, this will be "REG", or for cancellations it will be "CANCEL". Show specific sessions' values for this field will vary depending on setup. The field can contain digits and characters
- Description – Provides a short description of the item.
- LimitQty – Represents the quantity of the item. If there is no limit to the amount of purchases that can be made of this item, the LimitQty will be zero.
- ReservedQty – The amount of reserved instances of this item
- UsedQty – Represents how much of the quantity has been purchased. Subtracting this from the LimitQty will give the remaining quantity of the session.
- StartDate, EndDate – For sessions that represent events such as workshops or speaker sessions, these fields show the start date and end date times.
- SoldOutDate – Represents the date and time that this session has been sold out. If the item still has an available quantity, or is unlimited, the date will be as shown in the above example.
- InsertDate, UpdateDate – The dates/times that the item was inserted/updated in the system
- IsActive – Shows whether the item is being actively sold. If this is false, the item may no longer be purchased by registrants.
- ActiveStartDate, ActiveEndDate – These are the dates that the item will be available between. Once these dates have passed, or if the user comes in before the start date, the session will be marked as IsActive = false.

- MaxQtyPerReg –This field determines how many of each specific session that each registrant may purchase. Unlimited would be represented by a zero.
- HTMLDescription – Provides a long description of the session. May contain HTML to be presented on the web or in confirmations.
- Location – Contains a name for the location in which the Session is being held. This field should be blank for items such as Registration, Cancellation, and material items.
- ShowItemTypeCode – Represents the type of item that this Session represents.
    - SES – Represents a Session show item such as a Workshop, Orientation or Speaker session.
    - REG – Represents the Registration show item. Every registrant in a registration show should have one of these items. Cancellation show items will also appear with this type.
    - MER – This session is a merchandise show item.
    - MAT – Material
    - HOU – Housing Item such as a Hotel Deposit
- ShowFacilityID – Links to the facility (i.e. hotel or convention center) in which this session takes place
- ShowItemTreeID – A Show Item Tree is a grouping under which the sessions can fall, such as the Registration show item falling under the Registration tree. This ID links the session with the tree that it belongs to.
- ParentId – For showitem trees/grouping that exist within other trees, this represents the ID of the parent tree/grouping
- ShowItemTreeCode – The identifying code that is assigned to the tree this item belongs to.
- ShowItemTreeDescription – A short description of the tree/grouping
- Speakers – Shows the name(s) of the speakers for this session if applicable
- SessionsTracks – Tracks represent a way of grouping together related showitems if they are to be displayed together or handled similarly
    - ShowItemTrackCode – The unique code representing the track
    - ShowItemTrackDescription – A short description of the track
    - ShowItemTrackID – The unique identifying number associated with the track

ShowItemFeeList

The ShowItemFeeList object represents the amounts that will be charged for each session depending on their registration status, the date that they registered, and their registration

status. It contains fields for the amount, date status, member status, registration status, show item code, and the show item tree that the show item belongs to.

```
<ShowItemFeeList CurrentPage="1" PageSize="100" ShowCode="AFH557">
    <ShowItemFee Amount="10.0000" DateStatus="A" MemberStatus="M"
        RegistrationStatus="R1" ShowItemCode="AS" ShowItemFeeId="1037"
        ShowItemTreeCode="SHOWITEM"/>
    <ShowItemFee Amount="10.0000" DateStatus="A" MemberStatus="M"
        RegistrationStatus="R2" ShowItemCode="AS" ShowItemFeeId="1041"
        ShowItemTreeCode="SHOWITEM"/>
    <ShowItemFee Amount="0.0000" DateStatus="A" MemberStatus="M"
        RegistrationStatus="R3" ShowItemCode="AS" ShowItemFeeId="1045"
        ShowItemTreeCode="SHOWITEM"/>
        <ShowItemFee Amount="30.0000" DateStatus="O" MemberStatus="M"
        RegistrationStatus="R1" ShowItemCode="AS" ShowItemFeeId="1038"
        ShowItemTreeCode="SHOWITEM"/>
    <ShowItemFee Amount="30.0000" DateStatus="O" MemberStatus="M"
        RegistrationStatus="R2" ShowItemCode="AS" ShowItemFeeId="1042"
        ShowItemTreeCode="SHOWITEM"/>
    <ShowItemFee Amount="0.0000" DateStatus="O" MemberStatus="M"
        RegistrationStatus="R3" ShowItemCode="AS" ShowItemFeeId="1046"
        ShowItemTreeCode="SHOWITEM"/>
    <ShowItemFee Amount="20.0000" DateStatus="A" MemberStatus="N"
        RegistrationStatus="R1" ShowItemCode="AS" ShowItemFeeId="1039"
        ShowItemTreeCode="SHOWITEM"/>
</ShowItemFeeList>
```

Node Definitions

- Amount – The amount that will be charged for the item if the registrant matches the DateStatus, MemberStatus, and RegistrantStatus
- Status Fields – The registrant will need to match up against these fields in order to be eligible to purchase the item at the specified amount
    - DateStatus – represents a period of time within the show
    - MemberStatus – the membership status of the registrant
    - RegistrationStatus – the registration type of the registrant
- ShowItemCode – The code that uniquely identifies the session that the fee applies to
- ShowItemFeeId – A uniquely identifying field for this fee
- ShowItemTreeCode – The code that uniquely identifies the show item tree that the showitem falls under.

Registration Types

In any event, there could be many types that a registration can fall under. Their registration type can determine the sessions they are allowed to purchase, their prices, and in some cases, access to certain parts of an event. The RegTypeLookup object represents the different registration types that exist in a show.

RegTypeLookupList

```
<RegTypeLookupList CurrentPage="1" PageSize="100" ShowCode="SHO151">
        <RegTypeLookup IsActive="true" RegTypeCode="CN"
        RegTypeDescription="Conference A-la-Carte" RegTypeID="1006"/>
        <RegTypeLookup IsActive="true" RegTypeCode="DP" RegTypeDescription="Denied
        Press" RegTypeID="1010"/>
        <RegTypeLookup IsActive="true" RegTypeCode="EO" RegTypeDescription="Expo
        Only Pass" RegTypeID="1001"/>
        <RegTypeLookup IsActive="true" RegTypeCode="EX"
        RegTypeDescription="Exhibitor" RegTypeID="1011"/>
        <RegTypeLookup IsActive="true" RegTypeCode="FC" RegTypeDescription="Full
        Conference Pass" RegTypeID="1002"/>
</RegTypeLookupList>
```

Node Definitions

- RegTypeLookup
    - IsActive – Represents whether or not this registration type is active and available for purchase
    - RegTypeCode – This is the value that is seen on the Registrant Object that links the Registrant with their registration type.
    - RegTypeDescription – A short description of the registration type
    - RegTypeID – Unique identifier for the RegType

Speaker

Some sessions such as workshops contain speakers. If a session includes a speaker, that speaker is listed in the session object as shown above.

SpeakerList

```
<SpeakerList CurrentPage="1" PageSize="100" ShowCode="SHO151">
        <Session Address="" Address2="" BusinessTitle="" CertificationTitle="" City=""
        Company="" CountryCode="" Email="" FirstName="" ImageURL="" IsActive="true"
        LastName="" Middle="" Phone="" Prefix="" SocialBio="" SpeakerAffiliation=""
        SpeakerID="1000" SpeakerName="Test Me" StateCode="" Suffix="" ZipCode=""/>
        <Session Address="" Address2="" BusinessTitle="" CertificationTitle="" City=""
        Company="" CountryCode="" Email="" FirstName="" ImageURL="" IsActive="true"
        LastName="" Middle="" Phone="" Prefix="" SocialBio="" SpeakerAffiliation=""
```

```
        SpeakerID="1001" SpeakerName="Arnold Schwarzenegger" StateCode="" Suffix=""
        ZipCode=""/>
        <Session Address="" Address2="" BusinessTitle="" CertificationTitle="" City=""
        Company="" CountryCode="" Email="" FirstName="" ImageURL="" IsActive="true"
        LastName="" Middle="" Phone="" Prefix="" SocialBio="" SpeakerAffiliation=""
        SpeakerID="1002" SpeakerName="Testy McTest" StateCode="" Suffix=""
        ZipCode=""/>
</SpeakerList>
```

Node Definitions

- Session
    - Address, Address2, City, Company, CountryCode, Email, FirstName, LastName, Middle, Phone, Prefix, Suffix, SpeakerName – General contact information for the speaker
    - BusinessTitle – The Speaker's title
    - CeritificationTitle – Certification held by the speaker

Initializing Paged Pulls

Before beginning a paged pull, a call to one of the four initialization procedures will need to be completed. The procedures will return a token that is then used when calling the desired Pull method. In order to get the correct set of data, an export data type will need to be provided to the procedure.

The Supported ExportDataTypes are:

- RegistrantWithHousing
- Registrant
- ExhibitingCompany
- Exhibitor
- DemographicLookup
- Membership
- Session
- RegTypeLookup
- Speaker
- Status

It is also possible to order the data that is returned. This is done by passing the fieldname that the result is to be sorted by into the orderByClause parameter of the request. For

example, if we wished to have the results of the PullRegistrantList method to be sorted by first name, we would pass “FirstName” into the orderByClause parameter.

Once the token that is returned by these procedures has been used in the desired paged pull, the pull will need to be finalized using the FinalizePagedPull method. This method will end the data export and remove any references to the token.

InitializePagedPull

When using this initialization procedure, cancelled records will always be filtered out. An example of a request and response to this initialization procedure is provided below.

```
POST /RealTimeServices/Export.asmx HTTP/1.1
Host: ews.exl.mge360.com
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Header>
    <DataExportHeader xmlns="http://expoexchange.com/realtimeservices/">
      <HeaderShowCode>string</HeaderShowCode>
      <HeaderUsername>string</HeaderUsername>
      <HeaderPassword>string</HeaderPassword>
      <HeaderSQLEnvironment>PROD or QA or DEV</HeaderSQLEnvironment>
      <PagedResultsPageSize> int </PagedResultsPageSize>
      <FTPUsername>string</FTPUsername>
      <FTPFileExportType> None or CSV</FTPFileExportType>
      <UserAccountDomain>string</UserAccountDomain>
      <UseUTCFormat>string</UseUTCFormat>
    </DataExportHeader>
  </soap12:Header>
  <soap12:Body>
    <InitializePagedPull xmlns="http://expoexchange.com/realtimeservices/">
      <dataType> CustomCodeLookup or DemographicLookup or ExhibitingCompany or
Member or Messages or Registrant or RegistrantWithHousing or RFID or
RegTypeLookup or Searches or Session or Show or ShowSchedule or
ShowScheduleByClientID or EXSEPAccessURL or ShowCore or Exhibitor or
DialCustomerServiceInfo or Confirmation or Status or ShowItemFee or
RFRegistrantTag or Facility or Speaker </dataType>
      <BeginDate>dateTime</BeginDate>
      <EndDate>dateTime</EndDate>
      <orderByClause>string</orderByClause>
    </InitializePagedPull>
  </soap12:Body>
</soap12:Envelope>
```

```
HTTP/1.1 200 OK
```

```
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <InitializePagedPullResponse
     xmlns="http://expoexchange.com/realtimeservices/">
      <InitializePagedPullResult>
          <InitializePagedPull ShowCode="SHO151">
                <TokenInfo PageSize="100" PageToken="4A97DBD4-DB5B-4C05-AA24-
7161D6EFC435" TotalPages="2" TotalRecords="153"/>
          </InitializePagedPull>
     </InitializePagedPullResult>
    </InitializePagedPullResponse>
  </soap12:Body>
</soap12:Envelope>
```

ExportDataType: RegistrantWithHousing

If the RegistrantWithHousing DataType is used, the above would include the RegistrantWithHousing object from the above section. This data type will return registrants whose registration state is COM or Housing state is COM. This means that there could potentially be registrants returned who have an in progress registration or housing state. Cancelled registrants and registrants that were not updated between the provided Begin and End date will also be filtered out.

The Registrant will be filtered to only registrant records that meet the following criteria:

Registrant.IsCancelled = 0
AND (Registrant.RegState = 'COM' OR Registrant.HSNState = 'COM')
AND Registrant.UpdateDate >= {Begin Date Parameter}
AND Registrant.UpdateDate <= {End Date parameter}

ExportDataType: Registrant

If the Registrant Data Type is used, the above response would include the Registrant object. Registrants whose registration state is not COM will be filtered out as well as registrants that are cancelled.

Similar to the above, using this Initialization procedure along with this data type will cause registrant records to be filtered by this criteria:

Registrant.IsCancelled = 0
AND Registrant.RegTypeCode <> ''

AND Registrant.RegState = 'COM'
AND Registrant.UpdateDate >= {Begin Date Parameter}
AND Registrant.UpdateDate <= {End Date parameter}

InitializePagedPullRegistrant

Cancelled registrants can be returned if the AllowCancelled parameter is set to true. Below is an example of a request and response to this initialization procedure. This initialization procedure can only be used with the Registrant or RegistrantWithHousing DataTypes.

```
POST /RealTimeServices/Export.asmx HTTP/1.1
Host: ews.exl.mge360.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction:
"http://expoexchange.com/realtimeservices/InitializePagedPullRegistrant"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <DataExportHeader xmlns="http://expoexchange.com/realtimeservices/">
      <HeaderShowCode>string</HeaderShowCode>
      <HeaderUsername>string</HeaderUsername>
      <HeaderPassword>string</HeaderPassword>
      <HeaderSQLEnvironment>PROD or QA or DEV</HeaderSQLEnvironment>
      <PagedResultsPageSize>int</PagedResultsPageSize>
      <FTPUsername>string</FTPUsername>
      <FTPFileExportType>None or CSV</FTPFileExportType>
      <UserAccountDomain>string</UserAccountDomain>
      <UseUTCFormat>string</UseUTCFormat>
    </DataExportHeader>
  </soap:Header>
  <soap:Body>
    <InitializePagedPullRegistrant
xmlns="http://expoexchange.com/realtimeservices/">
      <dataType> Registrant or RegistrantWithHousing </dataType>
      <BeginDate>dateTime</BeginDate>
      <EndDate>dateTime</EndDate>
      <AllowCancelled>boolean</AllowCancelled>
      <orderByClause>string</orderByClause>
    </InitializePagedPullRegistrant>
  </soap:Body>
</soap:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <InitializePagedPullRegistrantResponse
xmlns="http://expoexchange.com/realtimeservices/">
      <InitializePagedPullRegistrantResult>
            <InitializePagedPull ShowCode="SHO151">
              <TokenInfo PageSize="100" PageToken="4A97DBD4-DB5B-4C05-AA24-
7161D6EFC435" TotalPages="2" TotalRecords="153"/>
          </InitializePagedPull>
      </InitializePagedPullRegistrantResult>
    </InitializePagedPullRegistrantResponse>
  </soap:Body>
</soap:Envelope>
```

ExportDataType: RegistrantWithHousing

If the RegistrantWithHousing DataType is used, the above would include the RegistrantWithHousing object from the above section. This data type will return registrants whose registration state is COM or Housing state is COM. This means that there could potentially be registrants returned who have an in progress registration or housing state. Cancelled registrants can be returned if AllowCancelled is set to true.

ExportDataType: Registrant

If the Registrant Data Type is used, the above response would include the Registrant object. Registrants whose registration state is not COM will be filtered out. However, as above, if the AllowCancelled field is set to true, cancelled registrants will be returned.

InitializeFilteredPagedPull

The InitializeFilteredPagedPull procedure provides the same service as the others, but with the ability to filter records based on the whereClause.

For Example, if we wanted to filter on registrants whose name began with the letter “A”, the term “FirstName like ‘A%’” could be passed into the whereClause parameter.

```
  ...
<soap:Body>
  <InitializeFilteredPagedPull
xmlns="http://expoexchange.com/realtimeservices/">
    <dataType>CustomCodeLookup or DemographicLookup or ExhibitingCompany or
Member or Messages or Registrant or RegistrantWithHousing or RFID or
RegTypeLookup or Searches or Session or Show or ShowSchedule or
ShowScheduleByClientID or EXSEPAccessURL or ShowCore or Exhibitor or
```

```
DialCustomerServiceInfo or Confirmation or Status or ShowItemFee or
RFRegistrantTag or Facility or Speaker</dataType>
    <BeginDate>dateTime</BeginDate>
    <EndDate>dateTime</EndDate>
    <whereClause>string</whereClause>
    <orderByClause>string</orderByClause>
  </InitializeFilteredPagedPull>
</soap:Body>
```

```
...
<soap:Body>
  <InitializeFilteredPagedPullResponse
xmlns="http://expoexchange.com/realtimeservices/">
    <InitializeFilteredPagedPullResult>
      <InitializePagedPull ShowCode="SHO151">
        <TokenInfo PageSize="100" PageToken="4A97DBD4-DB5B-4C05-AA24-7161D6EFC435"
TotalPages="2" TotalRecords="153"/>
      </InitializePagedPull>
    </InitializeFilteredPagedPullResult>
  </InitializeFilteredPagedPullResponse>
</soap:Body>
```

InitializeXMLFilteredPagedPull

There is also a method to filter to specific records based on snippets of XML. The InitializeXMLFilteredPagedPull returns a token and page information for records that meet the criteria set by the xmlFragment parameter.

For Example, if the fragment "<FirstName>John</FirstName>" was passed into the xmlFragment parameter, the registrant pull would only return registrants with the first name of "John".

```
...
<soap:Body>
  <InitializeXMLFilteredPagedPull
xmlns="http://expoexchange.com/realtimeservices/">
    <dataType>CustomCodeLookup or DemographicLookup or ExhibitingCompany or
Member or Messages or Registrant or RegistrantWithHousing or RFID or
RegTypeLookup or Searches or Session or Show or ShowSchedule or
ShowScheduleByClientID or EXSEPAccessURL or ShowCore or Exhibitor or
DialCustomerServiceInfo or Confirmation or Status or ShowItemFee or
RFRegistrantTag or Facility or Speaker</dataType>
    <BeginDate>dateTime</BeginDate>
    <EndDate>dateTime</EndDate>
    <xmlFragment>string</xmlFragment>
    <orderByClause>string</orderByClause>
  </InitializeXMLFilteredPagedPull>
</soap:Body>
```

```
...
```

```
<soap:Body>
  <InitializeXMLFilteredPagedPullResponse
xmlns="http://expoexchange.com/realtimeservices/">
    <InitializeXMLFilteredPagedPullResult>
      <InitializePagedPull ShowCode="SHO151">
<TokenInfo PageSize="100" PageToken="4A97DBD4-DB5B-4C05-AA24-7161D6EFC435"
TotalPages="2" TotalRecords="153"/>
      </InitializePagedPull>
    </InitializeXMLFilteredPagedPullResult>
  </InitializeXMLFilteredPagedPullResponse>
</soap:Body>
```

Data Pulls

Registrant data pulls can be used to pull Registrants that fit the criteria provided. When using the Registrant List methods, one of the initialize paged pull methods from above will first need to be used to retrieve a token and page information. These methods then take this token and the current page desired and return the list of Registrants that fit the criteria. If using a method that pulls by ID, the only parameter that is needed is the ID of the record that is being requested.

Paged Registrant Pulls

As shown below in the PullRegistrantListWithHousing method, calls to these paged Registrant pulls will need to be provided with a BeginDate, EndDate, token, and current page to pull. Only registrants that have been updated between the Begin and End dates provided will be returned.

```
...
<soap:Body>
  <PullRegistrantListWithHousing
xmlns="http://expoexchange.com/realtimeservices/">
    <pageToken>string</pageToken>
    <currentPage>int</currentPage>
  </PullRegistrantListWithHousing>
</soap:Body>
```

```
...
<soap:Body>
  <PullRegistrantListWithHousingResponse
xmlns="http://expoexchange.com/realtimeservices/">
    <PullRegistrantListWithHousingResult>
      <RegistrantList>
        <Registrant>
          ...
        </Registrant>
        ...
```

```
      </RegistrantList>
    </PullRegistrantListWithHousingResult>
  </PullRegistrantListWithHousingResponse>
</soap:Body>
```

PullRegistrantListWithHousing

This paged pull method will return registrants whose registration state is COM or Housing state is COM. This means that there could potentially be registrants returned who have an in progress registration or housing state. Cancelled registrants can be returned if AllowCancelled is set to true.

PullRegistrantList

Registrants whose registration state is not COM will be filtered out. However, as above, if the AllowCancelled field is set to true, cancelled registrants will be returned.

Registrant Transaction Data

Individual Registrant records can also be returned. These methods are not paged and do not require a call to an initialization procedure. They accept one parameter, the registrantid to be returned.

PullByRegistrantID or PullByRegistrantIDWithHousing

These methods allow a single registrant to be pulled. The registrant pulled back is the registrant whose RegistrantID field matches the ID that is passed in the request. These methods will return registrant information even if the registrant is in progress (INP). Below is an example of a request and response from the PullByRegistrantID method.

```
...
<soap:Body>
  <PullRegistrantByID xmlns="http://expoexchange.com/realtimeservices/">
    <registrantID>int</registrantID>
  </PullRegistrantByID>
</soap:Body>
```

```
...
<soap:Body>
  <PullRegistrantByIDResponse
xmlns="http://expoexchange.com/realtimeservices/">
    <PullRegistrantByIDResult>
      <RegistrantList>
        <Registrant>
          ...
        </Registrant>
        ...
      </RegistrantList>
```

```
    </PullRegistrantByIDResult>
  </PullRegistrantByIDResponse>
</soap:Body>
```

Searching by ID is not the only way to find specific registrants using RTS. There are even more Registrant specific pull methods including:

- PullRegistrantByMemberNumber
  - Takes a member number as a parameter and returns any registrants that have the number in their record. The formatting of this number may contain both characters and digits. For Example, "AAA-000" may be a valid member number.
- PullRegistrantByEmail
  - Takes Email as a parameter and returns any registrants that match.

Other Registrant pull methods include:

- PullRegistrantByConstID (Depends on whether or not the show uses this field)
- PullRegistrantRFIDReadsByID (For shows using RFID)
- PullRFRegistrantTagList (For shows using RFID)

Show Setup Methods

Real Time Services provide a suite of setup pull methods that allow the ability to look up show setup information such as demographics, sessions, registration types, exhibiting companies, and speakers. This allows the two systems to remain in sync when it comes to registrant responses and purchases.

PullDemographicLookupList

Used alongside an InitializePagedPull method, this takes a token and a page as its parameters and will return a DemographicLookupList object. Freeform fields will not be included in the results of this pull.

```
...
<soap:Body>
  <PullDemographicsLookupList
xmlns="http://expoexchange.com/realtimeservices/">
    <pageToken>string</pageToken>
    <currentPage>int</currentPage>
  </PullDemographicsLookupList>
</soap:Body>
```

```
...
```

```
<soap:Body>
  <PullDemographicsLookupListResponse
xmlns="http://expoexchange.com/realtimeservices/">
    <PullDemographicsLookupListResult>
      <DemographicsLookupList>
          ...
      </DemographicsLookupList>
    </PullDemographicsLookupListResult>
  </PullDemographicsLookupListResponse>
</soap:Body>
```

PullDemographicLookupListWithFreeForm

Unlike the PullDemographicLookupList method, freeform (open text) fields are included. This method does not take any parameters in order to pull information back.

PullMembershipList

In order to retrieve information regarding a registrant's membership, the PullMembershipList can be used. This method will return a list of Membership objects that contain a MemberNumber (which is held in Registrant records that have used this membership), and contact and MemberType information. The member objects returned will be members that have been updated between the given BeginDate and EndDate.

```
...
<soap:Body>
    <PullMembershipList xmlns="http://expoexchange.com/realtimeservices/">
      <BeginDate>dateTime</BeginDate>
      <EndDate>dateTime</EndDate>
      <pageToken>string</pageToken>
      <currentPage>int</currentPage>
    </PullMembershipList>
</soap:Body>
```

```
<soap:Body>
    <PullMembershipListResponse
xmlns="http://expoexchange.com/realtimeservices/">
      <PullMembershipListResult>
            <MembershipList>
               ...
            </MembershipList>
      </PullMembershipListResult>
    </PullMembershipListResponse>
</soap:Body>
```

PullSessionList

The PullSessionList pull method takes a token and page as parameters and returns a list of all of the Sessions/ShowItems that exist in the show. The information provided in these

objects include the ShowItemCode that is shown in the registrant object, a description, start and end dates, quantity, quantity used and more.

```
...
<soap:Body>
    <PullSessionList xmlns="http://expoexchange.com/realtimeservices/">
      <pageToken>string</pageToken>
      <currentPage>int</currentPage>
    </PullSessionList>
</soap:Body>
```

```
...
<soap:Body>
    <PullSessionListResponse
xmlns="http://expoexchange.com/realtimeservices/">
      <PullSessionListResult>
            <SessionList>
               ...
            </SessionList>
       </PullSessionListResult>
    </PullSessionListResponse>
</soap:Body>
```

PullRegTypeLookupList

On the Registrant object, there is a field for RegTypeCode. The RegTypeLookupList pull method returns a list of all of the registration types that exist in the show, providing a description to go along with each RegTypeCode and a field, "IsActive", that determines whether the Registration Type is currently in use in the show.

```
...
<soap:Body>
    <PullRegTypeLookupList xmlns="http://expoexchange.com/realtimeservices/">
      <pageToken>string</pageToken>
      <currentPage>int</currentPage>
    </PullRegTypeLookupList>
</soap:Body>
```

```
...
<soap:Body>
    <PullRegTypeLookupListResponse
xmlns="http://expoexchange.com/realtimeservices/">
      <PullRegTypeLookupListResult>
            <RegTypeLookupList>
                  ...
            </RegTypeLookupList>
      </PullRegTypeLookupListResult>
    </PullRegTypeLookupListResponse>
</soap:Body>
```

PullExCompanyList

Returns a list of ExCompany objects when given a BeginDate, EndDate, token and page. Only companies that have been updated between these given dates will be returned. The ExCompany object gives contact and booth information about each company.

```
...
<soap:Body>
    <PullExCompanyList xmlns="http://expoexchange.com/realtimeservices/">
      <BeginDate>dateTime</BeginDate>
      <EndDate>dateTime</EndDate>
      <pageToken>string</pageToken>
      <currentPage>int</currentPage>
    </PullExCompanyList>
</soap:Body>
```

```
...
<soap:Body>
    <PullExCompanyListResponse
xmlns="http://expoexchange.com/realtimeservices/">
      <PullExCompanyListResult>
            <ExCompanyList>
                  ...
            </ExCompanyList>
      </PullExCompanyListResult>
    </PullExCompanyListResponse>
</soap:Body>
```

PullExhibitorList

An Exhibitor is a registrant that is registered under an Exhibiting Company. They are linked with their company by their ExCompanyID inside the registrant record. This pull method will take a token and page and will return a RegistrantList object of only registrants that have a value in the ExCompanyID field.

PullSpeakerList

Returns a list of Speaker objects that have been updated between the begin and end date that were provided. The Speaker object is returned based on the token and page that is passed into the request.

```
...
<soap:Body>
    <PullSpeakerList xmlns="http://expoexchange.com/realtimeservices/">
      <pageToken>string</pageToken>
      <currentPage>int</currentPage>
    </PullSpeakerList>
</soap:Body>
```

```
...
```

```
<soap:Body>
    <PullSpeakerListResponse
xmlns="http://expoexchange.com/realtimeservices/">
      <PullSpeakerListResult>
           <SpeakerList>
             ...
           </SpeakerList>
      </PullSpeakerListResult>
    </PullSpeakerListResponse>
</soap:Body>
```

PullStatusList

Provided with a token and page, this pull method returns a list of status objects that represent membership types, registration types, and time periods. These objects help drive the registration process and the fees that are associated with show items and registration.

```
...
<soap:Body>
    <PullStatusList xmlns="http://expoexchange.com/realtimeservices/">
      <pageToken>string</pageToken>
      <currentPage>int</currentPage>
    </PullStatusList>
</soap:Body>
```

```
...
<soap:Body>
    <PullStatusListResponse
xmlns="http://expoexchange.com/realtimeservices/">
      <PullStatusListResult>
          <StatusList>
            ...
          </StatusList>
      </PullStatusListResult>
    </PullStatusListResponse>
</soap:Body>
```

Finalizing Paged Pulls

After a paged pull has been completed, it will need to be finalized. The finalization process will end the paged data export and will remove the references to the token that was created by the initialization method that was called.

FinalizePagedPull

In order to finalize a paged pull, the FinalizePagedPull method will need to be called. This method accepts the token that was created as its parameter and will return a success or

error message depending on whether or not the finalization process went smoothly. Below is an example of how a request to this method would look and the results that would be returned on success or failure.

```
...
<soap:Body>
  <FinalizePagedPull xmlns="http://expoexchange.com/realtimeservices/">
    <pageToken>string</pageToken>
  </FinalizePagedPull>
</soap:Body>
```

On Success:

```
<soap:Body>
  <FinalizePagedPullResponse
xmlns="http://expoexchange.com/realtimeservices/">
    <FinalizePagedPullResult>
      <FinalizePagedPull ShowCode="AFH557">
        <Success></Success>
      </FinalizePagedPull>
    </FinalizePagedPullResult>
  </FinalizePagedPullResponse>
</soap:Body>
```

On Failure:

```
<soap:Body>
  <FinalizePagedPullResponse
xmlns="http://expoexchange.com/realtimeservices/">
    <FinalizePagedPullResult>
      <FinalizePagedPull ShowCode="AFH557">
        <Errors>
          <Error Type="Unknown">Message goes here</Error>
        </Errors>
      </FinalizePagedPull>
    </FinalizePagedPullResult>
  </FinalizePagedPullResponse>
</soap:Body>
```