****

**Redbox**

OpenAPI – Release 2

Architecture Design

Part II – Design

**Author**: *Chris Rudolphi*

**Document Title**: OpenAPI R2 Architecture Design v1 DRAFT 13

**Updated**: *5/9/2011 8:38 AM*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Reviewer and Approvers** | | | | |
| **Name** | **Title/Role** | **Reviewer/**  **Approver** | **Approval**  **Date** | **Approval Method** |
| Saad Rehmani | Director of Architecture | Approver |  |  |
| Gunhan Tatman | VP of Development | Approver |  |  |
| Bart Dolega | Director of Platform Services | Approver |  |  |
| Michael Georgoff | Director of Business Development | Reviewer |  |  |
| Eric Buckner | Product Manager | Reviewer |  |  |
| Viju Hullur | SA for OpenAPI | Reviewer |  |  |
| Imran Maskatia | Digital Product Manager | Reviewer |  |  |

[1. Introduction 5](#_Toc292693578)

[1.1 Project Description 5](#_Toc292693579)

[1.2 Description and Scope of this Document 5](#_Toc292693580)

[2. Design 6](#_Toc292693581)

[2.1 Products Service 6](#_Toc292693582)

[2.2 Stores Service 9](#_Toc292693583)

[2.3 Top20 Service 11](#_Toc292693584)

[2.4 Inventory Service 12](#_Toc292693585)

[2.5 Reservation Service 14](#_Toc292693586)

[2.6 Customer Account Service 18](#_Toc292693587)

[2.7 Major Solution Components 21](#_Toc292693588)

[2.7.1 Products Gateway 21](#_Toc292693589)

[2.7.2 Kiosk (“Stores”) Gateway 22](#_Toc292693590)

[2.7.3 Top 20 Gateway 22](#_Toc292693591)

[2.7.4 Inventory Gateway 23](#_Toc292693592)

[2.7.5 Reservations Gateway 24](#_Toc292693593)

[2.7.6 Customer Account Gateway 24](#_Toc292693594)

[2.7.7 CreditCard Services Gateway 26](#_Toc292693595)

[2.7.8 Trailers Gateway 26](#_Toc292693596)

[2.7.9 Data Sources 27](#_Toc292693597)

[2.7.10 Cache Refresh Batch Jobs 28](#_Toc292693598)

[2.7.11 Change Notifiers 28](#_Toc292693599)

[2.7.12 Source System Change Service 29](#_Toc292693600)

[2.7.13 Cache Refresh Message Handlers 29](#_Toc292693601)

[2.7.14 Item Changed Message Handlers 29](#_Toc292693602)

[2.7.15 Apigee Components 30](#_Toc292693603)

[2.7.16 EPC Web Service 53](#_Toc292693604)

[2.7.17 Batch Scheduler 53](#_Toc292693605)

[2.7.18 Web Reservation Pipeline 53](#_Toc292693606)

[2.7.19 Caching Services 53](#_Toc292693607)

[2.7.20 Web Services Monitoring 53](#_Toc292693608)

[2.8 Physical Deployment 54](#_Toc292693609)

[2.9 Database Logical Design 56](#_Toc292693610)

[2.10 Open Systems Layer: Infrastructure Components & Dependencies 57](#_Toc292693611)

[2.10.1 Custom-built .NET Web Services (Product, Store, Top20 proxies) 57](#_Toc292693612)

[2.10.2 Source System Change Service 57](#_Toc292693613)

[2.10.3 Logging & Instrumentation 57](#_Toc292693614)

[2.11 Design Approaches to Non-Functional Qualities (availability, performance, scaling) 58](#_Toc292693615)

[2.11.1 Availability 58](#_Toc292693616)

[2.11.2 Performance 58](#_Toc292693617)

[2.11.3 Scale 58](#_Toc292693618)

[2.12 Security 59](#_Toc292693619)

[2.12.1 OpenAPI Service Endpoint Security 59](#_Toc292693620)

[2.12.2 Open Services Layer – Source System Gateway Security 59](#_Toc292693621)

[2.13 Dependencies & Assumptions 60](#_Toc292693622)

[3. Open Issues 61](#_Toc292693623)

[4. Risks 63](#_Toc292693624)

[5. Glossary 64](#_Toc292693625)

[6. Appendices 65](#_Toc292693626)

[6.1 Appendix A: Message Contracts & Schemas 65](#_Toc292693627)

[6.1.1 Common Schemas 65](#_Toc292693628)

[6.1.2 Source Movies Schema 67](#_Toc292693629)

[6.1.3 External Movie Schema 71](#_Toc292693630)

[6.1.4 External Products Browse Request Schema 75](#_Toc292693631)

[6.1.5 Product Search Request 76](#_Toc292693632)

[6.1.6 Paginated Product List 78](#_Toc292693633)

[6.1.7 Titles 81](#_Toc292693634)

[6.1.8 RecommendedProductIDList 87](#_Toc292693635)

[6.1.9 Source Stores Schema 88](#_Toc292693636)

[6.1.10 External Stores Schema 90](#_Toc292693637)

[6.1.11 Common External Store Schema 92](#_Toc292693638)

[6.1.12 Source Inventory List 93](#_Toc292693639)

[6.1.13 External Store Lookup Response Schema 95](#_Toc292693640)

[6.1.14 InventoryLookup Schema 97](#_Toc292693641)

[6.1.15 Source Top20 Schema 99](#_Toc292693642)

[6.1.16 Store and Inventory Search Result Schema 101](#_Toc292693643)

[6.1.17 External Top20 Schema 103](#_Toc292693644)

[6.1.18 Cart Schema 107](#_Toc292693645)

[6.1.19 CartValidation Schema 108](#_Toc292693646)

[6.1.20 PricedCart Schema 110](#_Toc292693647)

[6.1.21 PricedCartResponse Schema 112](#_Toc292693648)

[6.1.22 ReservationResponse Schema 115](#_Toc292693649)

[6.1.23 BasicProfile Schema 118](#_Toc292693650)

[6.1.24 Preferences 120](#_Toc292693651)

[6.1.25 RentalHistory 122](#_Toc292693652)

[6.1.26 QueuesListResponse 123](#_Toc292693653)

[6.1.27 QueueInfo 124](#_Toc292693654)

[6.1.28 RecommendedProductsResponse 125](#_Toc292693655)

[6.1.29 AccountListResponse 126](#_Toc292693656)

[6.1.30 AccountInfo 127](#_Toc292693657)

[6.1.31 CreditBalance 128](#_Toc292693658)

[6.1.32 CreditsAvailable 129](#_Toc292693659)

[6.1.33 CreditUseHistory 130](#_Toc292693660)

[6.1.34 CreditCardInfo 132](#_Toc292693661)

[6.1.35 CacheRefreshCommand Message Schema 133](#_Toc292693662)

[6.1.36 ProductsCacheRefreshCommand Message Schema 134](#_Toc292693663)

[6.1.37 StoresCacheRefreshCommand Message Schema 135](#_Toc292693664)

[6.1.38 Top20CacheRefreshCommand Message Schema 136](#_Toc292693665)

[6.1.39 InventoryCacheRefreshCommandMessage Schema 137](#_Toc292693666)

[6.1.40 ItemChangedEventMessage Schema 138](#_Toc292693667)

[6.1.41 StoreChangedEventMessage Schema 139](#_Toc292693668)

[6.1.42 InventoryChangedEventMessage Schema 140](#_Toc292693669)

[6.1.43 ItemChangesMessage Schema 141](#_Toc292693670)

[6.1.44 StoreChangesMessage Schema 142](#_Toc292693671)

[6.1.45 InventoryChangesMessage Schema 144](#_Toc292693672)

[6.1.46 MovieTrailers Schema 145](#_Toc292693673)

[7. Change Log 147](#_Toc292693674)

# Introduction

## Project Description

The purpose of the Redbox application programming interface (API) is to leverage Redbox information and feature assets into an interface that will allow business partners and affiliates to integrate Redbox with partner and affiliate web properties.

## Description and Scope of this Document

This document describes the architectural design for implementing OpenAPI. This document will provide an overview of the solution by describing each major component, the data flows among components, the sequencing of key activities, and the data that will be used by OpenAPI. This design document represents the detailed architecture design and is based upon its sister document, *Architecture Design Part I - Conceptual Overview*.

The design is driven by requirements as documented in the following OpenAPI deliverable:

* OpenAPI Product Requirements Document (v2): <https://boxoffice.redbox.com/digital/Shared%20Documents/Open%20API/Definition%20and%20Requirements/Release%202%20(Digital)/Open%20API%20Release%202%20(Digital)%20v1.1%20BPS.docx>

The scope of this document covers:

* Exposing Product (Title) information via the API
* Exposing Top20 Product Rental statistics via the API
* Exposing Kiosk information via the API
* Exposing Inventory
* Transacting Reservations via the API
* Exposing Customer/Account profile information
* Integration with Apigee as a cloud-proxy for the API
* Mechanisms for tracking traffic

This document does NOT describe:

* The design of how Redbox.com will be modified to provide the required pre-filled cart landing page
* The design of any changes required to EPC to fulfill the needs of OpenAPI

The following sections describe the application of this pattern for the services required in R2 of the OpenAPI.

# Design

## Products Service

The Products Service will provide to information about Movies in the Redbox catalog, as depicted in Figure 2.1‑1 Products Service. The major components of the Products Service include:

* The Internal Source System will be the Enterprise Product Catalog Service (EPC)
* EPC data will be supplemented by product circulation status from the ODS
* ‘Similar Titles’ recommendations will be provided by calling a Website service
* Trailer metadata will be provided by BrightCove
* The Products Gateway (described further in 2.7.1)
* The Trailers Gateway (described in 2.7.8)
* The Products Cache Refresh Job (see 2.7.10.1)
* The Products Cache Refresh Message Handler (see 2.7.13.1)
* The Products Service Endpoint in the Apigee Layer (see 2.7.15.1.1)

The Products Service provides the following operations:

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Description** | **Input** | **Output** |
| P0 | Status Monitor – called by the Redbox service monitoring infrastructure to confirm the operation of the service. | None | Status |
| P1 | Retrieve a bulk listing of all ‘current’ Movie product metadata. | Optional flag to include Coming Soon movies | Products (Bulk list, younger than 250 days with ‘Display On Website’ set) |
| P2 | Retrieves a bulk listing of all ‘in-circulation’ Movie product metadata | None | Metadata for all products in circulation |
| P3 | Retrieve a list of ‘Coming Soon’ product metadata | None | Metadata for all ‘coming soon’ products |
| P4 | Retrieve Movie product metadata for a specific set of Products. | MovieBrowseRequest (list of Product Ids) | MovieBrowseResponse |
| P5 | Retrieve product metadata about a single Product | ProductId ID | MovieBrowseResponse |
| P6 | Search the Product catalog | ProductSearchRequest:  Search Criteria  Sort Criteria  Paging Info | Paginated Product Search Response |
| P7 | Retrieve Title metadata (multiple formats collapsed into a single item) | Optional flag to include Coming Soon movies | Titles (Bulk list, younger than 250 days with ‘Display On Website’ set) |
| P8 | Retrieve Coming Soon Title metadata (multiple formats collapsed into a single item) | None | Coming Soon Titles |
| P9 | Provide Recommendations of similar titles | Product ID | Similar Titles metadata |
| P13 | Retrieve Trailer Information | Product ID | Trailer metadata |

The Products Service will process changes in bulk from EPC on an hourly basis. There is no need to process changes as they occur. Therefore, the *ItemChangedEvent* sequence is not used by this service.

*Caching*

Most of the above service operations will be executed at the Apigee layer using the contents of the Products Caches.

The Products Gateway will retrieve all Movies from EPC and filter that to those titles that are ‘in-circulation’ based upon title status data from the ODS. This will also include the ‘Coming Soon’ titles. This set of data will be provided to the Apigee layer, which will split this into three caches:

* the ‘Default Cache’: the set of products that make up the Default browsable catalog (younger than 250 days plus ‘Coming Soon’);
* One for all products, called the ‘In-Circulation Cache’;
* One for Titles

The operations will use the caches in the following way:

|  |  |
| --- | --- |
| **Operation** | **Cache** |
| P1 | Default Cache |
| P2 | In-Circulation Cache |
| P3 | Default Cache |
| P4 | In-Circulation Cache |
| P5 | In-Circulation Cache |
| P6 | In-Circulation Cache |
| P7 | Titles Cache |
| P8 | Titles Cache |
| P9 | In-Circulation Cache |
| P13 | Trailers Cache |

The URL of the R1 request has changed from /Products/Movies to /Products/Movies/Default.

A KEY difference from R1 to R2, is that the responsibility for filtering EPC data will change:

- The Products Gateway will filter EPC data to the set of Movies that are ‘in circulation’. This will be accomplished by consuming circulation status from the ODS and using it to filter the EPC data. This filtered set of EPC data will be pushed to Apigee. The ‘Source Schema’ for the Products Gateway has been changed to include the ‘Display On Website’ flag so that this may be shared with Apigee.

- For the Bulk Product Listing, the Apigee layer will filter the Products to those that are younger than 250 days old and have the ‘Display On Website’ flag set.

- The Apigee Layer will maintain two caches of Product Data, one for each type of time ‘window’.

- Added a call to BrightCove to retrieve metadata about trailers.



Figure ‑ Products Service

## Stores Service

The Stores Service will provide read-only information about the Stores (kiosks), as depicted in Figure 2.2‑1 Stores Service. The major components of the Stores Service include:

* The Internal Source System will be the Kiosk table of the ODS.
* Stores Change Notifier (see 2.7.11.1)
* The Stores Gateway (described further in 2.7.2)
* The Inventory Gateway (see 2.7.4 )
* The Stores Cache Refresh Job (see 2.7.10.2)
* The Stores Cache Refresh Message Handler (see 2.7.13.2)
* The Store Changed Message Handler (see 2.7.14.1)
* The Stores Service in the Apigee Layer (see 2.7.15.1.2)

The service operations for the Stores Service include:

* Bulk Stores List
* Store Info by Store ID
* Store Search by Lat/Long

This service will provide:

* a bulk output; a Store List (schema described in Appendix A, Section 6.1.10)
* a store lookup result (schema described in Section 6.1.13)

The Stores Service endpoint (Apigee layer) will be responsible for resolving location requests by finding the closest stores to a given location. {In other words, all of these requests are fulfilled at the Apigee layer, none result in requests going all the way back to Redbox.} All but the first of these operations (Bulk Stores List) will provide support for paging output.

| **ID** | **Description** | **Input** | **Output** |
| --- | --- | --- | --- |
| S0 | Status Monitor – called by the Redbox service monitoring infrastructure to confirm the operation of the service. | None | Status |
| S1 | Retrieve a bulk listing of all Kiosk metadata. | None | Stores |
| S2 | Retrieves paged Store metadata given StoreID(s). | List of StoreIDs to search for;  Banner name to filter;  Page Number and Page Size | Stores |
| S3 | Retrieves paged Store metadata given a location. | Lat/Long of stores to search for;  Radius to search;  Count of number of Stores to return;  Banner name to filter;  Sorting options;  Page Number and Page Size | Stores |

**NOTE**: While this design fully specifies the use of the item changed pattern for store status changes, the requriements to-date will allow for simply performing full refreshes of Stores data on a periodic basis. Therefore, the following components (while specified in the design) are out of scope for R2:

* Stores Change Notifier (see 2.7.11.1)
* StoreChangedEvent Queue
* The Store Changed Message Handler (see 2.7.14.1)
* The /StoreChanged operation of the Stores Gateway (2.7.2)



Figure ‑ Stores Service

The key differences between this release (R2) and the Bing (R1) release are:

• Store status changes will be recognized and processed as soon as possible via the Store ChangedEventMessage mechanism.

• Communication status is added as a property of the Store metadata provided to consumers.

• Added the ability to search for stores by StoredID.

• Added the ability to search for stores based upon location (lat/long).

## Top20 Service

The Top20 Service will provide a bulk listing of the Top20 Movies, as depicted in Figure 2.3‑1 Top20 Service. The major components of the Top20 Service include:

* The Internal Source System will be the Product Service in Redbox.com’s Website Services.
* The Top20 Gateway (described further in 2.7.3)
* The Top20 Cache Refresh Job (see 2.7.10.3)
* The Top20 Cache Refresh Message Handler (see 2.7.13.3)
* The Products Service in the Apigee Layer (see 2.7.15.1.1)

This service will provide one output: a Top20 List (schema described in Appendix A, Section 8.1.15).

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Description** | **Input** | **Output** |
| P10 | Retrieve a Top20 list | Period (7 or 30) | Top20List |

The Top20 Source System Gateway will retrieve information about rental volumes of products from the ODS. It will also access the Products Canonical Format Cache from the Products Gateway to retrieve the Products list. This will provide the Top20 Gateway with the correlation between the Product ID provided by ODS and the external Product Identifier that should be provided to the Apigee layer. The service will then create the Top20 list in its canonical format.



Figure ‑ Top20 Service

Major Changes:

-- When the ApiKey query string parameter belongs to a Digital partner, the response should include the partner Product ID for each movie.

## Inventory Service

The Inventory Service will provide inventory data, as depicted in Figure 2.4‑1 Inventory Service. The major components of the Inventory Service include:

* The Internal Source System will be the Inventory data exposed in the ODS.
* The Inventory Change Notifier (see 2.7.11.2)
* The Inventory Gateway (described further in 2.7.4)
* The Inventory Cache Refresh Job (see 2.7.10.4)
* The Inventory Cache Refresh Message Handler (see 2.7.13.4)
* The Inventory Changed Event Message Handler (see 2.7.14.2)
* The Inventory Endpoint Service in the Apigee Layer (see 2.7.15.1.3)

Inventory data changes frequently. The Inventory Service will use the ‘item changed event’ pattern as described in the Architectural Pattern’s description of the “Item Changed Event Sequence” (see *Architecture Desig, Part I – Conceptual Overview*). The ODS will send individual inventory change notifications to the Inventory Changed Event Queue. The queue handler will call the Inventory Gateway, which will update its internal cache and then notify the Inventory Endpoint in Apigee.

The Inventory API exposes the following operations:

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Description** | **Input** | **Output** |
| I0 | Status Monitor – called by the Redbox service monitoring infrastructure to confirm the operation of the service. | None | Status |
| I1 | Retrieve a paged listing of Inventory for a given set of stores. | A StoreList of the list of Store Ids for which Inventory is to be retrieved;  List of Product IDs for which Inventory status is to be retrieved;  Page Number and Page Size | Inventory Lookup |
| I2 | Retrieve a paged listing of Inventory by finding the stores near a location that carry a particular product (given an ID for that product). | Product ID;  Location (Lat/Long);  Store Count (max number of stores to return);  Radius (search radius in miles);  Sort options;  Page Number and Page Size | StoresInventory Lookup |

These Inventory API operations will be handled by the Apigee layer with data fromthe Apigee Stores and Inventory caches.

The output of the first operation, an inventory lookup, is a list, by StoreId, of the inventory status for each product that has been seen by that kiosk in the last 30 days. (Some product will be shown as ‘OutOfStock’). If the kiosk is Offline (as recorded in the Store Cache on Apigee), there will be no Inventory returned for that kiosk.

The second operation, I2, is a combination of a store lookup and inventory lookup. The primary use case is that a user has identified a single movie and would like to find which local kiosks carry the movie. The product (movie) will have an identifier. This operation is designed to flexibly accept a number of different forms of product identifiers. In a Digital subscriber scenario, the Digital partner application will provide the Digital partner product ID and the location of the customer (as Lat/Long). The output will include the store metadata (identical to the Store metadata provided by the Stores service) and the inventory status for the product in those locations.

For the Digital release (R2 of OpenAPI), this operation will support the use of Redbox Product ID (GUIDs) and Digital Partner IDs as means to identify products. In the future this might be expanded to include the use of UPCs, Baseline IDs and other forms of product identity.



Figure ‑ Inventory Service

## Reservation Service

The Reservation Service will provide the capability to submit reservations for physical discs, as depicted in Figure 2.5‑1 Reservation Service. The major components of the Reservation Service include:

* The Internal Source System will be the Pipeline Service of the Redbox.com application services.
* The Reservation Gateway (described further in 2.7.5)
* The Reservation Endpoint in Apigee (see 2.7.15.1.4)



Figure ‑ Reservation Service

The Reservation Service API will be conceptually similar to the reservation API exposed by the Redbox.com website. Applications will be able to add items to a shopping cart. The cart is associated with a single store. After adding item(s) to a cart, the service will validate and price the cart, returning sufficient information to allow the customer to confirm the reservation and proceed to Checkout. The application will then submit the cart to ‘Checkout’, at which time the cart will be revalidated, the credit card information authorized and the reservation transaction submitted to the Rental stack.

The Reservation Service will differ from the Redbox.com website in the following ways:

* The Reservation Service provides no mechanism to add a new credit card (this can be done through the Profile Service)
* The cart content will be passed between client & server with each request (to keep the server session-less which will help with scalability). Cart contents will not be committed to any databases until the Checkout.

The Reservation Service will not keep any Cart or Reservation data in cache and thus will not use any of the Cache Refresh mechanisms defined by the architectural pattern. However, it is possible that a product may go out of stock before a cart can be checked-out. The Web Reservation pipeline will respond with an error message when this happens and the Reservation Gateway will send an **Inventory Changed Event Message** to the **Inventory Changed Event Message Queue** (see section 2.7.14.2).

The following table summarizes the Reservation API’s structure.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Description** | **Processed By** | **Returns** |
| R0 | Status Monitor – called by the Redbox service monitoring infrastructure to confirm the operation of the service. | None | Status |
| R1 | Adds an item to the user's cart for a reservation. The body of the message contains a Cart to which the item should be added. | Validated by Apigee Layer | CartValidation |
| R2 | Removes an item from the user's cart. The body of the message contains a Cart from which the item should be removed. | Validated by Apigee Layer | CartValidation |
| R3 | Assigns a Store to the cart. The body of the message contains a Cart. The Store is assigned/changed to the given StoreID. Product inventory is re-validated when the Store is set/changed. | Validated by Apigee Layer | CartValidation |
| R4 | Validate the cart's current detail, including pricing, tax, etc (ensure it is ready for checkout) (executes the CartView Pipeline). The body of the message contains a Cart. | Pipeline Service (via Reservation Gateway) | PricedCart |
| R5 | Processes the checkout of the cart; return a Reservation ID (executes the Reservation Pipeline). The body of the message contains a PricedCart. | Pipeline Service (via Reservation Gateway) | ReservationResponse |

The Cart API provides specific API calls to add/delete products from the cart and to assign/change a Store. The API does NOT provide a specific call to assign a credit card. The application developer will populate a credit card ID in the cart to indicate which card should be used. The presumption is this card ID will be retrieved separately by using the Customer Account Service.

The Cart API also provides a means for the customer to indicate whether credits should be applied to this transaction. After the Cart is priced, the PricedCart response will provide information about which Credits are available (via the user’s Customer Profile). The assumption made is that Credits will be automatically applied during billing.

This style of API will support two ways of interaction between client application and the Reservation Service.

1. The client may call the Reservation API upon each primitive action (add Product, assign Store) and have the Cart validated at each step. This provides the client application with a means to provide the user with rapid feedback of any issues.
2. The client application may wait until the Cart is fully populated (with Products, a CreditCard and a Store) and then call upon the Reservation Service to validate and price the Cart.

*Note on additions/customization to Pipelines*

The Pipeline Service is used twice during the processing of a reservation. The first pipeline is used to validate and price a proposed cart. The second pipeline processes the reservation checkout. The Redbox.com site uses a set of pipeline components which were built to the requirements of the website. To fit the requirements of the Digital program, the OpenAPI Reservation Service will build a customized version of these pipelines. Most of the existing pipeline components will be re-used, but a few will be customized. The existing Redbox.com pipelines are:

**CartView Pipeline**

|  |  |  |
| --- | --- | --- |
| **Component Name** | **Description** | **Customized** |
| KioskProfileComponent | Given the StoreID assigned to the cart, retrieves other metadata about the store and adds it to the pipeline context. |  |
| ProductProfileComponent | Given the ProductIds assigned to the cart, retrieves other metadata about them and adds this to the pipeline context. |  |
| CustomerProfileComponent | Given the CustomerID, retrieves other metadata about the customer and adds it to the pipeline context. | Modified to fetch profile from the Customer Profile Service.  Retrieves Digital Credits from profile. |
| KioskValidationComponent | Validates that the kiosk is able to accept reservations |  |
| InventoryValidationComponent | Validates that the inventory is available |  |
| ProductValidationComponent | Validates that Product can be rented |  |
| PricingComponent | Applies pricing rules to calculate taxes, totals, etc. | Modified to support application of Credits to a Reservation. |

**Reservation Pipeline**

| **Component Name** | **Description** | **Customized** |
| --- | --- | --- |
| KioskProfileComponent | Given the StoreID assigned to the cart, retrieves other metadata about the store and adds it to the pipeline context. |  |
| ProductProfileComponent | Given the ProductIds assigned to the cart, retrieves other metadata about them and adds this to the pipeline context. |  |
| CustomerProfileComponent | Given the CustomerID, retrieves other metadata about the customer and adds it to the pipeline context. | Same as above |
| KioskValidationComponent | Validates that the kiosk is able to accept reservations |  |
| InventoryValidationComponent | Validates that the inventory is available |  |
| ProductValidationComponent | Validates that Product can be rented |  |
| PricingComponent | Applies pricing rules to calculate taxes, totals, etc. | Same as above |
| *CardDataValidationComponent* | Validates the credit card data. | *This component may not be needed for OpenAPI Reservations because we will be using only previously entered/validated credit cards.* |
| CardIntegrityComponent | Validates that this card is valid and that the user is allowed to rent. |  |
| *CardPersistenceComponent* | Saves new credit card data. | *This component may not be needed for OpenAPI Reservations because we will be using only previously entered/validated credit cards.* |
| *LoyaltyValidationComponent* |  | *This component may not be needed for OpenAPI Reservations because we may not be supporting the use of Web Credits.* |
| PendingTransactionPersistenceComponent | This component saves the Reservation to the Transaction table. |  |
| MerchantAuthorizationComponent | This component calls MerchantService to authorize the charge. |  |
| KioskReservationComponent | This component calls KioskBroker to call out to the kiosk to reserve the disc. |  |
| ReservationPersistenceComponent | This component saves the reservation to the Reservation table. |  |
| *LoyaltyPersistenceComponent* |  | *This component may not be needed for OpenAPI Reservations because we may not be supporting the use of Web Credits.* |
| InvoiceCloseoutComponent | This component closes out invoices when an error has occurred in any of the previous steps. |  |
| NotificationComponent | This component sends the customer an email with the Reservation detail. |  |

## Customer Account Service

The Customer Account Service will provide access to a customer’s profile and data related to a user, as depicted in Figure 2.6‑1: Customer Account Service. The major components of the Customer Account Service include:

* The Internal Source System will include:
  + Redbox Customer Profile Service
  + Redbox.com’s recommendation service
  + ODS (for Rental History)
  + Merchant Service
* The Customer Account Gateway (described further in 2.7.6)
* The CreditCardServices Gateway (described in 2.7.7)
* The Customer AccountService Endpoint in Apigee (see 2.7.15.1.5)

The following table summarizes the major features of the Customer Account Service and indicates which internal source system will house the data:

|  |  |  |
| --- | --- | --- |
| **Feature** | **Description** | **Source System** |
| Basic Profile information | name, address, email, display name | CustomerProfile Service |
| Genre and Format Preferences | List of genres and favorite format | CustomerProfile Preferences attributes |
| Favorite Stores (kiosks) | List of StoreIds | CustomerProfile Preferences attributes |
| Rental History | Past rental information | ODS (summarized from Rental) |
| Queues | Named queues, each of which contains an ordered list of ProductIds | CustomerProfile extended attributes |
| Recommended Products | ProductIds of recommended movies | Redbox.com |
| Accounts | Customer Accounts (credit card name, last 4 digits, exp date) | Customer Profile Account Service |
| Credits | Credit Balances, Credit Usage History | Credits |
| *AddCreditCard* | *Ability to add a new credit card* | *Merchant Service* |

In order to support these features, the following customizations/improvements to the internal Redbox systems will be required:

1. Customer rental history must be available in the ODS (updated ? often)
2. The logic that provides recommendations on Redbox.com website must be refactored into an application service so that OpenAPI can access it.
3. Data structures for genre and format preferences, favorite stores, and queues must be added to the internal Customer Profile service.



Figure ‑: Customer Account Service

The API structure of the Customer Account Service is described in the table below. ALL calls to these URLs will be passed through Apigee and forwarded to the appropriate gateway. Most will go to the Customer Account Gateway, while credits related calls will go to the Credits Gateway and the last call, AddCreditCard, will be processed by the CreditCard Services Gateway. The assumption made for this design is that PCI compliance rules will allow credit card details to be passed through Apigee. The gateway for credit cards has been split away from the main Customer Account Gateway to allow for deployment flexibility, such as if this unit of code needs to be segmented to a highly protected network segment. The AddCreditCard operation will accept an encrypted credit card number, which has been encrypted with the public key of Merchant Service (so that Merchant Service is the only party with the ability to decrypt the credit card).

| **API ID** | **Description** | **Input** | **Output** |
| --- | --- | --- | --- |
| CP0 | Status Monitor – called by the Redbox service monitoring infrastructure to confirm the operation of the service. | None | Status |
| CP1 | Retrieve the ‘basic profile’ for a customer | CustomerNumber | Basic Profile |
| CP2 | Saves the basic profile | CustomerNumber  BasicProfile | None |
| CP3 | Retrieve all preference data | CustomerNumber | Preferences |
| CP4 | Saves preferences | CustomerNumber  Preferences |  |
| CP5 | Retrieve rental history | CustomerNumber | Rental History info |
| CP6 | Retrieve the customer’s list of Queues | CustomerNumber | List of Queue Info |
| CP7 | Retrieve a single Queue | CustomerNumber  QueueId | Queue Info (ordered list of Product Ids) |
| CP8 | Replace a Queue | CustomerNumber  QueueId  Queue Info |  |
| CP9 | Delete a Queue | CustomerNumber  QueueId |  |
| CP10 | Add to or Move a Product in a Queue to the given position | CustomerNumber  QueueId  Position number  ProductId | Queue Info |
| CP11 | Remove an item from a Queue | CustomerNumber  QueueId  ProductId | Queue Info |
| CP12 | Retrieve recommendations | CustomerNumber  ProductType | List of Product Ids |
| CP13 | Retrieve the customer’s accounts | CustomerNumber | List of Account Info objects |
| CP14 | Retrieve a single account | CustomerNumber  AccountNumber | Account Info |
| CP15 | Updates attributes of an account | CustomerNumber  AccountNumber  AccountInfo |  |
| CP16 | Retrieve the Rental Credits balance for a customer | CustomerNumber | Credits Balances |
| CP17 | Retrieve the Rental Credits history for a customer | CustomerNumber | Credits History |
| CP18 | Add a CreditCard | CustomerNumber  CreditCard | Account Info |
| CP19 | Create a New Customer | Email Address | CustomerNumber |

*The design for C19 (Create Customer) is subject to change pending design completion of how authentication will be handled and confirmation of the need for and implementation of a Captcha guard on this operation.*

## Major Solution Components

New Components

The following components will be built for OpenAPI:

### Products Gateway

This will be a new web service, written in .NET as a WCF REST service. This service will be responsible for consuming the EPC Products list and transforming it into the canonical Products list for OpenAPI. The output of the Products Gateway will be the ‘Source MoviesList’ (see schema 6.1.2 in Appendix A). The output will identify each Product via an external GUID identifier. This service will strip out the internal ProductNumber before providing the output. The service will keep a record of the association between external GUID and internal ProductNumber, either in cache and/or in a database, as services added to OpenAPI in the future (such as a Reservation Service) will receive the external GUID as an input parameter and will need a mechanism to map from external back to internal identifier.

The Products Gateway will filter the set of EPC Movies that are provided to the Apigee Layer by using circulation status information provided by the ODS.

This service needs to be accessible from Apigee servers, which implies the need for a firewall rule to be implemented to allow this traffic.

Operations:

* /Products/Movies
  + Input: None
  + Output: ProductList, see schema 6.1.2 in Appendix A
  + Description: Respond to a request for a full list of Movies in Products Source Format.
* /Products/Movies/SimilarTitles
  + Input: productId
  + Output: RecommendedProductIDList, see schema 6.1.8 in Appendix A
  + Description: Calls the website recommendation service to obtain a list of Product IDs similar to the given ProductID. This operation will then transform the list from internal ProductIDs to external ProductID Guids.
* /FullRefresh
  + Input: None
  + Output: None
  + Description: Invoked by the **Products Cache Refresh Command Handler**, this method will call the **Products Endpoint** to cause a full cache refresh of both the Cloud Layer and the Open Services Layer.
* /Products/StatusMonitor
  + Input: None
  + Output: Status
  + Description: Confirm connectivity with EPC and report back status.

#### Products Gateway Caching

The Open Services Layer (WCF) will cache the Products metadata in source format. This will be used when transforming other entity types (such as Top20 and Inventory) that need to convert internal/source Product IDs to external Product IDs.

### Kiosk (“Stores”) Gateway

This will be a new web service, written in .NET as a WCF REST service. This service will provide both a bulk list of stores and a provide change notifications when the status of a store changes. The data source for the gateway will be the ODS and the service will transform this ODS data into the canonical Stores list for OpenAPI.

Changes to kiosk status will be routed to the Stores Gateway by the **Stores ChangedEventMessageHandler**. The gateway will update its cache and then notify the **Stores Endpoint** of the status change.

The Stores Gateway will be called by the Stores Endpoint when a full refresh of the Apigee cache is required. The gateway will output a ‘Source StoreList’ (see schema 6.1.9 in Appendix A). The output will identify each Store via an external GUID identifier. This service will strip out the internal Kiosk ID before providing the output. The service should keep a record of the association between external GUID and internal Kiosk ID, either in cache and/or in a database, as services added to OpenAPI in the future (such as a Reservation Service) will receive the external GUID and will need a mechanism to map from external back to internal identifier.

This service needs to be accessible from Apigee servers, which implies the need for a firewall rule to be implemented.

Operations:

* /Stores
  + Input: None
  + Output: StoreList in Stores Source Format; see schema 6.1.9 in Appendix A
  + Description: Respond to a request for a full list of Stores in **Stores Source Format**.
* /FullRefresh
  + Input: None
  + Output: None
  + Description: Invoked by the **Stores Cache Refresh Command Handler**, this method will call the **Stores Endpoint** to cause a full cache refresh.
* /StoreChanged *(Out of scope for R2)*
  + Input: StoreChangedEventMessage (see schema StoreChangedEventMessage Schema 6.1.41)
  + Output: None
  + Description: Invoked by the **StoresChangedEventHandler**, this operation will update the gateway’s cache and then call the **Stores Endpoint** to notify it of the status change.
* /Stores/StatusMonitor
  + Input: None
  + Output: Status
  + Description: Confirm connectivity with ODS and report back status.

### Top 20 Gateway

This will be a new web service, written in .NET as a WCF REST service. This service will be responsible for retrieving data from the ODS to retrieve the top 20 product ranking and transforming it into the canonical Top20 list for OpenAPI. The output of theTop20 Gateway will be the ‘Source Top20’ (see schema 6.1.15 in Appendix A). This output will list the items in the Top20 by identifier and provide information about their position in the list.

This service needs to be accessible from Apigee servers, which implies the need for a firewall rule to be implemented.

Operations:

* /Top20/Movies?p={period}
  + Input: Period – the 7 or 30 day window for the Top20 statistics
  + Output: Top20; see schema 6.1.15 in Appendix A
  + Description: Respond to a request for the Top20 Movies
* /FullRefresh
  + Input: None
  + Output: None
  + Description: Invoked by the **Top20 Cache Refresh Command Handler**, this method will call the **Top20 Endpoint** to cause a full cache.
* /Top20/StatusMonitor
  + Input: None
  + Output: Status
  + Description: Confirm connectivity with ODS and report back status.

### Inventory Gateway

This will be a new web service, written in .NET as a WCF REST service. This service will process inventory change notifications and pass them along to the Apigee layer. The data source for the gateway will be the **ODS** and the service will transform this **ODS** data into the canonical Inventory format for OpenAPI.

Changes to inventory status will be routed to the **Inventory Gateway** by the **Inventory Changed Event Message Handler**. The gateway will update its cache and then notify the Apigee **Inventory Endpoint** of the status change. Inventory change events will be bundled together before they are sent to the Inventory Endpoint. The design intent is to reduce/manage the traffic flow from the gateway to the Apigee layer for this frequently changing data. Events will be bundled into groups (the size of a group being configurable) before being fired off to Apigee. Event groups will be sent to Apigee when the group is filled or when a time-out is reached, whichever comes first. The time-out amount will also be configurable.

The **Inventory Gateway** will be called by the **Inventory** **Endpoint** when a full refresh of the Apigee cache is required. The gateway will output an Inventory List (see schema 6.1.12 in Appendix A). This output will include each Store and its inventory. This service will strip out the internal Kiosk ID before providing the output.

This service needs to be accessible from Apigee servers, which implies the need for a firewall rule to be implemented.

Operations:

* /Inventory
  + Input: None
  + Output: Inventory List; see schema 6.1.12 in Appendix A
  + Description: Called by the Apigee layer, this operation responds to a request for a full list of Inventory.
* /FullRefresh
  + Input: None
  + Output: None
  + Description: Invoked by the **Inventory Cache Refresh Command Handler**, this method will call the **Inventory Endpoint** to cause a full cache refresh.
* /InventoryChanged
  + Input: **InventoryChangedEventMessage** (see schema 6.1.42)
  + Output: None
  + Description: Invoked by the **InventoryChangedEventHandler**, this operation will forward the update to the Apigee layer **Inventory Endpoint**.
* /Inventory/StatusMonitor
  + Input: None
  + Output: Status
  + Description: Confirm connectivity with ODS and report back status.

### Reservations Gateway

This will be a new web service, written in .NET as a WCF REST service. This service will process reservation requests by interacting with the website’s Pipeline Service. This gateway will also interact with the Customer Profile service to retrieve credit card information.

Operations:

* /ExecuteReservationPricingPipeline
  + Input: Cart (see schema 6.1.18)
  + Output: PricedCart (see schema 6.1.20)
  + Description: Called by the Apigee layer, this operation calls the Pipeline service to execute the CartView pipeline (which validates and prices the cart).
* /Checkout
  + Input: PricedCart (see schema 6.1.20)
  + Output: ReservationResponse (see schema 6.1.22)
  + Description: Called by the Apigee layer, this operation calls the Pipeline service to execute the Reservation pipeline (which validates the cart and reserves the products).
* /StatusMonitor
  + Input: None
  + Output: Status
  + Description: Confirm connectivity with the Reservation Pipeline & Customer Profile Service and report back status.

### Customer Account Gateway

This will be a new web service, written in .NET as a WCF REST service. The Customer Account Service will provide access to a customer’s profile and data related to a user by interacting with several source systems and databases:

* Customer Profile Service (CP) – maintains the attributes about a customer
* ODS – provides a customer’s rental history
* Redbox.com – provides Movie recommendations
* Credits Service – maintains credit balance and history

The Apigee layer will pass through all calls to to this gateway. The API structure is the same for both the Endpoint and this service, and is described in Section 2.7.6.

The operations provided by this gateway include:

| **API ID** | **HTTP Method** | **URL structure**  Note: all URLs are prefixed with:/Customers/{CustomerNumber} | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| CP0 | GET | /StatusMonitor | None | Status |
| CP1 | GET | /Profile | CustomerNumber | Basic Profile |
| CP2 | PUT | /Profile | CustomerNumber  BasicProfile | None |
| CP3 | GET | /Preferences | CustomerNumber | Preferences |
| CP4 | PUT | /Preferences | CustomerNumber  Preferences |  |
| CP5 | GET | /RentalHistory | CustomerNumber | Rental History info |
| CP6 | GET | /Queues | CustomerNumber | List of Queue Info |
| CP7 | GET | /Queues/{queueId} | CustomerNumber  QueueId | Queue Info (ordered list of Product Ids) |
| CP8 | PUT | /Queues/{queueId} | CustomerNumber  QueueId  Queue Info |  |
| CP9 | DELETE | /Queues/{queueId} | CustomerNumber  QueueId |  |
| CP10 | PUT | /Queues/{queueId}/{position},{productId} | CustomerNumber  QueueId  Position number  ProductId | Queue Info |
| CP11 | DELETE | /Queues/{queueId}/{productId} | CustomerNumber  QueueId  ProductId | Queue Info |
| CP12 | GET | /RecommendedProducts?productType={type} | CustomerNumber  ProductType | Recommended Products Metadata |
| CP13 | GET | /Accounts | CustomerNumber | List of Account Info objects |
| CP14 | GET | /Accounts/{AccountNo} | CustomerNumber  AccountNumber | Account Info |
| CP15 | PUT | /Accounts/{AccountNo} | CustomerNumber  AccountNumber  AccountInfo |  |
| CP16 | GET | /Credits/Balance | CustomerNumber | Credits Balances |
| CP17 | GET | /Credits/History | CustomerNumber | Credits History |
| CP19 | POST | /CreateCustomer {does not use the /Customers/{CustomerNumber} prefix} | Email Address | Customer Number |

The data contracts used and provided by this service include:

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Contract** | **Output From** | **Input To** | **Schema Location** |
| Basic Profile | CP1 | CP2 | 6.1.23 |
| Preferences | CP3 | CP4 | 6.1.24 |
| RentalHistory | CP5 |  | 6.1.25 |
| QueuesListResponse | CP6 |  | 6.1.26 |
| QueueInfo | CP7, CP10, CP11 | CP8 | 6.1.27 |
| RecommendedProductsResponse | CP12 |  | 6.1.28 |
| AccountListResponse | CP13 |  | 6.1.29 |
| AccountInfo | CP14 | CP15 | 6.1.30 |
| CreditBalances | CP16 |  | 6.1.32 |
| CreditUseHistory | CP17 |  | 6.1.33 |

This service will be responsible for interacting with the source systems and translating the data to/from that required by the external format (eg, removing extra information, removing internal identifiers, etc). All interaction between consuming applications and this service will be stateless, meaning that this service will not maintain any session information about API interactions with consuming applications. There is no need to cache the data in the service.

The CreateCustomer operation, CP19, is slightly different than most other CustomerAccount operations. It does not use the conventional URL prefix for Customer operations, but instead is simply /CreateCustomer. The operation simply accepts an email address in the HTTP body of the POST request. If the creation of the customer within the source system (Customer Profile Service) is successful, then the output will contain the customer number as the HTTP body of the response. The HTTP Response Code will be set to 201 (Created) and a Location header will include the URL of the profile that has been created (ex: /Customers/{new Customer Number}/Profile.

### CreditCard Services Gateway

This will be a new web service, written in .NET as a WCF REST service. This service will be called by consuming applications to add a new credit card to an existing Customer Profile. The service will provide the following operation:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **API ID** | **HTTP Method** | **URL structure**  **Note: all URLs are prefixed with: /Customers/{CustomerNumber}** | **Input** | **Output** |
| CP18 | POST | /CreditCard | CustomerNumber  CreditCard | Account Info |

This operation will accept a description of a credit card that is to be added to a customer’s Profile and pass it to Merchant Services. Once the backend system has added the card, the gateway will coordinate with the CustomerProfile service to associate an Account with the Customer’s Profile and then return an Account info object that describes the card.

### Trailers Gateway

This will be a new web service, written in .NET as a WCF REST service. This service provides a gateway to the information about trailers that are housed at BrightCove. It will be called by the Apigee Layer to retrieve a bulk list of trailer metadata. The service will provide the following operations:

Operations:

* /FullRefresh
  + Input: None
  + Output: None
  + Description: Invoked by the **Products Cache Refresh Command Handler**, this method will call the **Products Endpoint** (operation P13 and include a forceRefresh=’true’ query string parameter) to cause a full cache refresh of the Trailers cache on Cloud (i.e., it will cause Apigee to call the next operation - /Products/Movies/Trailers).
* /Products/Movies/Trailers
  + Input: None
  + Output: MovieTrailers, see schema 6.1.46 in Appendix A
  + Description: Invoked by the Cloud Layer (Apigee) and returns a refreshed version of all Trailer metadata. This method calls BrightCove to fetch the Trailer metadata, which is then transformed into the external Trailers schema before passing back to the Cloud Layer.
* /Trailers/StatusMonitor
  + Input: None
  + Output: Status
  + Description: Confirm connectivity with BrightCove and report back status.

### Data Sources

#### **ODS**

The ODS will be enhanced to provide data on Stores (kiosks), Inventory, and Top20 entries. The ODS (Operational Data Store) is a database that is refreshed on a periodic basis with information from a variety of sources, primarily the RBDB. Consuming information from the ODS, instead of directly from RBDB, has the advantage of limiting the performance impact of OpenAPI on RBDB. The high-level design of ODS is depicted in Figure 2.7‑1: ODS Overview.



Figure ‑: ODS Overview

### Cache Refresh Batch Jobs

For R2 of the OpenAPI, there will be a need for these Cache Refresh Jobs:

#### **Products Cache Refresh Job**

Since Product metadata changes slowly, this job will be run on an hourly basis. It will simply add a **Products** **CacheRefreshCommand** **Message** (see schema in Appendix A, Section 6.1.36) onto the **Products** **CacheRefreshCommand** **Queue**.

#### **Stores Cache Refresh Job**

Since Kiosk metadata changes slowly, this job will be run on a daily basis. It will simply add a **Stores** **CacheRefreshCommand** **Message** (see schema in Appendix A, Section 6.1.37) onto the **Stores** **CacheRefreshCommand** **Queue**.

#### **Top20 Cache Refresh Job**

Top20 data is updated daily by a Data Services batch job. Therefore this batch job will execute daily after the completion of the existing job. It will simply add a **Top20** **CacheRefreshCommand** **Message** (see schema in Appendix A, Section 6.1.38) onto the **Top20** **CacheRefreshCommand** **Queue**.

#### **Inventory Cache Refresh Job**

When it becomes necessary to resynchronize the inventory information in the Inventory Gateway and the Apigee layer, this job will be invoked which will cause a full refresh of inventory data within OpenAPI. This job will add an **Inventory CacheRefreshCommand Message** (see schema 6.1.39 ) onto the **Inventory CacheRefreshCommand Queue**.

### Change Notifiers

#### **Store Change Notifier *(Out of Scope for R2)***

Store communication status changes frequently. This component will recognize communication status changes in the ODS data and submit a **Store Changed Event Message** (see schema 6.1.41 ) onto the **Store Changed Event Message Queue**. This will start the process of refreshing the Store gateway and the Apigee layer with the latest Store communication status data.

#### **Inventory Change Notifier**

This component will recognize inventory status changes in the ODS data and submit an **Inventory Changed Event Message** (see schema 6.1.42 ) onto the **Inventory Changed Event Message Queue**. This will kick-off the process of refreshing the Inventory gateway with the latest inventory information.

### Source System Change Service

The **Source System Change Service** will be a Windows NT Service. It will host the message handlers that process queue items.

### Cache Refresh Message Handlers

For R2 of OpenAPI there will be the need for the following message handlers:

#### **Products Cache Refresh Message Handler**

This message handler will consume messages from the **Products CacheRefreshCommand Queue**. It will invoke the **Products Gateway** to cause the cache of Products information to be refreshed.

#### **Stores Cache Refresh Message Handler**

This message handler will consume messages from the **Stores CacheRefreshCommand Queue**. It will invoke the **Stores Gateway** to cause the cache of Stores information to be refreshed.

#### **Top20 Cache Refresh Message Handler**

This message handler will consume messages from the **Top20 CacheRefreshCommand Queue**. It will invoke the **Top20 Gateway** to cause the cache of Top20 information to be refreshed.

#### **Inventory Cache Refresh Message Handler**

This message handler will consume messages from the **Inventory CacheRefreshCommand Queue**. It will invoke the **Inventory Gateway** to cause the cache of Inventory data to be refreshed.

### Item Changed Message Handlers

#### **Store ChangedEventMessage Handler *(Out of Scope for R2)***

This handler will consume messages from the **Stores Changed Event Message Queue**. It will forward the content of the message to the **Stores Gateway**.

#### **Inventory ChangedEventMessage Handler**

This handler will consume messages from the **Inventory Changed Event Message Queue**. It forwards the content of the message to the **Inventory Gateway**.

### Apigee Components

#### Endpoint Components

Apigee exposes each of the Service’s external Endpoints for each service.

Each endpoint will follow REST web service conventions for indicating the status of each service call. The HTTP Response Codes that are returned will include:

* 200 – OK : the operation succeeded
* 400 – Bad Request : the client provided information that failed validation or is otherwise in error
* 401 – Unauthorized : the service invocation was not properly authenticated or the caller does not have permission to perform that operation.
* 404 – Not Found : the operation identified an item that doesn’t exist (such as a ProductId or StoreId)
* 405 – Method Not Allowed : the client attempted to use an HTTP method not supported by this endpoint (ex: attempting to use a PUT operation on the Products Endpoint, since Products is a read-only service)
* 500 – Internal Server Error : returned when some internal processing error occurred that prevents the operation from completing

Additional HTTP Response Codes may be returned from service Endpoints under specific circumstances. These are noted in the operation descriptions below.

##### Products Endpoint

The Apigee Products Endpoint will provide the following operations:

| **ID** | **HTTP Method** | **URL structure** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| P0 | GET | /Products/StatusMonitor | None | Combined Status of Products, Top20 and Trailers gateways. |
| P1 | GET | /Products/Movies/Default | includeComingSoon flag | Products (Bulk list, younger than 250 days with ‘Display On Website’ set); may include ComingSoon movies if the ‘includeComingSoon’ flag is set to True |
| P2 | GET | /Products/Movies | None | Product Metadata for all products |
| P3 | GET | /Products/Movies/ComingSoon | None | Product Metadata for Coming Soon products |
| P4 | POST | /Products/Movies | ProductBrowseRequest (list of Product Ids) | Product Metadata |
| P5 | GET | /Products/Movies/{ProductID} | ProductId ID | Product Metadata |
| P6 | POST | /Products/Search&pageNum={pN}&pageSize={pS} | ProductSearchRequest:  Search Criteria  Sort Criteria  Page Number  Page Size | Paginated Product Search Response |
| P7 | GET | /Titles/Movies/Default | includeComingSoon flag | Title listing |
| P8 | GET | /Titles/Movies/ComingSoon | None | Title listing |
| P9 | GET | /Products/Movies/Similar?productId={pid} | Product ID | Products Metadata |
| P10 | GET | /Products/Movies/Top20?p={period} | Period: 7 or 30 | Top20 |
| P11 | GET | /Products/Movies?forceRefresh=’true’ | None | Product Metadata for all Products |
| P12 | GET | /Products/Movies/Top20? p={period}&forceRefresh=’true’ | Period: 7 or 30 | Refreshed Top20 |
| P13 | GET | /Products/Movies/Trailers/{ProductId} | Product ID | Trailer Metadata |
| P14 | Get | /Products/Movies/Trailers?forceRefresh=’true’ | None | Refreshed Trailers |

*All outputs may include an identifier for each movie specific to our Digital partner. This DigitalPartnerProductID will be included in the output only when the APIKey of the calling application belongs to the Digital Partner or Redbox; otherwise it will be omitted.*

Operations:

* P0: Get the Status of the Service
  + URL: /Products/StatusMonitor?apiKey={key}
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
  + Output: Status
  + Description: This operation will be called on a periodic basis by the Redbox operations monitoring service. This operation will call the Status Monitor operations of the Products, Top20, and Trailers gateways and report the combined result.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this ApiKey.
* P1: Get the Default Movie Browse List
  + URL: /Products/Movies/Default?apiKey={key}&includeComingSoon={true/false}
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
    - includeComingSoon: optional (default: false)
  + Output: ProductList, see schema 6.1.3 in Appendix A
  + Description: This represents a request for the default list of Movies (as displayed on the website) in Redbox External format by providing the contents of the **Products Default Catalog Cache**. The list will be filtered down to the Movies that are younger than 250 days old and have their ‘Display on Website’ flag set.
  + Security: The caller will provide an ApiKey which will be validated by Apigee.
  + Caching Control: The Apigee layer will set cache-control headers on the response to allow this response to be cached by downstream clients. The cache-control: max-age header will be set to a configurable value (eg: 60 minutes). The Apigee layer should respond with an HTTP status code of 304 (Not Modified) if the client submits a conditional GET or HEAD request with an If-Unmodified-Since header that equals or is later than the most recent update timestamp in the **Products Default Catalog Cache**.
* P2: Get the Full Movie Catalog
  + URL: /Products/Movies?apiKey={key}
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
  + Output: ProductList, see schema 6.1.3 in Appendix A
  + Description: This represents a request for the full list of Movies in the catalog in Redbox External format by providing the contents of the **Products In-Circulation Cache**.
  + Security: The caller will provide an ApiKey which will be validated by Apigee.
  + Caching Control: The Apigee layer will set cache-control headers on the response to allow this response to be cached by downstream clients. The cache-control: max-age header will be set to a configurable value (eg: 60 minutes). The Apigee layer should respond with an HTTP status code of 304 (Not Modified) if the client submits a conditional GET or HEAD request with an If-Unmodified-Since header that equals or is later than the most recent update timestamp in the **Products In-Circulation Cache**.
* P3: Get the Coming Soon Movie List
  + URL: /Products/Movies/ComingSoon?apiKey={key}
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
  + Output: ProductList, see schema 6.1.3 in Appendix A
  + Description: This represents a request for the ComingSoon list of Movies (as displayed on the website) in Redbox External format by providing the contents of the **Products Default Catalog Cache**. The list will be filtered down to the Movies that are not yet released, but are have a scheduled street date of less than 30 days in the future.
  + Security: The caller will provide an ApiKey which will be validated by Apigee.
  + Caching Control: The Apigee layer will set cache-control headers on the response to allow this response to be cached by downstream clients. The cache-control: max-age header will be set to a configurable value (eg: 60 minutes). The Apigee layer should respond with an HTTP status code of 304 (Not Modified) if the client submits a conditional GET or HEAD request with an If-Unmodified-Since header that equals or is later than the most recent update timestamp in the **Products Default Catalog Cache**.
* P4: Get metadata about a set of Movies by ID
  + URL: /Products/Movies?apiKey={key}&pageNum={pN}&pageSize={pS}
  + Method: POST
  + Input:
    - ApiKey: mandatory; the security key
    - ProductBrowseRequest (see schema in Section 6.1.4)
    - Paging Input (PageNum and PageSize)
  + Output: PaginatedProductBrowseResponse, see schema 6.1.6
  + Description: This is a request to retrieve specific products from the Movie catalog. The Products Service will fulfill the request by fetching the contents from the **Products In-Circulation Cache**.
  + Security: The caller will provide an ApiKey which will be validated by Apigee.
  + Caching Control: The Apigee layer will set cache-control headers on the response to allow this response to be cached by the consuming client, ie, Private. The cache-control: max-age header will be set to a configurable value (eg: 60 minutes).
* P5: Get metadata of a single Movie
  + URL: /Products/Movies/{ProductId}?apiKey={key}
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
    - ProductID: mandatory; the id of a single product
  + Output: ProductList, see schema 6.1.3 in Appendix A
  + Description: This is a request to retrieve a single product from the Movie catalog. The Products Service will fulfill the request by fetching the contents from the **Products In-Circulation Cache**.
  + Security: The caller will provide an ApiKey which will be validated by Apigee.
  + Caching Control: The Apigee layer will set cache-control headers on the response to allow this response to be cached by downstream clients. The cache-control: max-age header will be set to a configurable value (eg: 60 minutes). The Apigee layer should respond with an HTTP status code of 304 (Not Modified) if the client submits a conditional GET or HEAD request with an If-Unmodified-Since header that equals or is later than the most recent update timestamp for this item in the **Products In-Circulation Cache**.
* P6: Search for Movies
  + URL: /Products/Search?apiKey={key}&pageNum={pN}&pageSize={pS}
  + Method: POST
  + Input:
    - ApiKey: mandatory; the security key
    - ProductSearchRequest; see schema 6.1.5
    - Paging Input (PageNum and PageSize)
  + Output: PaginatedProductBrowseResponse, see schema 6.1.6
  + Description: This operation will search through the cached Product metadata (in the **Products** **In-Circulation Cache** to find the products that match the search criteria and sort the result set according to the given sorting criteria. The result set can optionally be paginated.
  + Security: The caller will provide an ApiKey which will be validated by Apigee.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private. The cache-control: max-age header will be set to a configurable value (eg: 60 minutes).
* P7: Get the Titles Feed
  + URL: /Titles/Movies/Default?apiKey={key}&includeComingSoon={true/false}
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
    - includeComingSoon: optional (default: false)
  + Output: Titles; see schema 6.1.7 in Appendix A
  + Description: Retrieve the list of Products collapsed on BaselineFilmID.
  + Security: The caller will provide an ApiKey which will be validated by Apigee.
  + Caching Control: The Apigee layer will set cache-control headers on the response to allow this response to be cached by downstream clients. The cache-control: max-age header will be set to a configurable value (eg: 60 minutes). The Apigee layer should respond with an HTTP status code of 304 (Not Modified) if the client submits a conditional GET or HEAD request with an If-Unmodified-Since header that equals or is later than the most recent update timestamp in the **Titles Default Cache**.
* P8: Get the Titles Coming Soon list
  + URL: /Titles/Movies/ComingSoon?apiKey={key}
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
  + Output: Titles; see schema 6.1.7 in Appendix A
  + Description: Retrieve the list of ComingSoon Products collapsed on BaselineFilmID.
  + Security: The caller will provide an ApiKey which will be validated by Apigee.
  + Caching Control: The Apigee layer will set cache-control headers on the response to allow this response to be cached by downstream clients. The cache-control: max-age header will be set to a configurable value (eg: 60 minutes). The Apigee layer should respond with an HTTP status code of 304 (Not Modified) if the client submits a conditional GET or HEAD request with an If-Unmodified-Since header that equals or is later than the most recent update timestamp in the **Titles Default Cache**.
* P9: Recommend Similar Titles
  + URL: /Products/Movies/SimilarTitles?apiKey={key}&productId={pid}
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
    - ProductId: mandatory; the product ID upon which to base recommendations
  + Output: ProductList, see schema 6.1.3 in Appendix A
  + Description: This operation will provide a list of movies similar to the one identified by the productId query string. This operation will call the Products Gateway’s SimilarTitles operation, which will reply back with a set of ProductIds. The Apigee layer will then lookup those products from the **Products In-Circulation Cache** to build the output of the operation.
  + Security: The caller will provide an ApiKey which will be validated by Apigee.
  + Caching Control: The Apigee layer will set cache-control headers on the response to allow this response to be cached by downstream clients. The cache-control: max-age header will be set to a configurable value (eg: 60 minutes). The Apigee layer will set cache-control headers on the response to Private.
* P10: Get the Top20 list
  + URL: /Products/Movies/Top20?p={period}&apiKey={key}
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
    - Period – the 7 or 30 day window for the Top20 statistics
  + Output: Top20; see schema 6.1.17 in Appendix A
  + Description: Respond to a request for the Top20 Movies by providing the contents of the **Top20 External Format Cache**.
  + Security: The caller will provide an ApiKey which will be validated by Apigee.
  + Caching Control: The Apigee layer will set cache-control headers on the response to allow this response to be cached by downstream clients. The cache-control: max-age header will be set to a configurable value (eg: 60 minutes). The Apigee layer should respond with an HTTP status code of 304 (Not Modified) if the client submits a conditional GET or HEAD request with an If-Unmodified-Since header that equals or is later than the most recent update timestamp in the **Top20 External Format Cache**.
* P11: Refresh the Products Cache
  + URL: /Products/Movies?apiKey={key}&forceRefresh=’true’
  + Input :
    - ApiKey: mandatory; the security key
  + Output: The contents of the newly refreshed cache
  + Description: Refresh cache content by calling the Products Source System gateway and then transform the result into the Products External Format and replace the cache.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this key.
* P12: Refresh the Top20 Cache
  + URL: /Products/Movies/Top20/?apiKey={key}&p={period}&forceRefresh=’true’
  + Input :
    - ApiKey: mandatory; the security key
    - Period – the 7 or 30 day window for the Top20 statistics
  + Output: The contents of the newly refreshed cache
  + Description: Refresh cache content by calling the Top20 Source System gateway and then transform the result into the Top20 External Format and replace the cache. The data from the Top20 Source System Gateway will be merged with product metadata to create the full Top20 list. The product metadata will be retrieved from the Products External Format cache held in the Apigee layer.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this key.
* P13: Get Trailer metadata of a single Movie
  + URL: /Products/Movies/Trailers/{ProductId}?apiKey={key}
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
    - ProductID: mandatory; the id of a single product
  + Output: MovieTrailers, see schema 6.1.46 in Appendix A
  + Description: This is a request to retrieve trailer metadata for a single product from the Movie catalog. This operation will fulfill the request by providing data from the Trailers Cache.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. ALSO, the Apigee layer will validate that the ApiKey has been given permission to view Trailer information.
  + Caching Control: The Apigee layer will set cache-control headers on the response to allow this response to be cached by downstream clients. The cache-control: max-age header will be set to a configurable value (eg: 60 minutes).
* P14: Refresh Trailer metadata for all movies
  + URL: /Products/Movies/Trailers?apiKey={key}&forceRefresh=’true’
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
  + Output: MovieTrailers, see schema 6.1.46 in Appendix A
  + Description: This request will be issued by the Trailers Gateway to tell Apigee to refresh its cache of trailer metadata by calling back to the Trailers Gateway to get a full list of trailer data.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. The only ApiKey allowed to call this operation will be the Redbox key.

##### Stores Endpoint

The Apigee Stores Endpoint provides the following operations:

| **ID** | **HTTP Method** | **URL structure** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| S0 | GET | /Stores/StatusMonitor | None | Status |
| S1 | GET | /Stores | None | Stores |
| S2 | GET | /Stores? storeList={storeIDs}&banner={b}&pageNum={pN}&pageSize={pS} | List of StoreIDs to search for;  Banner name to filter;  Page Number and Page Size | StoreLookupList |
| S3 | GET | /Stores/LatLong/{lat},{longt}?radius={r}&count={c}&retailer={ret}&sortBy={s}&sortDir={asc/desc}&pageNum={pN}&pageSize={pS} | Lat/Long of stores to search for;  Radius to search;  Count of number of Stores to return;  Retailer name to filter;  SortBy (Distance or Retailer);  SortDirection;  Page Number and Page Size | StoresLookupList |
| S4 | GET | /Stores?forceRefresh=’true’ | None | Stores |
| S5 | POST | /Stores/StoreChanged | StoreChanges |  |

Operations:

All operations will use HTTP GET method.

* S0: Get the Status of the Service
  + URL: /Stores/StatusMonitor?apiKey={key}
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
  + Output: Status
  + Description: This operation will be called on a periodic basis by the Redbox operations monitoring service. This operation will call the Status Monitor operations of the Stores gateway and report the result.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this ApiKey.
* S1: Get All Stores
  + URL: /Stores?apiKey={key}
  + Input: None
  + Output: StoreList; see schema 6.1.10 in Appendix A
  + Description: Respond to a request for a full list of Stores by providing the contents of the **Stores External Format Cache**. This operation is intended for use by affiliates that will consume store data on a daily (or other infrequent basis). The output will NOT include inventory data, nor will it include the Communication Status property.
  + Security: The caller will provide an ApiKey which will be validated by Apigee.
  + Caching Control: The Apigee layer will set cache-control headers on the response to allow this response to be cached by downstream clients. The cache-control: max-age header will be set to a configurable value (eg: 5 minutes). The Apigee layer should respond with an HTTP status code of 304 (Not Modified) if the client submits a conditional GET or HEAD request with an If-Unmodified-Since header that equals or is later than the most recent update timestamp of the full Store list in the **Stores External Format Cache**.
* S2: Get Store Metadata by StoreID(s)
  + URL: /Stores/?apiKey={key}&storeList={storeIDs}&banner={b}&pageNum={pN}&pageSize={pS}
  + Input:
    - StoreList: mandatory; a comma-separated of StoreIDs indicating which stores for which information should be retrieved. Maximum of 50 stores allowed.
    - Banner: optional; filter by the name of a retail partner (such as ‘Walgreens’)
    - PageNumber: optional; when specified the results will be paged and will return a page-worth of results. When not included, the operation will return all stores matching the other criteria.
    - PageSize: optional; when specified along with the PageNumber, sets the number of stores to return per-page. Default: 10
  + Output: StoreLookupList; see schema 6.1.13 in Appendix A
  + Description: Respond to a request for a list of Stores that are located near a given location.
  + Security: The caller will provide an ApiKey which will be validated by Apigee.
  + Caching Control: The Apigee layer will set cache-control headers on the response to allow this response to be cached by downstream clients. The cache-control: max-age header will be set to a configurable value (eg: 5 minutes). The Apigee layer should respond with an HTTP status code of 304 (Not Modified) if the client submits a conditional GET or HEAD request with an If-Unmodified-Since header that equals or is later than that of the most recently updated Store from the StoreList in the **Stores External Format Cache**.
* S3: Find Stores by Lat/Long
  + URL: /Stores/LatLong/{lat},{longt}?apiKey={key}&radius={r}&count={c}&retailer={ret}&sortBy={sB}&sortDir={asc/desc}&pageNum={pN}&pageSize={pS}
  + Input:
    - Lat, Long: location expressed as latitude, longitude coordinates
    - Radius: optional; maximum distance in miles from the location, default: 10
    - Count: optional; the maximum number of stores to return, default and Max: 50
    - Retailer: optional; filter by the name of a retail partner (such as ‘Walgreens’)
    - SortBy: optional; sort by either Distance from location or by Retailer name. Default: Distance
    - SortDirection: optional; Ascending or Descending. Default: Desc.
    - PageNumber: optional; when specified the results will be paged and will return a page-worth of results. When not included, the operation will return all stores matching the other criteria.
    - PageSize: optional; when specified along with the PageNumber, sets the number of stores to return per-page. Default: 10
  + Output: StoreLookupList; see schema 6.1.13 in Appendix A. This result will include the distance of each store from the given Lat/Long location.
  + Description: Respond to a request for a list of Stores that are located near a given location.
  + Security: The caller will provide an ApiKey which will be validated by Apigee.
  + Caching Control: The Apigee layer will set cache-control headers on the response to allow this response to be cached by downstream clients. The cache-control: max-age header will be set to a configurable value (eg: 5 minutes). The Apigee layer should respond with an HTTP status code of 304 (Not Modified) if the client submits a conditional GET or HEAD request with an If-Unmodified-Since header that equals or is later than that of the most recently updated Store from the set of Stores identified by the query in the **Stores External Format Cache**.
* S4: Refresh the Stores Cache
  + URL: /Stores?apiKey={key}&forceRefresh=’true’
  + Input : None
  + Output: The contents of the newly refreshed cache
  + Description: Refresh cache content by calling the Stores Source System gateway and then transform the result into the Stores External Format and replace the cache.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this key.
* S5: Update Stores Cache Item
  + URL: /Stores/StoreChanged?apiKey={key} (HTTP POST)
  + Input :
    - StoreChangesMessage (see 6.1.44)
  + Output: The contents of the newly refreshed cache
  + Description: Update cache content with the data provided in the StoreChangesMessage and then transform the result into the Stores External Format and replace the affected cache items.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this key.

##### Inventory Endpoint

The Inventory Endpoint will expose the following operations:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **HTTP Method** | **URL structure** | **Input** | **Output** |
| I0 | GET | /Inventory/StatusMonitor | None | Status |
| I1 | GET | /Inventory?storeList={storeIDs}&inventoryProducts={listOfProductIDs}&pageNum={pN}&pageSize={pS} | A StoreList of the list of Store Ids for which Inventory is to be retrieved;  List of Product IDs for which Inventory status is to be retrieved;  Page Number and Page Size | Inventory Lookup |
| I2 | GET | /StoresInventoryForProduct/{productIdType}/{productId}/{lat,long}?storeCount={c}&radius={r}&retailer={ret}&sortBy={s}&sortDir={asc/desc}&pageNum={pN}&pageSize={pS} | Product ID Type;  Product ID;  Location (Lat/Long);  Store Count (max number of stores to return);  Radius (search radius in miles);  Retailer to filter by;  SortBy (Distance or Retailer);  SortDirection;  Page Number and Page Size | StoresInventory Lookup |
| I3 | POST | /Inventory/InventoryChanged?apiKey={key} | InventoryChanges | None |
| I4 | GET | /Inventory?forceRefresh=’true’ | None | None |

*Inventory call I2 is designed to support multiple types of product identity. For R2 of OpenAPI, this will be limited to ProductID (for Redbox Product GUIDs) and DigitalID for use by the Digital Partner.*

Operations:

* I0: Get the Status of the Service
  + URL: /Inventory/StatusMonitor?apiKey={key}
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
  + Output: Status
  + Description: This operation will be called on a periodic basis by the Redbox operations monitoring service. This operation will call the Status Monitor operations of the Inventory gateway and report the result.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this ApiKey.
* I1: Get Inventory For Store(s)
  + URL: /Inventory/?apiKey={key}&storeList={storeIDs}&inventoryProducts={listOfProductIDs}&pageNum={pN}&pageSize={pS}
  + Input:
    - StoreList: mandatory; a comma-separated list of StoreIDs indicating which stores for which information should be retrieved. Maximum of 50 stores allowed.
    - InventoryProducts: optional; a comma separated list of Redbox ProductID GUIDs. This parameter will filter the inventory returned to just the ProductIDs provided.
    - PageNumber: optional; when specified the results will be paged and will return a page-worth of results. When not included, the operation will return all stores matching the other criteria.
    - PageSize: optional; when specified along with the PageNumber, sets the number of stores to return per-page. Default: 10
  + Output: InventoryLookup; see schema 6.1.14 in Appendix A
  + Description: Provides the inventory for a set of stores, optionally filtered to a set of products. If no ProductIDs are provided, this operation returns all the Inventory records for the identified Store(s).
  + Security: The caller will provide an ApiKey which will be validated by Apigee.
  + Caching Control: The Apigee layer will set cache-control headers on the response to allow this response to be cached by downstream clients. The cache-control: max-age header will be set to a configurable value (eg: 5 minutes). The Apigee layer should respond with an HTTP status code of 304 (Not Modified) if the client submits a conditional GET or HEAD request with an If-Unmodified-Since header that equals or is later than that of the most recently updated InventoryLookup from the set of Stores identified by the query in the **Inventory External Format Cache**.
* I2: Return Store Metadata & Inventory By ProductID
  + URL: /StoresInventory/{productIdType}/{productId}/{lat}, {long}?apiKey={key}&radius={r}&count={c}&retailer={ret}&sortBy={sB}&sortDir={asc/desc}&pageNum={pN}&pageSize={pS}
  + Input:
    - ProductIdType: mandatory; identifies the type of Product ID (Redbox, UPC, Baseline, or DigitalPartner)
    - ProductID: mandatory; the Product ID of a movie.
    - Lat, Long: mandatory; the location around which to search for stores and inventory.
    - Radius: optional; maximum distance from the location to search (default: 10 miles).
    - Count: optional; maximum number of kiosks to return (default: 50).
    - Retailer: optional; name of a retail location to filter on.
    - SortBy: optional; sort by either Distance from location or by Retailer name. Default: Distance
    - SortDirection: optional; Ascending or Descending. Default: Desc.
    - PageNumber: optional; when specified the results will be paged and will return a page-worth of results. When not included, the operation will return all stores matching the other criteria.
    - PageSize: optional; when specified along with the PageNumber, sets the number of stores to return per-page. Default: 10
  + Output: StoresInventoryLookup; see schema 6.1.16 in Appendix A
  + Description: Provides the store metadata and inventory for a single product as identified by the Product ID. When called by the Digital Partner, the Digital Partner will provide a Digital Partner ID. Other callers may use another form of Product ID.
  + Security: The caller will provide an ApiKey which will be validated by Apigee.
  + Caching Control: The Apigee layer will set cache-control headers on the response to allow this response to be cached by downstream clients. The cache-control: max-age header will be set to a configurable value (eg: 5 minutes). The Apigee layer should respond with an HTTP status code of 304 (Not Modified) if the client submits a conditional GET or HEAD request with an If-Unmodified-Since header that equals or is later than that of the most recently updated StoresInventory from the set of Stores identified by the query in the **Inventory External Format Cache**.
* I3: Update the Inventory cache for a group of changed Inventory items
  + URL: /Inventory/InventoryChanged?apiKey={key} (HTTP POST)
  + Input :
    - InventoryChangesMessage (see schema 6.1.45)
  + Output: None
  + Description: Update Inventory cache content with the data provided in the InventoryChangesMessage. This may include updates for a single or multiple inventory items.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this key.
* I4: Force Update to entire Inventory Cache
  + URL: /Inventory?apiKey={key}&forceRefresh=’true’
  + Input : None
  + Output: None
  + Description: Refresh cache content by calling the Inventory gateway and replace the cache.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this key.

##### Reservation Endpoint

The Reservation Endpoint will process cart manipulation events from the client application and process requests to ‘Check-out’ the cart.

The operations exposed by Reservations are:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **HTTP Method** | **URL structure** | **Input** | **Returns** |
| R0 | GET | /Reservation/StatusMonitor | None | Status |
| R1 | PUT | /Cart/{UserID}/{ProductID} | User ID of customer  Product ID (GUID)  Cart | CartValidation |
| R2 | DELETE | /Cart/{UserID}/{ProductID} | User ID of customer  Product ID (GUID) | CartValidation |
| R3 | POST | /Cart/{UserID}/AssignStore/{StoreID} | User ID  Store ID  Cart | CartValidation |
| R4 | POST | /Cart/{UserID}/Price | User ID  Cart | PricedCartResponse |
| R5 | POST | /Reservation/Checkout | PricedCart | ReservationResponse |

**See Section 3.13.1 on Security for more information on how these sensitive operations will be secured.**

Operation Detail:

* R0: Get the Status of the Service
  + URL: /Reservation/StatusMonitor?apiKey={key}
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
  + Output: Status
  + Description: This operation will be called on a periodic basis by the Redbox operations monitoring service. This operation will call the Status Monitor operations of the Reservation gateway and report the result.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this ApiKey.
* R1: Add a Product to a Cart
  + URL: /Cart/{UserID}/{ProductID}?apiKey={key}
  + Method: PUT
  + Input :
    - UserID: GUID that identifies the customer
    - ProductID (GUID) that identifies the product to be added to the cart
    - Cart (see schema: 6.1.18)
  + Output: CartValidation (see schema: 6.1.19)
  + Description: Apigee layer will validate that the Product is InStock at the given Store by confirming the Product’s inventory status in the Inventory cache at the Apigee layer. It will also validate that the number of items in the cart passes preliminary business rule checks. If validation succeeds, the product will be added to the cart.
  + Security: The caller will provide an ApiKey which will be validated by Apigee.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* R2: Delete a Product from a Cart
  + URL: /Cart/{UserID}/{ProductID}?apiKey={key}
  + Method: DELETE
  + Input :
    - UserID: GUID that identifies the customer
    - ProductID (GUID) that identifies the product to be removed from the cart
    - Cart (see schema: 6.1.18)
  + Output: CartValidation (see schema: 6.1.19)
  + Description: Apigee layer will validate the removal of the product from the cart and return a CartValidation object.
  + Security: The caller will provide an ApiKey which will be validated by Apigee.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* R3: Assign a Store to the Cart
  + URL: /Cart/{UserID}/AssignStore/{StoreID}?apiKey={key}
  + Method: POST
  + Input :
    - UserID: GUID that identifies the customer
    - StoreID (GUID) that identifies the store to be assigned to the cart
    - Cart (see schema: 6.1.18)
  + Output: CartValidation (see schema: 6.1.19)
  + Description: Apigee layer will validate that the Store is Online (based upon data in the Store Cache). It will then validate that all products in the Cart are available at that Store.
  + Security: The caller will provide an ApiKey which will be validated by Apigee.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* R4: Price the Cart
  + URL: /Cart/{UserID}/Price?apiKey={key}
  + Method: POST
  + Input :
    - UserID: GUID that identifies the customer
    - Cart (see schema: 6.1.18 )
  + Output: PricedCartResponse (see schema: 6.1.21 )
  + Description: Apigee layer will call the Reservation Gateway to have the Cart validated and priced. The output of the Reservation Gateway will be transformed into the PricedCart schema(see schema: 6.1.20)
  + Security: The caller will provide an ApiKey which will be validated by Apigee. This operation must include the customer’s security token in the HTTP Authorization Header as described in Section 2.12.1.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* R5: Check-out the Cart
  + URL: /Reservation/Checkout?apiKey={key}
  + Method: POST
  + Input :
    - PricedCart (see schema: 6.1.20)
  + Output: ReservationResponse (see schema: 6.1.22)
  + Description: Apigee layer will call the Reservation Gateway to process the Checkout request.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. This operation must include the customer’s security token in the HTTP Authorization Header as described in Section 2.12.1.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.

##### CustomerAccount Endpoint

The CustomerAccount endpoint will provide operations that support viewing and manipulation of information about the customer. The endpoint will support the following operations:

| **API ID** | **HTTP Method** | **URL structure**  Note: all URLs are prefixed with:/Customers/{CustomerNumber} | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| CP0 | GET | /StatusMonitor | None | Status |
| CP1 | GET | /Profile | CustomerNumber | Basic Profile |
| CP2 | PUT | /Profile | CustomerNumber  BasicProfile | None |
| CP3 | GET | /Preferences | CustomerNumber | Preferences |
| CP4 | PUT | /Preferences | CustomerNumber  Preferences |  |
| CP5 | GET | /RentalHistory | CustomerNumber | Rental History info |
| CP6 | GET | /Queues | CustomerNumber | List of Queue Info |
| CP7 | GET | /Queues/{queueId} | CustomerNumber  QueueId | Queue Info (ordered list of Product Ids) |
| CP8 | PUT | /Queues/{queueId} | CustomerNumber  QueueId  Queue Info |  |
| CP9 | DELETE | /Queues/{queueId} | CustomerNumber  QueueId |  |
| CP10 | PUT | /Queues/{queueId}/{position},{productId} | CustomerNumber  QueueId  Position number  ProductId | Queue Info |
| CP11 | DELETE | /Queues/{queueId}/{productId} | CustomerNumber  QueueId  ProductId | Queue Info |
| CP12 | GET | /RecommendedProducts?productType={type} | CustomerNumber  ProductType | List of Product Ids |
| CP13 | GET | /Accounts | CustomerNumber | List of Account Info objects |
| CP14 | GET | /Accounts/{AccountNo} | CustomerNumber  AccountNumber | Account Info |
| CP15 | PUT | /Accounts/{AccountNo} | CustomerNumber  AccountNumber  AccountInfo |  |
| CP16 | GET | /Credits/Balance | CustomerNumber | Credits Balances |
| CP17 | GET | /Credits/History | CustomerNumber | Credits History |
| CP18 | POST | /CreditCard | CustomerNumber  CreditCard | Account Info |
| CP19 | POST | /CreateCustomer | Email Address | CustomerNumber |

All of these operations will be passed directly through to the CustomerAccount Gateway, with the exception of CP18 which will be passed through to the CreditCard Gateway.

Operation Detail:

* CP0: Get the Status of the Service
  + URL: /StatusMonitor?apiKey={key}
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
  + Output: Status
  + Description: This operation will be called on a periodic basis by the Redbox operations monitoring service. This operation will call the Status Monitor operations of the CustomerAccount gateway and report the result.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this ApiKey.
* CP1: Get Profile
  + URL: /Customers/{CustomerNumber}/Profile?apiKey={key}
  + Method: GET
  + Input :
    - CustomerNumber: ID that identifies the customer
  + Output: BasicProfile (see schema: 6.1.23)
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an ApiKey which will be validated by Apigee. This operation must include the customer’s security token in the HTTP Authorization Header as described in Section 2.12.1.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP2: Save Profile
  + URL: /Customers/{CustomerNumber}/Profile?apiKey={key}
  + Method: PUT
  + Input :
    - CustomerNumber: ID that identifies the customer
    - BasicProfile (see schema: 6.1.23)
  + Output: None
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an ApiKey which will be validated by Apigee. This operation must include the customer’s security token in the HTTP Authorization Header as described in Section 2.12.1.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP3: Get Preferences
  + URL: /Customers/{CustomerNumber}/Preferences?apiKey={key}
  + Method: GET
  + Input :
    - CustomerNumber: ID that identifies the customer
  + Output: Preferences (see schema: 6.1.24)
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an ApiKey which will be validated by Apigee. This operation must include the customer’s security token in the HTTP Authorization Header as described in Section 2.12.1.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP4: Save Preferences
  + URL: /Customers/{CustomerNumber}/Preferences?apiKey={key}
  + Method: PUT
  + Input :
    - CustomerNumber: ID that identifies the customer
    - Preferences (see schema: 6.1.24)
  + Output: None
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an ApiKey which will be validated by Apigee. This operation must include the customer’s security token in the HTTP Authorization Header as described in Section 2.12.1.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP5: Get Rental History
  + URL: /Customers/{CustomerNumber}/RentalHistory?apiKey={key}
  + Method: GET
  + Input :
    - CustomerNumber: ID that identifies the customer
  + Output: RentalHistory (see schema: 6.1.25)
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an ApiKey which will be validated by Apigee. This operation must include the customer’s security token in the HTTP Authorization Header as described in Section 2.12.1.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP6: Get Queues
  + URL: /Customers/{CustomerNumber}/Queues?apiKey={key}
  + Method: GET
  + Input :
    - CustomerNumber: ID that identifies the customer
  + Output: List of Queue Info (see schema: 6.1.26)
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an ApiKey which will be validated by Apigee. This operation must include the customer’s security token in the HTTP Authorization Header as described in Section 2.12.1.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP7: Get a Queue
  + URL: /Customers/{CustomerNumber}/Queues/{queueId}?apiKey={key}
  + Method: GET
  + Input :
    - CustomerNumber: ID that identifies the customer
    - QueueId: ID that identifies a single Queue
  + Output: Queue Info (see schema: 6.1.27)
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an ApiKey which will be validated by Apigee. This operation must include the customer’s security token in the HTTP Authorization Header as described in Section 2.12.1.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP8: Update Queue
  + URL: /Customers/{CustomerNumber}/Queues/{queueId}?apiKey={key}
  + Method: PUT
  + Input :
    - CustomerNumber: ID that identifies the customer
    - QueueId
    - QueueInfo
  + Output: None
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an ApiKey which will be validated by Apigee. This operation must include the customer’s security token in the HTTP Authorization Header as described in Section 2.12.1.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP9: Delete a Queue
  + URL: /Customers/{CustomerNumber}/Queues/{queueId}?apiKey={key}
  + Method: DELETE
  + Input :
    - CustomerNumber: ID that identifies the customer
    - QueueId
  + Output: None
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an ApiKey which will be validated by Apigee. This operation must include the customer’s security token in the HTTP Authorization Header as described in Section 2.12.1.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP10: Add/Update a Product to a Queue
  + URL: /Customers/{CustomerNumber}/Queues/{queueId}/{position}, {productId}?apiKey={key}
  + Method: PUT
  + Input :
    - CustomerNumber: ID that identifies the customer
    - QueueId: identifier of a queue
    - Position: index position within the queue to add the product or to move the product
    - ProductId: identifier (GUID) of the Product
  + Output: QueueInfo : updated queue
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an ApiKey which will be validated by Apigee. This operation must include the customer’s security token in the HTTP Authorization Header as described in Section 2.12.1.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP11: Remove a Product from a Queue
  + URL: /Customers/{CustomerNumber}/Queues/{queueId}/{productId}?apiKey={key}
  + Method: DELETE
  + Input :
    - CustomerNumber: ID that identifies the customer
    - QueueId: identifier of the queue
    - ProductId: identifier of the Product
  + Output: QueueInfo : updated queue
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an ApiKey which will be validated by Apigee. This operation must include the customer’s security token in the HTTP Authorization Header as described in Section 2.12.1.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP12: Get Recommendations
  + URL: /Customers/{CustomerNumber}/RecommendedProducts?apiKey={key}&productType={pT}
  + Method: GET
  + Input :
    - CustomerNumber: ID that identifies the customer
    - pT: Product Type identifier (Movie, Game)
  + Output: Recommendations (list of Product IDs) (see schema: 6.1.28)
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an ApiKey which will be validated by Apigee. This operation must include the customer’s security token in the HTTP Authorization Header as described in Section 2.12.1.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP13: Get Accounts
  + URL: /Customers/{CustomerNumber}/Accounts?apiKey={key}
  + Method: GET
  + Input :
    - CustomerNumber: ID that identifies the customer
  + Output: List of Account Info (see schema: 6.1.29)
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an ApiKey which will be validated by Apigee. This operation must include the customer’s security token in the HTTP Authorization Header as described in Section 2.12.1.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP14: Get Account
  + URL: /Customers/{CustomerNumber}/Account/{AccountNo}?apiKey={key}
  + Method: GET
  + Input :
    - CustomerNumber: ID that identifies the customer
    - AccountNo: Identifier of an account
  + Output: Account Info (see schema: 6.1.30)
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an ApiKey which will be validated by Apigee. This operation must include the customer’s security token in the HTTP Authorization Header as described in Section 2.12.1.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP15: Update Account
  + URL: /Customers/{CustomerNumber}/Accounts/{AccountNo}?apiKey={key}
  + Method: PUT
  + Input :
    - CustomerNumber: ID that identifies the customer
    - AccountNo: identifier of the account
    - Account Info: an object describing an account
  + Output: None
  + Description: Pass-thru the request to the CustomerAccount Gateway to update the attributes of an account.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. This operation must include the customer’s security token in the HTTP Authorization Header as described in Section 2.12.1.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP16: Get Credit Balance
  + URL: /Customers/{CustomerNumber}/Credits/Balance?apiKey={key}
  + Method: GET
  + Input :
    - CustomerNumber: ID that identifies the customer
  + Output: CreditsBalance (see schema: 6.1.32)
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an ApiKey which will be validated by Apigee. This operation must include the customer’s security token in the HTTP Authorization Header as described in Section 2.12.1.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP17: Get Credit Use History
  + URL: /Customers/{CustomerNumber}/Credits/History?apiKey={key}
  + Method: GET
  + Input :
    - CustomerNumber: ID that identifies the customer
  + Output: CreditsHistory (see schema: 6.1.33)
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an ApiKey which will be validated by Apigee. This operation must include the customer’s security token in the HTTP Authorization Header as described in Section 2.12.1.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP18: Add a Credit Card
  + URL: /Customers/{CustomerNumber}/CreditCard?apiKey={key}
  + Method: POST
  + Input :
    - CustomerNumber: ID that identifies the customer
    - CreditCard (see schema 6.1.34)
  + Output: AccountInfo (see schema: 6.1.30)
  + Description: Pass-thru the request to the CreditCard Gateway
  + Security: The caller will provide an ApiKey which will be validated by Apigee. This operation must include the customer’s security token in the HTTP Authorization Header as described in Section 2.12.1.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP19: Create A New Customer
  + URL: /CreateCustomer?apiKey={key}
  + Method: POST
  + Input :
    - Email Address: Login email address of the new customer
  + Output: CustomerNumber
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Unlike other operations for this endpoint, this operation will NOT require a customer to have been previously authenticated.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.

The operation simply accepts an email address in the HTTP body of the POST request. If the creation of the customer within the source system (Customer Account Gateway) is successful, then the output will include the customer number as the HTTP body of the response. The HTTP Response Code will be set to 201 (Created) and a Location header will include the URL of the profile that has been created (ex: /Customers/{new Customer Number}/Profile).

#### Policy components

Policy components get plugged into the Apigee pipeline stages at the appropriate spots to implement the various mapping, transformation, and caching behaviors required. By convention, each policy component will include a name that expresses its use case and its version. The transformation components that will be required will include but are not limited to:

* Transform Source Product data into External Product
* Transform Source Top20 data into ExternalTop20
* Transform Source Store into External Store
* Transform Source Inventory updates into External Inventory
* Split Source Products data into multiple caches based upon the release window of the product
* Other policy components will be required that:
  + Search through Product, Store & Inventory caches to find items that match input parameters
  + Propogate Store and Inventory item change notifications to all caches
  + Validate authentication keys (ApiKey)
  + Validate customer authentication tokens (SWT tokens)

#### Caches

Data retrieved from the Source System proxies will be cached by the Apigee layer and then used to serve up results to API consumers. The caches will be segregated by data type and version. The caches that will be required are:

* Products Default Cache
* Products In-Circulation Cache
* Products Coming Soon Cache
* Titles Default Cache
* Titles Coming Soon Cache
* Stores Bulk Cache
* Stores IndividualItem Cache
* Top20 P7 Cache
* Top20 P30 Cache
* Inventory Cache
* Trailers Cache

#### Monitoring

Apigee’s ControlCenter utility will provide monitoring of the Services, Endpoints, PolicyComponents and Caches. It will report on the overall status, errors, error rates, message rates, response time and throughput of each component.

#### Alerting

Apigee ControlCenter will be used to monitor the status and performance. It provides Alert panels that display errors. It will be configured to forward alert notifications to Redbox {tbd}.

#### Reporting

*Apigee’s Analytics Center utility will be used to generate data for reports for Release 1*.

Existing Components

The following components already exist in the Redbox environment and will be re-used:

### EPC Web Service

The EPC service will provide the raw data for the Products Service. It must be modified to add a new attribute for the external GUID identifier. It will be called by the **Products Service Gateway** (see 2.7.1). The gateway will call the EPC service to get all Products in bulk.

### Batch Scheduler

The batch scheduler, Quartz, will be configured to execute each of the Cache Refresh Batch Jobs on a daily basis.

### Web Reservation Pipeline

The reservation pipeline service will be called by the Reservation service to price carts and process reservations.

### Caching Services

The OpenServices gateways will use the shared AppFabric Caching cluster to store Product and Store metadata in cache. This cached metadata will consist of ID maps that allow the gateways to map incoming IDs in external format into internal Redbox IDs that can be used when calling the Web Reservation Pipeline and other internal services.

### Web Services Monitoring

SiteScope will be used to monitor the health of the OpenServices gateways. Sitescope will monitor the following custom performance counters kept by the gateways:

|  |  |
| --- | --- |
|  | **Open Services** |
| Products | Number of total EPC failure calls per second |
| Stores | Number of total Store Service failures per second |
| Top20 | Total Top20 service failures per second |
|  | Total EPC failures per second |
|  | **Cache Update Service** |
| Products | # of failures when queuing Product refreshes/sec |
|  | # of failed Product refreshes/sec |
| Stores | # of failures when queuing Stores refreshes/sec |
|  | # of failed Stores refreshes/sec |
| Top20 | # of failures when queuing Top20 refreshes/sec |
|  | # of failed Top20 refreshes/sec |

In addition, Sitescope will periodically call the ‘Status Monitor’ operation of each Service (Apigee endpoint) and internal gateway and record the status returned. For instance, Sitescope will call /Products/StatusMonitor at the Apigee Products endpoint. This Apigee operation will in turn call the Products Gateway’s /StatusMonitor operation. This will provide Sitescope with an end-to-end status of the connectivity and operation of the Products service. Sitescope will also call the Products internal Gateway’s /StatusMonitor operation directly in order to provide visibility into the health of the internal gateway.

## Physical Deployment

The planned physical deployment is depicted in Figure 2.8‑1. End-users of the Bing/MSN and Digital applications communicate over the Internet to their respective applications. In the case of Bing/MSN, end-users will not communicate directly with the OpenAPI. Bing/MSN applications will communicate over the Internet to the OpenAPI on behalf of their users. The Apigee cloud service will be the public end-point for the OpenAPI.

The Apigee Cloud Service will communicate with Redbox by calling the Open Services Layer over the Internet. The external Redbox firewall will be configured to allow only traffic from Apigee’s servers into the Open Services Layer. Only HTTP (port 80) traffic will be allowed.

The Open Services Layer will consist of a farm of web servers in a DMZ network segment in the Lisle Datacenter. Traffic will be routed via round-robin to these services via a load-balancer.

Traffic between the Open Services Layer and the internal network of source applications will pass through another firewall. The Change Notifier components will execute with their host/source systems. The Cache Refresh Queues and ItemChangedEventQueuesand the **Source System Change Service** will execute in the Open Services Layer on those servers. The firewall between the Open Services Layer and the internal applications will be configured to allow port 80 traffic from the Open Systems layer to the Internal Network and allow MQ traffic from the Internal Network to the Open Systems layer.



Figure ‑

## Database Logical Design

At this time there is no database required specifically for the OpenAPI.

This section of the design document is reserved for use in a subsequent revision of the design.

Table Descriptions:

|  |  |  |
| --- | --- | --- |
| Table | Primary Key | Notes |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Open Systems Layer: Infrastructure Components & Dependencies

The custom-built components of the Open System Layer (such as the proxies and message handlers) will be built using standard Redbox technology components.

### Custom-built .NET Web Services (Product, Store, Top20 proxies)

The Source System Proxies will be built as WCF REST services. The input and output data will be structured as XML messages.

It is recommended that the solution structure follow the pattern established by EPC:

* Endpoint Project: hosts the WCF service methods and WCF Service Host
* Contracts Project: houses the data contracts
* Service Façade Project: houses the implementations of the service methods.

### Source System Change Service

The **Source System Change Service** will be implemented as a Windows Service. Standard technical components will include:

* XmlSerializer (instead of DataContractSerializer)
* IOC: Autofac, Autofac.Integration.WCF
* Logging: Common.Logging, Common.Logging.Log4Net, Log4Net
* Service Validation: CuttingEdge.Conditions

### Logging & Instrumentation

The **Source System Change Service** and the proxies of the Open Systems Layer will record activity metrics. The metrics that will be recorded include:

* Count of Transactions (processed by this instance of the service)
* Count of Successful Transactions
* Count of Failed Transactions
* Response Time for each service call

These metrics will be recorded as Windows Performance Counters.

The Windows Performance Counters will also be used to record the running sum of all response times for all service calls to the service instance. Windows performance monitoring will then be used to calculate the following rates:

* Transactions/sec
* SuccessTransactions/sec
* Failedransactions/sec
* Avg Response Time (i.e., Running sum / Count of Txns)

Log4Net will be used to record other log information to text log files. Such information will include error and warning messages captured during execution of the service.

## Design Approaches to Non-Functional Qualities (availability, performance, scaling)

For Release 1 of OpenAPI, the design of the services exposed by OpenAPI is very simple. It is anticipated that Bing/MSN will interact with the services very infrequently (approx. once per day).

### Availability

Apigee provides a virtualized service; therefore availability of the Apigee-hosted endpoints will be high.

The source-system service proxies in the Open Services Layer will be hosted on a set of load-balanced web servers which will provide for redundancy in case any one of the servers goes down.

### Performance

#### Response Time (External)

No specific requirement has been confirmed for response time. However, the risk of large response times is mitigated by the use of pre-computed and pre-fetched caches of data held within the Apigee layer.

#### Throughput (External)

No specific requirement has been confirmed for throughput.

#### Response Time (Source System Proxies)

No specific requirement has been confirmed for the response times of each source system gateway.

#### Throughput (Source System Proxies)

No specific requirement has been confirmed for the throughput of each source system gateway.

### Scale

*TBD for R2*

## Security

### OpenAPI Service Endpoint Security

Access to the OpenAPI Service Endpoints will be secured by an “API Key”. This is a secret key known only by Partners/Affiliates and Apigee. Clients will provide the API Key with each request. The Client Request stage of Apigee processing will confirm the validity of the key before allowing the request to be processed.

Operations that make use of sensitive customer data will be further secured. The apiKey mechanism will be used to control which partner/affiliates may submit these operation requests on behalf of users. In addition, the customer (user) must also be securely identified.

This design makes the following assumptions about user authentication (and is subject to change pending Partner integration design):

* Each of the secured operations must be accompanied with an HTTP Authorization Header that identifies the Customer.
* The contents of the HTTP Authorization Header will be an OAuth token that has been created by and signed by a trusted Identity Provider (this may be a Digital Partner or it may be Redbox).
* The OAuth token will be mappable to a Customer Number.

### Open Services Layer – Source System Gateway Security

Access to the source system proxies will be controlled so that only two types of callers may interact with the services:

* Apigee – The Redbox firewall will be configured to deny all external internet traffic, except for traffic originating from Apigee (i.e., “white listing”). This will be done by identifying all of the Apigee servers that should be able to send traffic and configuring the firewall with their IP addresses.
* Other Redbox applications – The firewall will be configured to allow HTTP traffic (port 80) from any redbox internal server.

*{Need additional info on security for authentication/authorization of internal apps calling this endpoint.}*

## Dependencies & Assumptions

* BingMSN will be able to consume the Products list as-is without the need for a custom schema.
* BingMSN will be able to consume the Top20 list as-is without the need for a custom schema.
* BingMSN will be able to consume the Kiosk list, in bulk, as-is without the need for a custom schema.
* EPC will be able to add an external identifier (GUID) to each product entry.
* The Rental systems team will be able to modify the KioskClient table with the addition of an external identifier(GUID).
* ODS will be able to provide Rental History data.
* Redbox.com will be modified to provide Recommendations as an application service.
* Redbox.com will be modified to support web single sign-on.
* Redbox.com will be modified to support the rental credits (appropriate modifications will be made to the reservation pipeline that can be shared with OpenAPI).

# Open Issues

| **#** | **Issue** | **Owner** |
| --- | --- | --- |
|  | Appropriate use of Apigee constructs. Apigee leverages a hierarchy of ‘Domain/Application/Service/Endpoint’ when building service proxies and configuring and enforcing policies. How should we organize this for maximum effectiveness?  Ex:   * OpenAPI/Products/Bulk/V1, * Redbox/OpenAPI/Products/Bulk-v1? | Chris Rudolphi – will coordinate with Saggezza |
|  | ~~Eventual Scope of OpenAPI: will it include digital partner integration, ingestion & sync of partner’s product catalog?~~ | ~~Dennis /Imran~~ |
|  | SLAs for response time for each operation need to be defined | Chris Rudolphi |
|  | Do we need SLA reporting for policies? | Chris Rudolph/Saggezza |
|  | What alerting is required? How (email, SMS, SNMP)? | Chris Rudolphi |
|  | BingMSN requested a customized schema for the Products list. Do we comply with that request? If so, we’ll need to add that detail to this design. | Dennis |
|  | ~~Value Tracking: design of how to correlate API activity with rental activity not yet fully defined~~ | ~~Chris Rudolphi~~ |
|  | ~~Exposure of internal Redbox identifiers: Redbox.com exposes the internal movie product identifier (ProductNumber) in several spots on the site (such as thru image tags and data provided via its JavaScript api). The EPC product metadata also includes the ProductNumber as part of the image URL. Need to define a path to remediating this (not strictly an OpenAPI problem)~~ | ~~Saad Rehmani~~ |
|  | Need further definition of the authentication/authorization design for when internal Redbox applications call the OpenServices layer proxies. | Chris Rudolphi |
|  | ~~Need to confirm with Data Services whether OpenAPI will use an ‘ODS’ data source for kiosk and Top20 information (rather than the RBDB and ProductsSvc as currently defined).~~ | ~~Mike Galvin to coordinate with Bart and Chris to finalize decision.~~ |
|  | ~~Need to confirm mechanism for securing Admin Endpoints.~~ | ~~Chris with Sweta~~ |
|  | Need to confirm monitoring infrastructure that will be used (Apigee facilities alone, or integration with CorpIT) | Chris |
|  | Business rules for validating Reservation Cart additions/changes need to be confirmed/defined. Potential for these to affect the architecture. | Dennis to document. |
|  | ~~How, or even IF, a credit must be specified and applied to a Cart is not yet confirmed. Need to confirm if Reservation service will support application of other promotional discounts (such as Web Credits) also needs to be defined.~~ | ~~Eric~~ |
|  | ~~Need to confirm how we want to track devices. Business requirement is for a “Device” identifier to be included in the Reservation service input and stored with the RBDB Reservation/Transaction table(s). What is the format for this value? What components?~~ | ~~Chris, Eric~~ |
|  | ~~Need to confirm if geocoding of customer locations will be a responsibility of consumer applications (web, CE, STB) or the Stores service. If the latter, need to select and contract with a vendor.~~ | ~~Chris~~ |
|  | ~~Need to define the registry of allowed URIs for the ‘rel’ attribute of atom:links when describing Trailers for Movies~~ | Chris & Dennis |
|  | Need to resolve which vocabulary of Genres will be supported when setting favorite Genres in Customer account service (and what is supported in underlying Customer Profile service) | Chris & CP team |
|  | ~~Need to design how Trailers will be represented in Products metadata.~~ | Chris & Dennis |

# Risks

|  |  |  |
| --- | --- | --- |
| **#** | **Risk** | **Mitigation Approach(es)** |
| 1. | Introduction of New Technology: Apigee. The use of Apigee as a technology and as a service provider is new to Redbox. The learning curve might be steeper than anticipated. The product/service might not be capable of all features we need. | * Leverage Saggezza because they have significant prior experience. * The OpenAPI project is starting small/simple by focusing first on the relatively simple BingMSN requirements. |
| 2. | Exposing data from internal systems might require enhancements/changes to those systems (ex: the addition of external GUIDs to EPC and KioskClient). Such changes might conflict (in terms of schedule/priority) with the development plans for those systems. | * Decouple the source systems from the OpenAPI where possible. The Source System proxies can mitigate some of this risk by performing mapping and transformations that the source systems might not be able to do at this time. * Align development schedules/releases when possible. |
| 3. | The design makes use of queues as the device for decoupling the OpenAPI infrastructure from the internal applications. The use of MQ is not fully mature within Redbox. This may present risks to our ability to provision queues on a timely basis to meet the needs of the OpenAPI team, and risks that the organization may not be able to operate/maintain them reliably. | The detailed design will isolate the use of MQ behind a pluggable interface. If needed, we can fall back to the use of databases as pseudo-queues until the MQ infrastructure matures. |
| 4. |  |  |
| 5. |  |  |

# Glossary

| **Term** | **Definition** |
| --- | --- |
| API | Application Programming Interface. The mechanism used for programmatic manipulation of the data and capabilities of a system. |
| Apigee | Vendor providing a cloud-based API management infrastructure. |
| Cache | A temporary storage of a copy of data. Caches are accessed in lieu of the source data in order to save time and/or to avoid unnecessary load on the source system. Caches typically expire after a set time period. |
| Cache Prefetch | The act of obtaining data from the source system and populating a cache in advance of a request for that data from the cache. This avoids making a request wait while the source system is contacted when the cache is empty. |
| Cloud Provider | Hosting of services “in the internet” as a virtual service. Avoids the cost of provisioning servers in a company’s datacenter. |
| EPC | Enterprise Product Catalog; Redbox’s strategic direction for a centralized master metadata store for product information. |
| GUID | Globally Unique Identifier; a special data type that guarantees the uniqueness of each value. |
| IOC | “Inversion Of Control”, is an abstract principle describing an aspect of some software architecture designs in which the flow of control of a system is inverted in comparison to procedural programming. Typically assisted with the use of an IOC container utility such as Autofac. |
| ODS | Operational Data Store; provides Kiosk, Top20, and Inventory data pulled from RBDB and other source systems. |
| Policy Component | Within the Apigee framework, each step of processing is accomplished through the use of a ‘policy component’. These components enforce the security, caching, data transformation, and management policies of the services one implements within Apigee. |
| Proxy | In Web Services, a proxy is a wrapper around a source system that isolates the source system from the callers of the web service. The OpenAPI architecture is made up of two layers of proxies: the Apigee layer and the source-systems layer of proxies. |
| Rate Limiting | Rate limiting controls how often a caller may invoke a service. Ex: caller A may call a service 10 times per day while caller B will be allowed to call it 100 times per minute. |
| REST | A style of Web Service that leverages a light-weight set of conventions for exchanging information (as compared to SOAP style of services). |
| Schema | A formal description of the shape of data. This design uses XML schemas to describe the format of data/messages at each endpoint. |

# 

# Appendices

## Appendix A: Message Contracts & Schemas

As established in this appendix and examples, some Product APIs include elements that contain a set of <atom:link> elements. Atom is an industry standard XML schema for publishing metadata and the “Link” element is the conventional mechanism for documenting navigational references (where the user might go next). The Link element relies on two primary attributes, the href which contains the URL of the navigational link, and the ‘rel’ attribute which describes the type of link (ie, it describes the relationship between the containing item and the link). Atom:Link elements are used in several places throughout this schema specification:

* To convey the links to related ‘pages’ of items when paging a result set (next, previous, etc).
* To document the links to boxart images.

When documenting the links to boxart images, the ‘rel’ attribute of each <atom:link> will be used to describe the type of image, using the a list of URIs established herein:

* rel="http://api.redbox.com/Links/BoxArt/Original”
* rel="http://api.redbox.com/Links/BoxArt/Thumbnail”
* rel="http://api.redbox.com/Links/BoxArt/Thumbnail(150)”
* rel="http://api.redbox.com/Links/BoxArt/Full"
* rel="http://api.redbox.com/Links/BoxArt/Kiosk"
* rel="http://api.redbox.com/Links/BoxArt/Detail(370)"

### Common Schemas

The following schema defines the Paging and Extensions elements used throughout this design. The Paging element also uses <atom:link> elements to identify the URLs for the Next, Previous, First, and Last pages of the set. The <atom:link> ‘rel’ URIs used for paging are:

<atom:link rel="first" href="http://uri" />

<atom:link rel="last" href="http://uri" />

<atom:link rel="prev" href="http://uri" />

<atom:link rel="next" href="http://uri" />

<xs:schema

    xmlns:tns="http://api.redbox.com/OpenAPI/v1"

    elementFormDefault="qualified"

    targetNamespace="http://api.redbox.com/OpenAPI/v1"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

    xmlns:atom="http://www.w3.org/2005/Atom"

>

  <xs:import namespace="http://www.w3.org/2005/Atom" />

  <xs:complexType name="Paging" mixed="false" >

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="4" ref="atom:link" xmlns:atom="http://www.w3.org/2005/Atom" />

    </xs:sequence>

    <xs:attribute name="pageNum" type="xs:integer" use="optional"  />

    <xs:attribute name="pageSize" type="xs:integer" use="optional" />

    <xs:attribute name="totalNumItems" type="xs:integer" use="optional"  />

    <xs:attribute name="pageCount" type="xs:integer" use="optional"  />

  </xs:complexType>

  <xs:complexType name="ExtensionData">

    <xs:sequence>

      <xs:any minOccurs="0" maxOccurs="unbounded" />

    </xs:sequence>

  </xs:complexType>

</xs:schema>

### Source Movies Schema

NOTE: This has changed from R1 (new schema version).

Reference for http://localhost:56745/SourceAPIProducts/Products/Movies

**Url:** http://localhost:56745/SourceAPIProducts/Products/Movies

**HTTP Method:** GET

|  |  |  |
| --- | --- | --- |
| **Message direction** | **Format** | **Body** |
| Request | N/A | The Request body is empty. |
| Response | Xml | [Example](http://localhost:56745/sourceAPIProducts/help/operations/GetAllSourceMovies#response-xml),[Schema](http://localhost:56745/sourceAPIProducts/help/operations/GetAllSourceMovies#response-schema) |

The following is an example response Xml body:

<Products lastUpdated="1900-01-01T01:01:01" xmlns="http://api-source.redbox.com/Products/v2">

  <Movie productId="productId1" displayOnWebsite="true" format="format1" websiteUrl="http://uri1">

    <Title>Title1</Title>

    <SortTitle>SortTitle1</SortTitle>

    <RedboxReleaseDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    <RedboxComingSoonDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    <StreetReleaseDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    <ReleaseYear xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    <Genres>

      <Genre xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      <Genre>Genre1</Genre>

      <Genre xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    </Genres>

    <BoxArtImages>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Original” xmlns="http://www.w3.org/2005/Atom"></link>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Thumbnail” xmlns="http://www.w3.org/2005/Atom"></link>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Thumbnail(150)” xmlns="http://www.w3.org/2005/Atom"></link>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Full” xmlns="http://www.w3.org/2005/Atom"></link>

    </BoxArtImages>

    <Extensions />

    <ExternalIds>

      <ExternalId source="Baseline" description="description1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      <ExternalId source="UPC" description="description2">ExternalId1</ExternalId>

      <ExternalId source="DigitalPartner" description="description3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    </ExternalIds>

    <DomesticHomeVideoDistributor>DomesticHomeVideoDistributor1</DomesticHomeVideoDistributor>

    <DomesticTheatricalDistributor>DomesticTheatricalDistributor1</DomesticTheatricalDistributor>

    <SynopsisShort>SynopsisShort1</SynopsisShort>

    <SynopsisLong>SynopsisLong1</SynopsisLong>

    <RunningLength xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    <MPAARating xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    <Actors>

      <Person xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      <Person>Person1</Person>

      <Person xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    </Actors>

    <Directors>

      <Person xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      <Person>Person2</Person>

      <Person xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    </Directors>

  </Movie>

  <Movie productId="productId2" displayOnWebsite="false" format="format2" websiteUrl="http://uri2">

    <Title>Title2</Title>

    <SortTitle>SortTitle2</SortTitle>

    <RedboxReleaseDate>1900-01-01</RedboxReleaseDate>

    <RedboxComingSoonDate>1900-01-01</RedboxComingSoonDate>

    <StreetReleaseDate>1900-01-01</StreetReleaseDate>

    <ReleaseYear>0</ReleaseYear>

    <Genres>

      <Genre>Genre2</Genre>

      <Genre xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      <Genre>Genre3</Genre>

    </Genres>

    <BoxArtImages>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Original” xmlns="http://www.w3.org/2005/Atom"></link>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Thumbnail” xmlns="http://www.w3.org/2005/Atom"></link>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Thumbnail(150)” xmlns="http://www.w3.org/2005/Atom"></link>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Full” xmlns="http://www.w3.org/2005/Atom"></link>

    </BoxArtImages>

    <Extensions />

    <ExternalIds>

      <ExternalId source="Baseline" description="description4">ExternalId2</ExternalId>

      <ExternalId source="UPC" description="description5" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      <ExternalId source="DigitalPartner" description="description6">ExternalId3</ExternalId>

    </ExternalIds>

    <DomesticHomeVideoDistributor>DomesticHomeVideoDistributor2</DomesticHomeVideoDistributor>

    <DomesticTheatricalDistributor>DomesticTheatricalDistributor2</DomesticTheatricalDistributor>

    <SynopsisShort>SynopsisShort2</SynopsisShort>

    <SynopsisLong>SynopsisLong2</SynopsisLong>

    <RunningLength>P396DT1H1M1S</RunningLength>

    <MPAARating>G</MPAARating>

    <Actors>

      <Person>Person3</Person>

      <Person xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      <Person>Person4</Person>

    </Actors>

    <Directors>

      <Person>Person5</Person>

      <Person xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      <Person>Person6</Person>

    </Directors>

  </Movie>

</Products>

The following is the response Xml Schema:

<xs:schema xmlns:tns="http://api-source.redbox.com/Products/v2" elementFormDefault="qualified"

           targetNamespace="http://api-source.redbox.com/Products/v2"

           xmlns:xs="http://www.w3.org/2001/XMLSchema">

 <xs:import namespace="http://api.redbox.com/OpenAPI/v1"  />

  <xs:import namespace="http://www.w3.org/2005/Atom" />

  <xs:element name="Products" nillable="true" type="tns:SourceProductList" />

  <xs:complexType name="SourceProductList">

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="Movie" type="tns:Movie" />

    </xs:sequence>

    <xs:attribute name="lastUpdated" type="xs:dateTime" use="required" />

  </xs:complexType>

  <xs:complexType name="Movie">

    <xs:complexContent mixed="false">

      <xs:extension base="tns:Product">

        <xs:sequence>

          <xs:element minOccurs="0" maxOccurs="1" name="ExternalIds" type="tns:ArrayOfExternalId" />

          <xs:element minOccurs="0" maxOccurs="1" name="DomesticHomeVideoDistributor" type="xs:string" />

          <xs:element minOccurs="0" maxOccurs="1" name="DomesticTheatricalDistributor" type="xs:string" />

          <xs:element minOccurs="0" maxOccurs="1" name="SynopsisShort" type="xs:string" />

          <xs:element minOccurs="0" maxOccurs="1" name="SynopsisLong" type="xs:string" />

          <xs:element minOccurs="1" maxOccurs="1" name="RunningLength" type="xs:duration" nillable="true" />

          <xs:element minOccurs="1" maxOccurs="1" name="MPAARating" type="tns:MPAARatings" nillable="true" />

          <xs:element minOccurs="0" maxOccurs="1" name="Actors" type="tns:ArrayOfPerson" />

          <xs:element minOccurs="0" maxOccurs="1" name="Directors" type="tns:ArrayOfPerson" />

        </xs:sequence>

      </xs:extension>

    </xs:complexContent>

  </xs:complexType>

  <xs:complexType name="Product" abstract="true">

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="1" name="Title" type="xs:string" />

      <xs:element minOccurs="0" maxOccurs="1" name="SortTitle" type="xs:string" />

      <xs:element minOccurs="1" maxOccurs="1" name="RedboxReleaseDate" type="xs:date" nillable="true" />

      <xs:element minOccurs="1" maxOccurs="1" name="RedboxComingSoonDate" type="xs:date" nillable="true" />

      <xs:element minOccurs="1" maxOccurs="1" name="StreetReleaseDate" type="xs:date" nillable="true" />

      <xs:element minOccurs="1" maxOccurs="1" name="ReleaseYear" type="xs:unsignedShort" nillable="true" />

      <xs:element minOccurs="0" maxOccurs="1" name="Genres" type="tns:ArrayOfGenre" />

      <xs:element minOccurs="0" maxOccurs="1" name="BoxArtImages" type="tns:BoxArtLinks" />

      <xs:element minOccurs="0" maxOccurs="1" name="Extensions" xmlns:q2="http://api.redbox.com/OpenAPI/v1" type="q2:ExtensionData" />

    </xs:sequence>

    <xs:attribute name="productId" type="xs:string" />

    <xs:attribute name="displayOnWebsite" type="xs:boolean" use="required" />

    <xs:attribute name="format" type="xs:string" />

    <xs:attribute name="websiteUrl" type="xs:anyURI" />

  </xs:complexType>

  <xs:complexType name="ArrayOfGenre">

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="Genre" nillable="true" type="tns:Genre" />

    </xs:sequence>

  </xs:complexType>

  <xs:complexType name="Genre">

    <xs:simpleContent>

      <xs:extension base="xs:string" />

    </xs:simpleContent>

  </xs:complexType>

  <xs:complexType name="ArrayOfExternalId">

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="ExternalId" nillable="true" type="tns:ExternalId" />

    </xs:sequence>

  </xs:complexType>

  <xs:complexType name="ExternalId">

    <xs:simpleContent>

      <xs:extension base="xs:string">

        <xs:attribute name="source" type="tns:ExternalIdProviders" use="required" />

        <xs:attribute name="description" type="xs:string" />

      </xs:extension>

    </xs:simpleContent>

  </xs:complexType>

  <xs:simpleType name="ExternalIdProviders">

    <xs:restriction base="xs:string">

      <xs:enumeration value="Baseline" />

      <xs:enumeration value="UPC" />

      <xs:enumeration value="DigitalPartner" />

    </xs:restriction>

  </xs:simpleType>

  <xs:simpleType name="MPAARatings">

    <xs:restriction base="xs:string">

      <xs:enumeration value="G" />

      <xs:enumeration value="PG" />

      <xs:enumeration value="PG-13" />

      <xs:enumeration value="R" />

      <xs:enumeration value="NR" />

    </xs:restriction>

  </xs:simpleType>

  <xs:complexType name="ArrayOfPerson">

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="Person" nillable="true" type="tns:Person" />

    </xs:sequence>

  </xs:complexType>

  <xs:complexType name="Person">

    <xs:simpleContent>

      <xs:extension base="xs:string" />

    </xs:simpleContent>

  </xs:complexType>

 <xs:complexType name="BoxArtLinks" >

   <xs:sequence>

     <xs:element  minOccurs="0" maxOccurs="unbounded" ref="atom:link" xmlns:atom="http://www.w3.org/2005/Atom" />

   </xs:sequence>

 </xs:complexType>

</xs:schema>

### External Movie Schema

Reference for http://localhost:56745/RedboxAPIProducts/Products/Movies

**Url:** http://localhost:56745/RedboxAPIProducts/Products/Movies

**HTTP Method:** GET

|  |  |  |
| --- | --- | --- |
| **Message direction** | **Format** | **Body** |
| Request | N/A | The Request body is empty. |
| Response | Xml | [Example](http://localhost:56745/redboxapiproducts/help/operations/GetAllMovies#response-xml),[Schema](http://localhost:56745/redboxapiproducts/help/operations/GetAllMovies#response-schema) |

The following is an example response Xml body:

<ProductList lastUpdated="1900-01-01T01:01:01" xmlns="http://api.redbox.com/Products/v2">

  <Movies>

    <Movie productId="productId1" format="format1" websiteUrl="http://uri1" xmlns="http://api.redbox.com/Products/MovieList/v2">

      <Title>Title1</Title>

      <SortTitle>SortTitle1</SortTitle>

      <RedboxReleaseDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      <RedboxComingSoonDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      <StreetReleaseDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      <ReleaseYear xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      <Genres>

        <Genre>Genre1</Genre>

        <Genre>Genre2</Genre>

        <Genre>Genre3</Genre>

      </Genres>

      <BoxArtImages>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Original” xmlns="http://www.w3.org/2005/Atom"></link>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Thumbnail” xmlns="http://www.w3.org/2005/Atom"></link>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Thumbnail(150)” xmlns="http://www.w3.org/2005/Atom"></link>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Full” xmlns="http://www.w3.org/2005/Atom"></link>

      </BoxArtImages>

      <Extensions />

      <ExternalIds>

        <ExternalId source="Baseline" description="description1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

        <ExternalId source="UPC" description="description2">ExternalId1</ExternalId>

        <ExternalId source="DigitalPartner" description="description3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      </ExternalIds>

      <DomesticHomeVideoDistributor>DomesticHomeVideoDistributor1</DomesticHomeVideoDistributor>

      <DomesticTheatricalDistributor>DomesticTheatricalDistributor1</DomesticTheatricalDistributor>

      <SynopsisShort>SynopsisShort1</SynopsisShort>

      <SynopsisLong>SynopsisLong1</SynopsisLong>

      <RunningLength xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      <MPAARating xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      <Actors>

        <Person>Person1</Person>

        <Person>Person2</Person>

        <Person>Person3</Person>

      </Actors>

      <Directors>

        <Person>Person4</Person>

        <Person>Person5</Person>

        <Person>Person6</Person>

      </Directors>

    </Movie>

    <Movie productId="productId2" format="format2" websiteUrl="http://uri2" xmlns="http://api.redbox.com/Products/MovieList/v2">

      <Title>Title2</Title>

      <SortTitle>SortTitle2</SortTitle>

      <RedboxReleaseDate>1900-01-01</RedboxReleaseDate>

      <RedboxComingSoonDate>1900-01-01</RedboxComingSoonDate>

      <StreetReleaseDate>1900-01-01</StreetReleaseDate>

      <ReleaseYear>0</ReleaseYear>

      <Genres>

        <Genre>Genre4</Genre>

        <Genre>Genre5</Genre>

        <Genre>Genre6</Genre>

      </Genres>

      <BoxArtImages>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Original” xmlns="http://www.w3.org/2005/Atom"></link>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Thumbnail” xmlns="http://www.w3.org/2005/Atom"></link>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Thumbnail(150)” xmlns="http://www.w3.org/2005/Atom"></link>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Full” xmlns="http://www.w3.org/2005/Atom"></link>

      </BoxArtImages>

      <Extensions />

      <ExternalIds>

        <ExternalId source="Baseline" description="description4">ExternalId2</ExternalId>

        <ExternalId source="UPC" description="description5" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

        <ExternalId source="DigitalPartner" description="description6">ExternalId3</ExternalId>

      </ExternalIds>

      <DomesticHomeVideoDistributor>DomesticHomeVideoDistributor2</DomesticHomeVideoDistributor>

      <DomesticTheatricalDistributor>DomesticTheatricalDistributor2</DomesticTheatricalDistributor>

      <SynopsisShort>SynopsisShort2</SynopsisShort>

      <SynopsisLong>SynopsisLong2</SynopsisLong>

      <RunningLength>P396DT1H1M1S</RunningLength>

      <MPAARating>G</MPAARating>

      <Actors>

        <Person>Person7</Person>

        <Person>Person8</Person>

        <Person>Person9</Person>

      </Actors>

      <Directors>

        <Person>Person10</Person>

        <Person>Person11</Person>

        <Person>Person12</Person>

      </Directors>

    </Movie>

  </Movies>

</ProductList>

The following is the response Xml Schema:

<xs:schema xmlns:tns="http://api.redbox.com/Products/v2"

           elementFormDefault="qualified"

           targetNamespace="http://api.redbox.com/Products/v2"

           xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:import namespace="http://api.redbox.com/OpenAPI/v1" />

  <xs:import namespace="http://api.redbox.com/Products/MovieList/v2" />

  <xs:element name="ProductList" nillable="true" type="tns:BulkProductResponse" />

  <xs:complexType name="BulkProductResponse">

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="1" name="Movies" xmlns:pml="http://api.redbox.com/Products/MovieList/v2" type="pml:ArrayOfMovie" />

    </xs:sequence>

    <xs:attribute name="lastUpdated" type="xs:dateTime" use="required" />

  </xs:complexType>

</xs:schema>

Additional response Xml Schemas:

<xs:schema xmlns:tns="http://api.redbox.com/Products/MovieList/v2"

           elementFormDefault="qualified"

           targetNamespace="http://api.redbox.com/Products/MovieList/v2"

           xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:import namespace="http://api.redbox.com/OpenAPI/v1" />

  <xs:import namespace="http://www.w3.org/2005/Atom" />

  <xs:complexType name="ArrayOfMovie">

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="Movie" nillable="false" type="tns:Movie" />

    </xs:sequence>

  </xs:complexType>

  <xs:complexType name="Movie">

    <xs:complexContent mixed="false">

      <xs:extension base="tns:Product">

        <xs:sequence>

          <xs:element minOccurs="0" maxOccurs="1" name="ExternalIds" type="tns:ArrayOfExternalId"  />

          <xs:element minOccurs="0" maxOccurs="1" name="DomesticHomeVideoDistributor" type="xs:string" />

          <xs:element minOccurs="0" maxOccurs="1" name="DomesticTheatricalDistributor" type="xs:string" />

          <xs:element minOccurs="0" maxOccurs="1" name="SynopsisShort" type="xs:string" />

          <xs:element minOccurs="0" maxOccurs="1" name="SynopsisLong" type="xs:string" />

          <xs:element minOccurs="1" maxOccurs="1" name="RunningLength" type="xs:duration" nillable="true"/>

          <xs:element minOccurs="1" maxOccurs="1" name="MPAARating" type="tns:MPAARatings" nillable="true"/>

          <xs:element minOccurs="0" maxOccurs="1" name="Actors" type="tns:ArrayOfPerson" />

          <xs:element minOccurs="0" maxOccurs="1" name="Directors" type="tns:ArrayOfPerson" />

        </xs:sequence>

      </xs:extension>

    </xs:complexContent>

  </xs:complexType>

  <xs:complexType name="Product" abstract="true">

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="1" name="Title" type="xs:string" />

      <xs:element minOccurs="0" maxOccurs="1" name="SortTitle" type="xs:string" />

      <xs:element minOccurs="1" maxOccurs="1" name="RedboxReleaseDate" type="xs:date" nillable="true"/>

      <xs:element minOccurs="1" maxOccurs="1" name="RedboxComingSoonDate" type="xs:date" nillable="true"/>

      <xs:element minOccurs="1" maxOccurs="1" name="StreetReleaseDate" type="xs:date" nillable="true"/>

      <xs:element minOccurs="1" maxOccurs="1" name="ReleaseYear" type="xs:unsignedShort" nillable="true"/>

      <xs:element minOccurs="0" maxOccurs="1" name="Genres" type="tns:ArrayOfGenre" />

      <xs:element minOccurs="0" maxOccurs="1" name="BoxArtImages" type="tns:BoxArtLinks" />

      <xs:element minOccurs="0" maxOccurs="1" name="Extensions" xmlns:oapi="http://api.redbox.com/OpenAPI/v1" type="oapi:ExtensionData" />

    </xs:sequence>

    <xs:attribute name="productId" type="xs:string" />

    <xs:attribute name="format" type="xs:string" />

    <xs:attribute name="websiteUrl" type="xs:anyURI" />

  </xs:complexType>

  <xs:complexType name="ArrayOfGenre">

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="Genre" type="tns:Genre" />

    </xs:sequence>

  </xs:complexType>

  <xs:complexType name="Genre">

    <xs:simpleContent>

      <xs:extension base="xs:string" />

    </xs:simpleContent>

  </xs:complexType>

  <xs:simpleType name="ExternalIdProviders">

    <xs:restriction base="xs:string">

      <xs:enumeration value="Baseline" />

      <xs:enumeration value="UPC" />

      <xs:enumeration value="DigitalPartner" />

    </xs:restriction>

  </xs:simpleType>

  <xs:simpleType name="MPAARatings">

    <xs:restriction base="xs:string">

      <xs:enumeration value="G" />

      <xs:enumeration value="PG" />

      <xs:enumeration value="PG-13" />

      <xs:enumeration value="R" />

      <xs:enumeration value="NR" />

    </xs:restriction>

  </xs:simpleType>

  <xs:complexType name="ArrayOfPerson">

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="Person" nillable="false" type="tns:Person" />

    </xs:sequence>

  </xs:complexType>

  <xs:complexType name="Person">

    <xs:simpleContent>

      <xs:extension base="xs:string" />

    </xs:simpleContent>

  </xs:complexType>

  <xs:complexType name="ExternalId">

    <xs:simpleContent>

      <xs:extension base="xs:string">

        <xs:attribute name="source" type="tns:ExternalIdProviders" use="required" />

        <xs:attribute name="description" type="xs:string" />

      </xs:extension>

    </xs:simpleContent>

  </xs:complexType>

  <xs:complexType name="ArrayOfExternalId">

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="ExternalId" nillable="true" type="tns:ExternalId" />

    </xs:sequence>

  </xs:complexType>

  <xs:complexType name="BoxArtLinks" >

    <xs:sequence>

      <xs:element  minOccurs="0" maxOccurs="unbounded" ref="atom:link" xmlns:atom="http://www.w3.org/2005/Atom" />

    </xs:sequence>

  </xs:complexType>

</xs:schema>

### External Products Browse Request Schema

Reference for http://localhost:56745/RedboxAPIProducts/Products/Movies?apiKey={KEY}

**Url:** http://localhost:56745/RedboxAPIProducts/Products/Movies?apiKey={KEY}&pageNum={PN}&pageSize={PS}

**HTTP Method:** POST

|  |  |  |
| --- | --- | --- |
| **Message direction** | **Format** | **Body** |
| Request | Xml | [Example](http://localhost:56745/redboxapiproducts/help/operations/BrowseMovies#request-xml),[Schema](http://localhost:56745/redboxapiproducts/help/operations/BrowseMovies#request-schema) |
| Response | Xml | [Example](http://localhost:56745/redboxapiproducts/help/operations/BrowseMovies#response-xml),[Schema](http://localhost:56745/redboxapiproducts/help/operations/BrowseMovies#response-schema) |

The following is an example request Xml body:

<MovieBrowseRequest xmlns="http://api.redbox.com/Products/v2">

<ProductId>1627aea5-8e0a-4371-9022-9b504344e724</ProductId>

<ProductId>1627aea5-8e0a-4371-9022-9b504344e724</ProductId>

</MovieBrowseRequest>

The following is the request Xml Schema:

<xs:schema xmlns:tns="http://api.redbox.com/Products/v2" elementFormDefault="qualified" targetNamespace="http://api.redbox.com/Products/v2" xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="MovieBrowseRequest" nillable="true" type="tns:MovieBrowseRequest" />

<xs:complexType name="MovieBrowseRequest">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="unbounded" name="ProductId" type="xs:string" />

</xs:sequence>

</xs:complexType>

</xs:schema>

### Product Search Request

The URL format for this request will be:

**Url:** http://localhost:56745/RedboxAPIProducts/Products/Search?apiKey={key}&pageNum={pN}&pageSize={pS}

The following is an example request Xml body:

<ProductSearchRequest xmlns="http://api.redbox.com/Products/v2">  
  <SearchCriteria searchField="Title" operator="contains" productTypes="DVD Blu-ray">text</SearchCriteria>  
  <SortOptions sortField="sortField1" sortOrder="asc" />  
</ProductSearchRequest>

The following is the request Xml Schema:

<xs:schema xmlns:tns="http://api.redbox.com/Products/v2"

           elementFormDefault="qualified"

           targetNamespace="http://api.redbox.com/Products/v2"

           xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:complexType name="ProductSearchRequestType" >

    <xs:sequence minOccurs="1" maxOccurs="1">

      <xs:element name="SearchCriteria" nillable="false" >

        <xs:complexType  mixed="true">

          <xs:attribute name="searchField" use="optional" default="Title" />

          <xs:attribute name="operator" use="optional" default="contains" type="tns:SearchOperators" />

          <xs:attribute name="productTypes" use="optional" default="DVD Blu-ray" type="tns:ProductTypesList" />

        </xs:complexType>

      </xs:element>

      <xs:element name="SortOptions" minOccurs="0" maxOccurs="1" >

        <xs:complexType  mixed="false">

          <xs:attribute name="sortField" use="required" type="xs:string" />

          <xs:attribute name="sortOrder" use="optional" type="tns:SortDir" />

        </xs:complexType>

      </xs:element>

    </xs:sequence>

  </xs:complexType>

  <xs:simpleType name="SearchOperators" >

    <xs:restriction base="xs:string">

      <xs:enumeration value="startsWith" />

      <xs:enumeration value="contains" />

    </xs:restriction>

  </xs:simpleType>

  <xs:simpleType name="ProductTypes">

    <xs:restriction base="xs:string">

      <xs:enumeration value="DVD"/>

      <xs:enumeration value="Blu-ray" />

      <xs:enumeration value="Games" />

      <xs:enumeration value="TV-series" />

    </xs:restriction>

  </xs:simpleType>

  <xs:simpleType name="ProductTypesList" >

    <xs:list itemType="tns:ProductTypes" />

  </xs:simpleType>

  <xs:simpleType name="SortDir" >

    <xs:restriction base="xs:string">

      <xs:enumeration value="asc" />

      <xs:enumeration value="desc" />

    </xs:restriction>

  </xs:simpleType>

  <xs:element name="ProductSearchRequest" type="tns:ProductSearchRequestType" />

</xs:schema>

### Paginated Product List

The following is an example response Xml body:

<PaginatedProductList lastUpdated="1900-01-01T01:01:01" xmlns="http://api.redbox.com/Products/v2">

  <Movies>

    <Movie productId="productId1" format="format1" websiteUrl="http://uri1" xmlns="http://api.redbox.com/Products/MovieList/v2">

      <Title>Title1</Title>

      <SortTitle>SortTitle1</SortTitle>

      <RedboxReleaseDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      <RedboxComingSoonDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      <StreetReleaseDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      <ReleaseYear xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      <Genres>

        <Genre>Genre1</Genre>

        <Genre>Genre2</Genre>

        <Genre>Genre3</Genre>

      </Genres>

      <BoxArtImages>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Original” xmlns="http://www.w3.org/2005/Atom"></link>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Thumbnail” xmlns="http://www.w3.org/2005/Atom"></link>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Thumbnail(150)” xmlns="http://www.w3.org/2005/Atom"></link>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Full” xmlns="http://www.w3.org/2005/Atom"></link>

      </BoxArtImages>

      <Extensions />

      <ExternalIds>

        <ExternalId source="Baseline" description="description1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

        <ExternalId source="UPC" description="description2">ExternalId1</ExternalId>

        <ExternalId source="DigitalPartner" description="description3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      </ExternalIds>

      <DomesticHomeVideoDistributor>DomesticHomeVideoDistributor1</DomesticHomeVideoDistributor>

      <DomesticTheatricalDistributor>DomesticTheatricalDistributor1</DomesticTheatricalDistributor>

      <SynopsisShort>SynopsisShort1</SynopsisShort>

      <SynopsisLong>SynopsisLong1</SynopsisLong>

      <RunningLength xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      <MPAARating xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      <Actors>

        <Person>Person1</Person>

        <Person>Person2</Person>

        <Person>Person3</Person>

      </Actors>

      <Directors>

        <Person>Person4</Person>

        <Person>Person5</Person>

        <Person>Person6</Person>

      </Directors>

    </Movie>

    <Movie productId="productId2" format="format2" websiteUrl="http://uri2" xmlns="http://api.redbox.com/Products/MovieList/v2">

      <Title>Title2</Title>

      <SortTitle>SortTitle2</SortTitle>

      <RedboxReleaseDate>1900-01-01</RedboxReleaseDate>

      <RedboxComingSoonDate>1900-01-01</RedboxComingSoonDate>

      <StreetReleaseDate>1900-01-01</StreetReleaseDate>

      <ReleaseYear>0</ReleaseYear>

      <Genres>

        <Genre>Genre4</Genre>

        <Genre>Genre5</Genre>

        <Genre>Genre6</Genre>

      </Genres>

      <BoxArtImages>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Original” xmlns="http://www.w3.org/2005/Atom"></link>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Thumbnail” xmlns="http://www.w3.org/2005/Atom"></link>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Thumbnail(150)” xmlns="http://www.w3.org/2005/Atom"></link>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Full” xmlns="http://www.w3.org/2005/Atom"></link>

      </BoxArtImages>

      <Extensions />

      <ExternalIds>

        <ExternalId source="Baseline" description="description4">ExternalId2</ExternalId>

        <ExternalId source="UPC" description="description5" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

        <ExternalId source="DigitalPartner" description="description6">ExternalId3</ExternalId>

      </ExternalIds>

      <DomesticHomeVideoDistributor>DomesticHomeVideoDistributor2</DomesticHomeVideoDistributor>

      <DomesticTheatricalDistributor>DomesticTheatricalDistributor2</DomesticTheatricalDistributor>

      <SynopsisShort>SynopsisShort2</SynopsisShort>

      <SynopsisLong>SynopsisLong2</SynopsisLong>

      <RunningLength>P396DT1H1M1S</RunningLength>

      <MPAARating>G</MPAARating>

      <Actors>

        <Person>Person7</Person>

        <Person>Person8</Person>

        <Person>Person9</Person>

      </Actors>

      <Directors>

        <Person>Person10</Person>

        <Person>Person11</Person>

        <Person>Person12</Person>

      </Directors>

    </Movie>

  </Movies>

  <Paging pageNum="1" pageSize="1" totalNumItems="1" pageCount="1" xmlns:atom="http://www.w3.org/2005/Atom">

<atom:link rel="first" href="http://uri" />

<atom:link rel="last" href="http://uri" />

<atom:link rel="prev" href="http://uri" />

<atom:link rel="next" href="http://uri" />

</Paging>   
</PaginatedProductList>

The following is the response Xml Schema:

This schema re-uses the **MovieList** schema defined in 8.1.3 External Movie List.

<xs:schema xmlns:tns="http://api.redbox.com/Products/v2"

           elementFormDefault="qualified"

           targetNamespace="http://api.redbox.com/Products/v2"

           xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:import namespace="http://api.redbox.com/OpenAPI/v1" />

  <xs:import namespace="http://api.redbox.com/Products/MovieList/v2" />

  <xs:element name="PaginatedProductList" nillable="true" type="tns:MovieBrowseResponse" />

  <xs:complexType name="MovieBrowseResponse">

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="1" name="Movies" xmlns:pml="http://api.redbox.com/Products/MovieList/v2" type="pml:ArrayOfMovie" />

      <xs:element minOccurs="0" maxOccurs="1" name="Paging" xmlns:oapi ="http://api.redbox.com/OpenAPI/v1" type="oapi:Paging" />

    </xs:sequence>

    <xs:attribute name="lastUpdated" type="xs:dateTime" use="required" />

  </xs:complexType>

</xs:schema>

### Titles

Example:

<Titles lastUpdated="1900-01-01T01:01:01" xmlns="http://api.redbox.com/Products/TitleList/v2">

  <Title>

    <TitleIdentifier source="Baseline" description="description1">TitleIdentifier1</TitleIdentifier>

    <Title>Title1</Title>

    <SortTitle>SortTitle1</SortTitle>

    <ReleaseYear xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    <Genres>

      <Genre>Genre1</Genre>

      <Genre>Genre2</Genre>

      <Genre>Genre3</Genre>

    </Genres>

    <DomesticHomeVideoDistributor>DomesticHomeVideoDistributor1</DomesticHomeVideoDistributor>

    <DomesticTheatricalDistributor>DomesticTheatricalDistributor1</DomesticTheatricalDistributor>

    <SynopsisShort>SynopsisShort1</SynopsisShort>

    <SynopsisLong>SynopsisLong1</SynopsisLong>

    <RunningLength xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    <MPAARating xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    <Actors>

      <Person>Person1</Person>

      <Person>Person2</Person>

      <Person>Person3</Person>

    </Actors>

    <Directors>

      <Person>Person4</Person>

      <Person>Person5</Person>

      <Person>Person6</Person>

    </Directors>

    <ProductRefs>

      <ProductRef productId="productId1" format="format1" websiteUrl="http://uri1">

        <Title>Title1</Title>

        <ExternalIds>

          <ExternalId source="Baseline" description="description1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

          <ExternalId source="UPC" description="description2">ExternalId1</ExternalId>

          <ExternalId source="DigitalPartner" description="description3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

        </ExternalIds>

        <BoxArtImages>

        </BoxArtImages>

        <RedboxReleaseDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

        <RedboxComingSoonDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

        <StreetReleaseDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      </ProductRef>

      <ProductRef productId="productId2" format="format2" websiteUrl="http://uri2">

        <Title>Title1</Title>

        <ExternalIds>

          <ExternalId source="Baseline" description="description4">ExternalId2</ExternalId>

          <ExternalId source="UPC" description="description5" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

          <ExternalId source="DigitalPartner" description="description6">ExternalId3</ExternalId>

        </ExternalIds>

        <BoxArtImages>

        </BoxArtImages>

        <RedboxReleaseDate>1900-01-01</RedboxReleaseDate>

        <RedboxComingSoonDate>1900-01-01</RedboxComingSoonDate>

        <StreetReleaseDate>1900-01-01</StreetReleaseDate>

      </ProductRef>

      <ProductRef productId="productId3" format="format3" websiteUrl="http://uri3">

        <Title>Title1</Title>

        <ExternalIds>

          <ExternalId source="Baseline" description="description7" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

          <ExternalId source="UPC" description="description8">ExternalId4</ExternalId>

          <ExternalId source="DigitalPartner" description="description9" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

        </ExternalIds>

        <BoxArtImages>

        </BoxArtImages>

        <RedboxReleaseDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

        <RedboxComingSoonDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

        <StreetReleaseDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      </ProductRef>

    </ProductRefs>

    <Extensions />

  </Title>

  <Title>

    <TitleIdentifier source="Baseline" description="description2">TitleIdentifier2</TitleIdentifier>

    <Title>Title2</Title>

    <SortTitle>SortTitle2</SortTitle>

    <ReleaseYear>0</ReleaseYear>

    <Genres>

      <Genre>Genre4</Genre>

      <Genre>Genre5</Genre>

      <Genre>Genre6</Genre>

    </Genres>

    <DomesticHomeVideoDistributor>DomesticHomeVideoDistributor2</DomesticHomeVideoDistributor>

    <DomesticTheatricalDistributor>DomesticTheatricalDistributor2</DomesticTheatricalDistributor>

    <SynopsisShort>SynopsisShort2</SynopsisShort>

    <SynopsisLong>SynopsisLong2</SynopsisLong>

    <RunningLength>P396DT1H1M1S</RunningLength>

    <MPAARating>G</MPAARating>

    <Actors>

      <Person>Person7</Person>

      <Person>Person8</Person>

      <Person>Person9</Person>

    </Actors>

    <Directors>

      <Person>Person10</Person>

      <Person>Person11</Person>

      <Person>Person12</Person>

    </Directors>

    <ProductRefs>

      <ProductRef productId="productId4" format="format4" websiteUrl="http://uri4">

        <Title>Title1</Title>

        <ExternalIds>

          <ExternalId source="Baseline" description="description10">ExternalId5</ExternalId>

          <ExternalId source="UPC" description="description11" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

          <ExternalId source="DigitalPartner" description="description12">ExternalId6</ExternalId>

        </ExternalIds>

        <BoxArtImages>

        </BoxArtImages>

        <RedboxReleaseDate>0001-01-01</RedboxReleaseDate>

        <StreetReleaseDate>0001-01-01</StreetReleaseDate>

      </ProductRef>

      <ProductRef productId="productId5" format="format5" websiteUrl="http://uri5">

        <Title>Title1</Title>

        <ExternalIds>

          <ExternalId source="Baseline" description="description13" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

          <ExternalId source="UPC" description="description14">ExternalId7</ExternalId>

          <ExternalId source="DigitalPartner" description="description15" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

        </ExternalIds>

        <BoxArtImages>

        </BoxArtImages>

        <RedboxReleaseDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

        <RedboxComingSoonDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

        <StreetReleaseDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      </ProductRef>

      <ProductRef productId="productId6" format="format6" websiteUrl="http://uri6">

        <Title>Title1</Title>

        <ExternalIds>

          <ExternalId source="Baseline" description="description16">ExternalId8</ExternalId>

          <ExternalId source="UPC" description="description17" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

          <ExternalId source="DigitalPartner" description="description18">ExternalId9</ExternalId>

        </ExternalIds>

        <BoxArtImages>

        </BoxArtImages>

        <RedboxReleaseDate>9999-12-31</RedboxReleaseDate>

        <RedboxComingSoonDate>9999-12-31</RedboxComingSoonDate>

        <StreetReleaseDate>9999-12-31</StreetReleaseDate>

      </ProductRef>

    </ProductRefs>

    <Extensions />

  </Title>

  <Title>

    <TitleIdentifier source="Baseline" description="description3">TitleIdentifier3</TitleIdentifier>

    <Title>Title3</Title>

    <SortTitle>SortTitle3</SortTitle>

    <ReleaseYear xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    <Genres>

      <Genre>Genre7</Genre>

      <Genre>Genre8</Genre>

      <Genre>Genre9</Genre>

    </Genres>

    <DomesticHomeVideoDistributor>DomesticHomeVideoDistributor3</DomesticHomeVideoDistributor>

    <DomesticTheatricalDistributor>DomesticTheatricalDistributor3</DomesticTheatricalDistributor>

    <SynopsisShort>SynopsisShort3</SynopsisShort>

    <SynopsisLong>SynopsisLong3</SynopsisLong>

    <RunningLength xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    <MPAARating xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    <Actors>

      <Person>Person13</Person>

      <Person>Person14</Person>

      <Person>Person15</Person>

    </Actors>

    <Directors>

      <Person>Person16</Person>

      <Person>Person17</Person>

      <Person>Person18</Person>

    </Directors>

    <ProductRefs>

      <ProductRef productId="productId7" format="format7" websiteUrl="http://uri7">

        <Title>Title1</Title>

        <ExternalIds>

          <ExternalId source="Baseline" description="description19" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

          <ExternalId source="UPC" description="description20">ExternalId10</ExternalId>

          <ExternalId source="DigitalPartner" description="description21" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

        </ExternalIds>

        <BoxArtImages>

        </BoxArtImages>

        <RedboxReleaseDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

        <RedboxComingSoonDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

        <StreetReleaseDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      </ProductRef>

      <ProductRef productId="productId8" format="format8" websiteUrl="http://uri8">

        <Title>Title1</Title>

        <ExternalIds>

          <ExternalId source="Baseline" description="description22">ExternalId11</ExternalId>

          <ExternalId source="UPC" description="description23" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

          <ExternalId source="DigitalPartner" description="description24">ExternalId12</ExternalId>

        </ExternalIds>

        <BoxArtImages>

        </BoxArtImages>

        <RedboxReleaseDate>1899-11-30</RedboxReleaseDate>

        <RedboxComingSoonDate>1899-11-30</RedboxComingSoonDate>

        <StreetReleaseDate>1899-11-30</StreetReleaseDate>

      </ProductRef>

      <ProductRef productId="productId9" format="format9" websiteUrl="http://uri9">

        <Title>Title1</Title>

        <ExternalIds>

          <ExternalId source="Baseline" description="description25" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

          <ExternalId source="UPC" description="description26">ExternalId13</ExternalId>

          <ExternalId source="DigitalPartner" description="description27" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

        </ExternalIds>

        <BoxArtImages>

        </BoxArtImages>

        <RedboxReleaseDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

        <RedboxComingSoonDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

        <StreetReleaseDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      </ProductRef>

    </ProductRefs>

    <Extensions />

  </Title>

</Titles>

Schema:

<?xml version="1.0" encoding="utf-8"?>

<xs:schema xmlns:tns="http://api.redbox.com/Products/TitleList/v2"

           elementFormDefault="qualified"

           targetNamespace="http://api.redbox.com/Products/TitleList/v2"

           xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:import namespace="http://api.redbox.com/OpenAPI/v1" />

  <xs:import namespace="http://www.w3.org/2005/Atom" />

  <xs:element name="Titles" type="tns:ArrayOfTitle" />

  <xs:complexType name="ArrayOfTitle">

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="Title" nillable="false" type="tns:Title" />

    </xs:sequence>

    <xs:attribute name="lastUpdated" type="xs:dateTime" use="required" />

  </xs:complexType>

  <xs:complexType name="Title">

    <xs:sequence>

      <xs:element minOccurs="1" maxOccurs="1" name="TitleIdentifier" type="tns:ExternalId" />

      <xs:element minOccurs="0" maxOccurs="1" name="Title" type="xs:string" />

      <xs:element minOccurs="0" maxOccurs="1" name="SortTitle" type="xs:string" />

      <xs:element minOccurs="1" maxOccurs="1" name="ReleaseYear" type="xs:unsignedShort" nillable="true"/>

      <xs:element minOccurs="0" maxOccurs="1" name="Genres" type="tns:ArrayOfGenre" />

      <xs:element minOccurs="0" maxOccurs="1" name="DomesticHomeVideoDistributor" type="xs:string" />

      <xs:element minOccurs="0" maxOccurs="1" name="DomesticTheatricalDistributor" type="xs:string" />

      <xs:element minOccurs="0" maxOccurs="1" name="SynopsisShort" type="xs:string" />

      <xs:element minOccurs="0" maxOccurs="1" name="SynopsisLong" type="xs:string" />

      <xs:element minOccurs="1" maxOccurs="1" name="RunningLength" type="xs:duration" nillable="true"/>

      <xs:element minOccurs="1" maxOccurs="1" name="MPAARating" type="tns:MPAARatings" nillable="true"/>

      <xs:element minOccurs="0" maxOccurs="1" name="Actors" type="tns:ArrayOfPerson" />

      <xs:element minOccurs="0" maxOccurs="1" name="Directors" type="tns:ArrayOfPerson" />

      <xs:element minOccurs="1" maxOccurs="1" name="ProductRefs" type="tns:ArrayOfProductRef" />

      <xs:element minOccurs="0" maxOccurs="1" name="Extensions" xmlns:oapi="http://api.redbox.com/OpenAPI/v1" type="oapi:ExtensionData" />

    </xs:sequence>

  </xs:complexType>

  <xs:complexType name="ArrayOfProductRef">

    <xs:sequence>

      <xs:element minOccurs="1" maxOccurs="unbounded" name="ProductRef" type="tns:ProductRef" />

    </xs:sequence>

  </xs:complexType>

  <xs:complexType name="ProductRef">

    <xs:sequence >

      <xs:element minOccurs="1" maxOccurs="1" name="Title" type="xs:string" />

      <xs:element minOccurs="0" maxOccurs="1" name="ExternalIds" type="tns:ArrayOfExternalId"  />

      <xs:element minOccurs="0" maxOccurs="1" name="BoxArtImages" type="tns:BoxArtLinks" />

      <xs:element minOccurs="1" maxOccurs="1" name="RedboxReleaseDate" type="xs:date" nillable="true"/>

      <xs:element minOccurs="1" maxOccurs="1" name="RedboxComingSoonDate" type="xs:date" nillable="true"/>

      <xs:element minOccurs="1" maxOccurs="1" name="StreetReleaseDate" type="xs:date" nillable="true"/>

    </xs:sequence>

    <xs:attribute name="productId" type="xs:string" />

    <xs:attribute name="format" type="xs:string" />

    <xs:attribute name="websiteUrl" type="xs:anyURI" />

  </xs:complexType>

  <xs:complexType name="ArrayOfGenre">

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="Genre" type="tns:Genre" />

    </xs:sequence>

  </xs:complexType>

  <xs:complexType name="Genre">

    <xs:simpleContent>

      <xs:extension base="xs:string" />

    </xs:simpleContent>

  </xs:complexType>

  <xs:simpleType name="ExternalIdProviders">

    <xs:restriction base="xs:string">

      <xs:enumeration value="Baseline" />

      <xs:enumeration value="UPC" />

      <xs:enumeration value="DigitalPartner" />

    </xs:restriction>

  </xs:simpleType>

  <xs:simpleType name="MPAARatings">

    <xs:restriction base="xs:string">

      <xs:enumeration value="G" />

      <xs:enumeration value="PG" />

      <xs:enumeration value="PG-13" />

      <xs:enumeration value="R" />

      <xs:enumeration value="NR" />

    </xs:restriction>

  </xs:simpleType>

  <xs:complexType name="ArrayOfPerson">

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="Person" nillable="false" type="tns:Person" />

    </xs:sequence>

  </xs:complexType>

  <xs:complexType name="Person">

    <xs:simpleContent>

      <xs:extension base="xs:string" />

    </xs:simpleContent>

  </xs:complexType>

  <xs:complexType name="ExternalId">

    <xs:simpleContent>

      <xs:extension base="xs:string">

        <xs:attribute name="source" type="tns:ExternalIdProviders" use="required" />

        <xs:attribute name="description" type="xs:string" />

      </xs:extension>

    </xs:simpleContent>

  </xs:complexType>

  <xs:complexType name="ArrayOfExternalId">

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="ExternalId" nillable="true" type="tns:ExternalId" />

    </xs:sequence>

  </xs:complexType>

  <xs:complexType name="BoxArtLinks" >

    <xs:sequence>

      <xs:element  minOccurs="0" maxOccurs="unbounded" ref="atom:link" xmlns:atom="http://www.w3.org/2005/Atom" />

    </xs:sequence>

  </xs:complexType>

</xs:schema>

### RecommendedProductIDList

Example:

<RecommendedProductIDList xmlns="http://api.redbox.com/Products/v2">

  <ProductId>ProductId1</ProductId>

  <ProductId>ProductId2</ProductId>

  <ProductId>ProductId3</ProductId>

</RecommendedProductIDList>

Schema:

<xs:schema id="RecommendedProductIDList"

    targetNamespace="http://api.redbox.com/Products/v2"

    elementFormDefault="qualified"

    xmlns="http://api.redbox.com/Products/v2"

    xmlns:tns="http://api.redbox.com/Products/v2"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:complexType name="RecommendedProductIDList" >

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="ProductId" type="xs:string" />

    </xs:sequence>

  </xs:complexType>

  <xs:element name="RecommendedProductIDList" type="RecommendedProductIDList" />

</xs:schema>

### Source Stores Schema

In this Source schema, the storeId attribute is the internal Redbox kioskId.

Reference for http://localhost:56745/SourceAPIStores/Stores

**Url:** http://localhost:56745/SourceAPIStores/Stores

**HTTP Method:** GET

|  |  |  |
| --- | --- | --- |
| **Message direction** | **Format** | **Body** |
| Request | N/A | The Request body is empty. |
| Response | Xml | [Example](http://localhost:56745/SourceAPIStores/help/operations/GetAllStores#response-xml),[Schema](http://localhost:56745/SourceAPIStores/help/operations/GetAllStores#response-schema) |

The following is an example response Xml body:

<Stores lastUpdated="1999-05-31T11:20:00" xmlns="http://api-source.redbox.com/Stores/v2">

<Store storeId="String content" kioskType="Indoor" kioskStatus="Online">

<Location lat="12678967.543233" long="12678967.543233">

<Address>String content</Address>

<Address2>String content</Address2>

<City>String content</City>

<State>String content</State>

<Zipcode>String content</Zipcode>

</Location>

<KioskLabel>String content</KioskLabel>

<KioskBanner>String content</KioskBanner>

<Channel>String content</Channel>

</Store>

<Store storeId="String content" kioskType="Indoor" kioskStatus="Online">

<Location lat="12678967.543233" long="12678967.543233">

<Address>String content</Address>

<Address2>String content</Address2>

<City>String content</City>

<State>String content</State>

<Zipcode>String content</Zipcode>

</Location>

<KioskLabel>String content</KioskLabel>

<KioskBanner>String content</KioskBanner>

<Channel>String content</Channel>

</Store>

</Stores>

The following is the response Xml Schema:

<xs:schema xmlns:tns="http://api-source.redbox.com/Stores/v2" elementFormDefault="qualified" targetNamespace="http://api-source.redbox.com/Stores/v2" xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="Stores" nillable="true" type="tns:StoreList" />

<xs:complexType name="StoreList">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="unbounded" name="Store" type="tns:Store" />

</xs:sequence>

<xs:attribute name="lastUpdated" type="xs:dateTime" use="required" />

</xs:complexType>

<xs:complexType name="Store">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="1" name="Location" type="tns:Location" />

<xs:element minOccurs="0" maxOccurs="1" name="KioskLabel" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="KioskBanner" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="Channel" type="xs:string" />

</xs:sequence>

<xs:attribute name="storeId" type="xs:string" />

<xs:attribute name="kioskType" type="tns:KioskTypeEnum" use="required" />

<xs:attribute name="kioskStatus" type="tns:CommStatusEnum" use="required" />

</xs:complexType>

<xs:complexType name="Location">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="1" name="Address" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="Address2" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="City" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="State" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="Zipcode" type="xs:string" />

</xs:sequence>

<xs:attribute name="lat" type="xs:decimal" use="required" />

<xs:attribute name="long" type="xs:decimal" use="required" />

</xs:complexType>

<xs:simpleType name="KioskTypeEnum">

<xs:restriction base="xs:string">

<xs:enumeration value="Indoor" />

<xs:enumeration value="Outdoor" />

</xs:restriction>

</xs:simpleType>

<xs:simpleType name="CommStatusEnum">

<xs:restriction base="xs:string">

<xs:enumeration value="Online" />

<xs:enumeration value="Offline" />

</xs:restriction>

</xs:simpleType>

</xs:schema>

### External Stores Schema

In this External Stores schema, storeId is the external Store GUID assigned to the kiosk (typed as a string).

Reference for http://localhost:56745/RedboxAPIStores/Stores

**Url:** http://localhost:56745/RedboxAPIStores/Stores

**HTTP Method:** GET

|  |  |  |
| --- | --- | --- |
| **Message direction** | **Format** | **Body** |
| Request | N/A | The Request body is empty. |
| Response | Xml | [Example](http://localhost:56745/redboxapistores/help/operations/GetAllStores#response-xml),[Schema](http://localhost:56745/redboxapistores/help/operations/GetAllStores#response-schema) |

The following is an example response Xml body:

<StoresBulkList lastUpdated="1900-01-01T01:01:01" xmlns="http://api.redbox.com/Stores/v2">

  <Store storeId="storeId1" storeType="Indoor" commStatus="Online">

    <Location lat="1" long="1">

      <Address>Address1</Address>

      <Address2>Address21</Address2>

      <City>City1</City>

      <State>State1</State>

      <Zipcode>Zipcode1</Zipcode>

    </Location>

    <Label>Label1</Label>

    <Retailer>Retailer1</Retailer>

    <Channel>Channel1</Channel>

    <DistanceFromSearchLocation>1</DistanceFromSearchLocation>

  </Store>

  <Store storeId="storeId2" storeType="Outdoor" commStatus="Offline">

    <Location lat="-79228162514264337593543950335" long="-79228162514264337593543950335">

      <Address>Address2</Address>

      <Address2>Address22</Address2>

      <City>City2</City>

      <State>State2</State>

      <Zipcode>Zipcode2</Zipcode>

    </Location>

    <Label>Label2</Label>

    <Retailer>Retailer2</Retailer>

    <Channel>Channel2</Channel>

    <DistanceFromSearchLocation>-79228162514264337593543950335</DistanceFromSearchLocation>

  </Store>

  <Store storeId="storeId3" storeType="Indoor" commStatus="Online">

    <Location lat="79228162514264337593543950335" long="79228162514264337593543950335">

      <Address>Address3</Address>

      <Address2>Address23</Address2>

      <City>City3</City>

      <State>State3</State>

      <Zipcode>Zipcode3</Zipcode>

    </Location>

    <Label>Label3</Label>

    <Retailer>Retailer3</Retailer>

    <Channel>Channel3</Channel>

    <DistanceFromSearchLocation>79228162514264337593543950335</DistanceFromSearchLocation>

  </Store>

</StoresBulkList>

The following is the response Xml Schema:

<xs:schema xmlns:tns="http://api.redbox.com/Stores/v2" elementFormDefault="qualified"

           targetNamespace="http://api.redbox.com/Stores/v2"

           xmlns:xs="http://www.w3.org/2001/XMLSchema"

           xmlns:oapi="http://api.redbox.com/OpenAPI/v1">

  <xs:import namespace="http://api.redbox.com/OpenAPI/v1" />

  <xs:include schemaLocation="store.xsd" />

  <xs:element name="StoreBulkList" nillable="true" type="tns:StoreBulkList" />

  <xs:complexType name="StoreBulkList">

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="Store" type="tns:Store" />

    </xs:sequence>

    <xs:attribute name="lastUpdated" type="xs:dateTime" use="required" />

  </xs:complexType>

</xs:schema>

### Common External Store Schema

The following common Store schema is shared by several of the External Store Schemas:

<xs:schema xmlns:tns="http://api.redbox.com/Stores/v2" elementFormDefault="qualified"

           targetNamespace="http://api.redbox.com/Stores/v2"

           xmlns:xs="http://www.w3.org/2001/XMLSchema"

           xmlns:oapi="http://api.redbox.com/OpenAPI/v1">

  <xs:import namespace="http://api.redbox.com/OpenAPI/v1" />

  <xs:complexType name="Store">

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="1" name="Location" type="tns:Location" />

      <xs:element minOccurs="0" maxOccurs="1" name="Label" type="xs:string" />

      <xs:element minOccurs="0" maxOccurs="1" name="Retailer" type="xs:string" />

      <xs:element minOccurs="0" maxOccurs="1" name="Channel" type="xs:string" />

      <xs:element minOccurs="0" maxOccurs="1" name="DistanceFromSearchLocation" type="xs:decimal" />

    </xs:sequence>

    <xs:attribute name="storeId" type="xs:string" />

    <xs:attribute name="storeType" type="tns:StoreTypeEnum" use="required" />

    <xs:attribute name="commStatus" type="tns:CommStatusEnum" use="required" />

  </xs:complexType>

  <xs:complexType name="Location">

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="1" name="Address" type="xs:string" />

      <xs:element minOccurs="0" maxOccurs="1" name="Address2" type="xs:string" />

      <xs:element minOccurs="0" maxOccurs="1" name="City" type="xs:string" />

      <xs:element minOccurs="0" maxOccurs="1" name="State" type="xs:string" />

      <xs:element minOccurs="0" maxOccurs="1" name="Zipcode" type="xs:string" />

    </xs:sequence>

    <xs:attribute name="lat" type="xs:decimal" use="required" />

    <xs:attribute name="long" type="xs:decimal" use="required" />

  </xs:complexType>

  <xs:simpleType name="StoreTypeEnum">

    <xs:restriction base="xs:string">

      <xs:enumeration value="Indoor" />

      <xs:enumeration value="Outdoor" />

    </xs:restriction>

  </xs:simpleType>

  <xs:simpleType name="CommStatusEnum">

    <xs:restriction base="xs:string">

      <xs:enumeration value="Online" />

      <xs:enumeration value="Offline" />

    </xs:restriction>

  </xs:simpleType>

</xs:schema>

### Source Inventory List

Reference for http://localhost:56745/SourceAPIInventory/Inventory

**Url:** http://localhost:56745/SourceAPIInventory/Inventory

**HTTP Method:** GET

|  |  |  |
| --- | --- | --- |
| **Message direction** | **Format** | **Body** |
| Request | N/A | The Request body is empty. |
| Response | Xml | [Example](http://localhost:56745/SourceAPIInventory/help/operations/GetAllInventory#response-xml),[Schema](http://localhost:56745/SourceAPIInventory/help/operations/GetAllInventory#response-schema) |

The following is an example response Xml body:

<Inventory xmlns="http://api-source.redbox.com/Inventory/v2">

<StoreInventory storeId="1627aea5-8e0a-4371-9022-9b504344e724" >

<ProductInventory productId="1627aea5-8e0a-4371-9022-9b504344e724" lastUpdated="1999-05-31T11:20:00" inventoryStatus="InStock" />

<ProductInventory productId="1627aea5-8e0a-4371-9022-9b504344e724" lastUpdated="1999-05-31T11:20:00" inventoryStatus="InStock" />

</StoreInventory>

<StoreInventory storeId="1627aea5-8e0a-4371-9022-9b504344e724" >

<ProductInventory productId="1627aea5-8e0a-4371-9022-9b504344e724" lastUpdated="1999-05-31T11:20:00" inventoryStatus="InStock" />

<ProductInventory productId="1627aea5-8e0a-4371-9022-9b504344e724" lastUpdated="1999-05-31T11:20:00" inventoryStatus="InStock" />

</StoreInventory>

</Inventory>

The following is the response Xml Schema:

<xs:schema xmlns:tns="http://api-source.redbox.com/Inventory/v2" elementFormDefault="qualified" targetNamespace="http://api-source.redbox.com/Inventory/v1" xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="Inventory" nillable="true" type="tns:ArrayOfStoreInventory" />

<xs:complexType name="ArrayOfStoreInventory">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="unbounded" name="StoreInventory" nillable="true" type="tns:StoreInventory" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="StoreInventory">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="unbounded" name="ProductInventory" type="tns:ProductInventory" />

</xs:sequence>

<xs:attribute name="storeId" type="string" use="required" />

</xs:complexType>

<xs:complexType name="ProductInventory">

<xs:attribute name="productId" type="string" use="required" />

<xs:attribute name="lastUpdated" type="xs:dateTime" use="required" />

<xs:attribute name="inventoryStatus" type="tns:InventoryStatusEnum" use="required" />

</xs:complexType>

<xs:simpleType name="InventoryStatusEnum">

<xs:restriction base="xs:string">

<xs:enumeration value="InStock" />

<xs:enumeration value="OutOfStock" />

<xs:enumeration value="LowStock" />

</xs:restriction>

</xs:simpleType>

</xs:schema>

### External Store Lookup Response Schema

This schema is returned from two operations:

* S2 – Paginated Store Lookup
* S3 – Paginated Store Location Search

The <DistanceFromSearchLocation> element is provided only in response to the S3 operation (and is omitted from the response for the S2 operation).

Reference for http://localhost:56745/RedboxAPIStores/StoreInventory/LatLong/{LAT},{LONGT}?apiKey={AKEY}&radius={R}&count={C}&banner={B}&pageNum={PN}&pageSize={PS}

**Url:** http://localhost:56745/RedboxAPIStores/StoreInventory/LatLong/{LAT},{LONGT}?apiKey={AKEY}&radius={R}&count={C}&banner={B}&pageNum={PN}&pageSize={PS}

**HTTP Method:** GET

|  |  |  |
| --- | --- | --- |
| **Message direction** | **Format** | **Body** |
| Request | N/A | The Request body is empty. |
| Response | Xml | [Example](http://localhost:56745/redboxapistores/help/operations/FindStoresByLatLong#response-xml),[Schema](http://localhost:56745/redboxapistores/help/operations/FindStoresByLatLong#response-schema) |

The following is an example response Xml body:

<StoresLookupList lastUpdated="1900-01-01T01:01:01" xmlns="http://api.redbox.com/Stores/v2">

  <Store storeId="storeId1" storeType="Indoor" commStatus="Online">

    <Location lat="1" long="1">

      <Address>Address1</Address>

      <Address2>Address21</Address2>

      <City>City1</City>

      <State>State1</State>

      <Zipcode>Zipcode1</Zipcode>

    </Location>

    <Label>Label1</Label>

    <Retailer>Retailer1</Retailer>

    <Channel>Channel1</Channel>

    <DistanceFromSearchLocation>1</DistanceFromSearchLocation>

  </Store>

  <Store storeId="storeId2" storeType="Outdoor" commStatus="Offline">

    <Location lat="-79228162514264337593543950335" long="-79228162514264337593543950335">

      <Address>Address2</Address>

      <Address2>Address22</Address2>

      <City>City2</City>

      <State>State2</State>

      <Zipcode>Zipcode2</Zipcode>

    </Location>

    <Label>Label2</Label>

    <Retailer>Retailer2</Retailer>

    <Channel>Channel2</Channel>

    <DistanceFromSearchLocation>-79228162514264337593543950335</DistanceFromSearchLocation>

  </Store>

  <Store storeId="storeId3" storeType="Indoor" commStatus="Online">

    <Location lat="79228162514264337593543950335" long="79228162514264337593543950335">

      <Address>Address3</Address>

      <Address2>Address23</Address2>

      <City>City3</City>

      <State>State3</State>

      <Zipcode>Zipcode3</Zipcode>

    </Location>

    <Label>Label3</Label>

    <Retailer>Retailer3</Retailer>

    <Channel>Channel3</Channel>

    <DistanceFromSearchLocation>79228162514264337593543950335</DistanceFromSearchLocation>

  </Store>

  <Paging pageNum="1" pageSize="1" totalNumItems="1" pageCount="1" xmlns:atom="http://www.w3.org/2005/Atom">

<atom:link rel="first" href="http://uri" />

<atom:link rel="last" href="http://uri" />

<atom:link rel="prev" href="http://uri" />

<atom:link rel="next" href="http://uri" />

</Paging>   
</StoreLookupList>

The following is the response Xml Schema. It re-uses the Store schema defined in Section 8.1.9.

<xs:schema xmlns:tns="http://api.redbox.com/Stores/v2" elementFormDefault="qualified"

           targetNamespace="http://api.redbox.com/Stores/v2"

           xmlns:xs="http://www.w3.org/2001/XMLSchema"

           xmlns:oapi="http://api.redbox.com/OpenAPI/v1">

  <xs:import namespace="http://api.redbox.com/OpenAPI/v1" />

  <xs:include schemaLocation="store.xsd" />

  <xs:element name="StoreLookupList" nillable="true" type="tns:StoreLookupList" />

  <xs:complexType name="StoreLookupList">

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="Store" type="tns:Store" />

      <xs:element minOccurs="0" maxOccurs="1" name="Paging" type="oapi:Paging" />

    </xs:sequence>

    <xs:attribute name="lastUpdated" type="xs:dateTime" use="required" />

  </xs:complexType>

</xs:schema>

### InventoryLookup Schema

Reference for http://localhost:56745/RedboxAPIInventory/Inventory?apiKey={AKEY}&storeIds={SIDS}&inventoryProducts={IP}&pageNum={PN}&pageSize={PS}

**Url:** http://localhost:56745/RedboxAPIInventory/Inventory?apiKey={AKEY}&storeIds={SIDS}&inventoryProducts={IP}&pageNum={PN}&pageSize={PS}

**HTTP Method:** GET

|  |  |  |
| --- | --- | --- |
| **Message direction** | **Format** | **Body** |
| Request | N/A | The Request body is empty. |
| Response | Xml | [Example](http://localhost:56745/redboxapiinventory/help/operations/GetInventoryForStores#response-xml),[Schema](http://localhost:56745/redboxapiinventory/help/operations/GetInventoryForStores#response-schema) |

The following is an example response Xml body:

<Inventory xmlns="http://api.redbox.com/Inventory/v2">  
  <StoreInventory storeId="storeId1" lastUpdated="1900-01-01T01:01:01">  
    <ProductInventory productId="productId1" inventoryStatus="InStock" />  
    <ProductInventory productId="productId2" inventoryStatus="OutOfStock" />  
    <ProductInventory productId="productId3" inventoryStatus="LowStock" />  
  </StoreInventory>  
  <StoreInventory storeId="storeId2" lastUpdated="0001-01-01T00:00:00"></StoreInventory>  
  <StoreInventory storeId="storeId3" lastUpdated="9999-12-31T23:59:59.9999999">  
    <ProductInventory productId="productId7" inventoryStatus="InStock" />  
    <ProductInventory productId="productId8" inventoryStatus="OutOfStock" />  
    <ProductInventory productId="productId9" inventoryStatus="LowStock" />  
  </StoreInventory>

  <Paging pageNum="1" pageSize="1" totalNumItems="1" pageCount="1" xmlns:atom="http://www.w3.org/2005/Atom">

<atom:link rel="first" href="http://uri" />

<atom:link rel="last" href="http://uri" />

<atom:link rel="prev" href="http://uri" />

<atom:link rel="next" href="http://uri" />

</Paging>  
</Inventory>

The following is the response Xml Schema:

<xs:schema xmlns:tns="http://api.redbox.com/Inventory/v2"  
           elementFormDefault="qualified"  
           targetNamespace="http://api.redbox.com/Inventory/v2"  
           xmlns:xs="http://www.w3.org/2001/XMLSchema"  
           xmlns:oapi="http://api.redbox.com/OpenAPI/v1">  
  <xs:import namespace="http://api.redbox.com/OpenAPI/v1" />  
  <xs:element name="Inventory" nillable="true" type="tns:InventoryLookupList" />  
  <xs:complexType name="InventoryLookupList">  
    <xs:sequence>  
      <xs:element minOccurs="0" maxOccurs="unbounded" name="StoreInventory" type="tns:ArrayOfProductInventory" />  
      <xs:element minOccurs="0" maxOccurs="1" name="Paging" type="oapi:Paging" />  
    </xs:sequence>  
  </xs:complexType>  
  <xs:complexType name="ArrayOfProductInventory">  
    <xs:sequence>  
      <xs:element minOccurs="0" maxOccurs="unbounded" name="ProductInventory" type="tns:ProductInventory" />  
    </xs:sequence>  
    <xs:attribute name="storeId" type="xs:string" use="required" />  
    <xs:attribute name="lastUpdated" type="xs:dateTime" use="required" />  
  </xs:complexType>  
  <xs:complexType name="ProductInventory">  
    <xs:attribute name="productId" type="xs:string" use="required"/>  
    <xs:attribute name="inventoryStatus" type="tns:InventoryStatusEnum" use="required" />  
  </xs:complexType>  
  <xs:simpleType name="InventoryStatusEnum">  
    <xs:restriction base="xs:string">  
      <xs:enumeration value="InStock" />  
      <xs:enumeration value="OutOfStock" />  
      <xs:enumeration value="LowStock" />  
    </xs:restriction>  
  </xs:simpleType>  
</xs:schema>

### Source Top20 Schema

Reference for http://localhost:56745/SourceAPIProducts/Top20/Movies?p={PERIOD}

**Url:** http://localhost:56745/SourceAPIProducts/Top20/Movies?p={PERIOD}

**HTTP Method:** GET

|  |  |  |
| --- | --- | --- |
| **Message direction** | **Format** | **Body** |
| Request | N/A | The Request body is empty. |
| Response | Xml | [Example](http://localhost:56745/sourceAPIProducts/help/operations/GetTopTwentyMovies#response-xml),[Schema](http://localhost:56745/sourceAPIProducts/help/operations/GetTopTwentyMovies#response-schema) |

The following is an example response Xml body:

<Top20 period="65535" productType="String content" lastUpdated="1999-05-31T11:20:00" xmlns="http://api-source.redbox.com/Products/v2">

<Item productId="1627aea5-8e0a-4371-9022-9b504344e724" position="32767" >

<NewEntry />

</Item>

<Item productId="1627aea5-8e0a-4371-9022-9b504344e724" position="32767" >

<PositionChange direction=”Up” amount=”2” />

</Item>

</Top20>

The following is the response Xml Schema:

<xs:schema xmlns:tns="http://api-source.redbox.com/Products/v2" elementFormDefault="qualified" targetNamespace="http://api-source.redbox.com/Products/v2" xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="Top20" nillable="true" type="tns:Top20" />

<xs:complexType name="Top20">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="unbounded" name="Item" type="tns:Top20Entry" />

</xs:sequence>

<xs:attribute name="period" type="xs:unsignedShort" use="required" />

<xs:attribute name="productType" type="xs:string" />

<xs:attribute name="lastUpdated" type="xs:dateTime" use="required" />

</xs:complexType>

<xs:complexType name="Top20Entry">

<xs:sequence>

<xs:choice minOccurs="1" maxOccurs="1">

<xs:element minOccurs="0" maxOccurs="1" name="PositionChange" type="tns:PositionChange" />

<xs:element minOccurs="0" maxOccurs="1" name="NewEntry" type="tns:NewEntry" />

</xs:choice>

</xs:sequence>

<xs:attribute name="productId" type="string" use="required" />

<xs:attribute name="position" type="xs:short" use="required" />

</xs:complexType>

<xs:complexType name="PositionChange">

<xs:attribute name="direction" type="tns:DirectionsEnum" use="required" />

<xs:attribute name="amount" type="xs:unsignedShort" use="required" />

</xs:complexType>

<xs:simpleType name="DirectionsEnum">

<xs:restriction base="xs:string">

<xs:enumeration value="Up" />

<xs:enumeration value="Down" />

<xs:enumeration value="NoChange" />

</xs:restriction>

</xs:simpleType>

<xs:complexType name="NewEntry" />

</xs:schema>

### Store and Inventory Search Result Schema

Example Result from operation I2:

<StoreAndInventorySearchResult xmlns="http://api.redbox.com/Inventory/v2">

  <StoreAndInventory>

    <Store storeId="storeId1" storeType="Indoor" commStatus="Online">

      <Location lat="1" long="1" xmlns="http://api.redbox.com/Stores/v2">

        <Address>Address1</Address>

        <Address2>Address21</Address2>

        <City>City1</City>

        <State>State1</State>

        <Zipcode>Zipcode1</Zipcode>

      </Location>

      <Label xmlns="http://api.redbox.com/Stores/v2">Label1</Label>

      <Retailer xmlns="http://api.redbox.com/Stores/v2">Retailer1</Retailer>

      <Channel xmlns="http://api.redbox.com/Stores/v2">Channel1</Channel>

      <DistanceFromSearchLocation xmlns="http://api.redbox.com/Stores/v2">1</DistanceFromSearchLocation>

    </Store>

    <StoreInventory storeId="storeId1" lastUpdated="1900-01-01T01:01:01">

      <ProductInventory productId="productId1" inventoryStatus="InStock" />

      <ProductInventory productId="productId2" inventoryStatus="OutOfStock" />

      <ProductInventory productId="productId3" inventoryStatus="LowStock" />

    </StoreInventory>

  </StoreAndInventory>

  <StoreAndInventory>

    <Store storeId="storeId2" storeType="Outdoor" commStatus="Offline">

      <Location lat="-79228162514264337593543950335" long="-79228162514264337593543950335" xmlns="http://api.redbox.com/Stores/v2">

        <Address>Address2</Address>

        <Address2>Address22</Address2>

        <City>City2</City>

        <State>State2</State>

        <Zipcode>Zipcode2</Zipcode>

      </Location>

      <Label xmlns="http://api.redbox.com/Stores/v2">Label2</Label>

      <Retailer xmlns="http://api.redbox.com/Stores/v2">Retailer2</Retailer>

      <Channel xmlns="http://api.redbox.com/Stores/v2">Channel2</Channel>

      <DistanceFromSearchLocation xmlns="http://api.redbox.com/Stores/v2">-79228162514264337593543950335</DistanceFromSearchLocation>

    </Store>

    <StoreInventory storeId="storeId2" lastUpdated="0001-01-01T00:00:00">

      <ProductInventory productId="productId4" inventoryStatus="InStock" />

      <ProductInventory productId="productId5" inventoryStatus="OutOfStock" />

      <ProductInventory productId="productId6" inventoryStatus="LowStock" />

    </StoreInventory>

  </StoreAndInventory>

  <StoreAndInventory>

    <Store storeId="storeId3" storeType="Indoor" commStatus="Online">

      <Location lat="79228162514264337593543950335" long="79228162514264337593543950335" xmlns="http://api.redbox.com/Stores/v2">

        <Address>Address3</Address>

        <Address2>Address23</Address2>

        <City>City3</City>

        <State>State3</State>

        <Zipcode>Zipcode3</Zipcode>

      </Location>

      <Label xmlns="http://api.redbox.com/Stores/v2">Label3</Label>

      <Retailer xmlns="http://api.redbox.com/Stores/v2">Retailer3</Retailer>

      <Channel xmlns="http://api.redbox.com/Stores/v2">Channel3</Channel>

      <DistanceFromSearchLocation xmlns="http://api.redbox.com/Stores/v2">79228162514264337593543950335</DistanceFromSearchLocation>

    </Store>

    <StoreInventory storeId="storeId3" lastUpdated="9999-12-31T23:59:59.9999999">

      <ProductInventory productId="productId7" inventoryStatus="InStock" />

      <ProductInventory productId="productId8" inventoryStatus="OutOfStock" />

      <ProductInventory productId="productId9" inventoryStatus="LowStock" />

    </StoreInventory>

  </StoreAndInventory>

  <Paging pageNum="1" pageSize="1" totalNumItems="1" pageCount="1">

<atom:link rel="first" href="http://uri" />

<atom:link rel="last" href="http://uri" />

<atom:link rel="prev" href="http://uri" />

<atom:link rel="next" href="http://uri" />

  </Paging>

</StoreAndInventorySearchResult>

The following is the response Xml Schema:

<xs:schema xmlns:tns="http://api.redbox.com/Inventory/v2"

           elementFormDefault="qualified"

           targetNamespace="http://api.redbox.com/Inventory/v2"

           xmlns:xs="http://www.w3.org/2001/XMLSchema"

           xmlns:oapi="http://api.redbox.com/OpenAPI/v1">

  <xs:import namespace="http://api.redbox.com/OpenAPI/v1" />

  <xs:import namespace="http://api.redbox.com/Stores/v2" />

  <xs:element name="StoreAndInventorySearchResult" type="tns:StoreAndInventorySearchResultType" />

  <xs:complexType name="StoreAndInventorySearchResultType">

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="StoreAndInventory" type="tns:StoreAndInventoryResult" />

      <xs:element minOccurs="0" maxOccurs="1" name="Paging" type="oapi:Paging" />

    </xs:sequence>

  </xs:complexType>

  <xs:complexType name="StoreAndInventoryResult" >

    <xs:sequence >

      <xs:element name="Store" type="store:Store" xmlns:store="http://api.redbox.com/Stores/v2" />

      <xs:element name="StoreInventory" type="tns:ArrayOfProductInventory" />

    </xs:sequence>

  </xs:complexType>

  <xs:complexType name="ArrayOfProductInventory">

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="ProductInventory" type="tns:ProductInventory" />

    </xs:sequence>

    <xs:attribute name="storeId" type="xs:string" use="required" />

    <xs:attribute name="lastUpdated" type="xs:dateTime" use="required" />

  </xs:complexType>

  <xs:complexType name="ProductInventory">

    <xs:attribute name="productId" type="xs:string" use="required"/>

    <xs:attribute name="inventoryStatus" type="tns:InventoryStatusEnum" use="required" />

  </xs:complexType>

  <xs:simpleType name="InventoryStatusEnum">

    <xs:restriction base="xs:string">

      <xs:enumeration value="InStock" />

      <xs:enumeration value="OutOfStock" />

      <xs:enumeration value="LowStock" />

    </xs:restriction>

  </xs:simpleType>

</xs:schema>

### External Top20 Schema

Reference for http://localhost:56745/RedboxAPIProducts/Products/Movies/Top20?p={PERIOD}

**Url:** http://localhost:56745/RedboxAPIProducts/Products/Movies/Top20?p={PERIOD}

**HTTP Method:** GET

|  |  |  |
| --- | --- | --- |
| **Message direction** | **Format** | **Body** |
| Request | N/A | The Request body is empty. |
| Response | Xml | [Example](http://localhost:56745/redboxapiproducts/help/operations/GetTopTwentyMovies#response-xml),[Schema](http://localhost:56745/redboxapiproducts/help/operations/GetTopTwentyMovies#response-schema) |

The following is an example response Xml body:

<Top20 period="0" productType="productType1" lastUpdated="1900-01-01T01:01:01" xmlns="http://api.redbox.com/Products/v2">

  <Item productId="productId1" format="format1" websiteUrl="http://uri1">

    <Title xmlns="http://api.redbox.com/Products/MovieList/v2">Title1</Title>

    <SortTitle xmlns="http://api.redbox.com/Products/MovieList/v2">SortTitle1</SortTitle>

    <RedboxReleaseDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" xmlns="http://api.redbox.com/Products/MovieList/v2" />

    <StreetReleaseDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" xmlns="http://api.redbox.com/Products/MovieList/v2" />

    <ReleaseYear xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" xmlns="http://api.redbox.com/Products/MovieList/v2" />

    <Genres xmlns="http://api.redbox.com/Products/MovieList/v2">

      <Genre>Genre1</Genre>

      <Genre>Genre2</Genre>

      <Genre>Genre3</Genre>

    </Genres>

    <BoxArtImages xmlns="http://api.redbox.com/Products/MovieList/v2">

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Original” xmlns="http://www.w3.org/2005/Atom"></link>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Thumbnail” xmlns="http://www.w3.org/2005/Atom"></link>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Thumbnail(150)” xmlns="http://www.w3.org/2005/Atom"></link>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Full” xmlns="http://www.w3.org/2005/Atom"></link>

    </BoxArtImages>

    <Extensions xmlns="http://api.redbox.com/Products/MovieList/v2" />

    <ExternalIds xmlns="http://api.redbox.com/Products/MovieList/v2">

      <ExternalId source="Baseline" description="description1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      <ExternalId source="UPC" description="description2">ExternalId1</ExternalId>

      <ExternalId source="DigitalPartner" description="description3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    </ExternalIds>

    <DomesticHomeVideoDistributor xmlns="http://api.redbox.com/Products/MovieList/v2">DomesticHomeVideoDistributor1</DomesticHomeVideoDistributor>

    <DomesticTheatricalDistributor xmlns="http://api.redbox.com/Products/MovieList/v2">DomesticTheatricalDistributor1</DomesticTheatricalDistributor>

    <SynopsisShort xmlns="http://api.redbox.com/Products/MovieList/v2">SynopsisShort1</SynopsisShort>

    <SynopsisLong xmlns="http://api.redbox.com/Products/MovieList/v2">SynopsisLong1</SynopsisLong>

    <RunningLength xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" xmlns="http://api.redbox.com/Products/MovieList/v2" />

    <MPAARating xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" xmlns="http://api.redbox.com/Products/MovieList/v2" />

    <Actors xmlns="http://api.redbox.com/Products/MovieList/v2">

      <Person>Person1</Person>

      <Person>Person2</Person>

      <Person>Person3</Person>

    </Actors>

    <Directors xmlns="http://api.redbox.com/Products/MovieList/v2">

      <Person>Person4</Person>

      <Person>Person5</Person>

      <Person>Person6</Person>

    </Directors>

    <Top20Entry xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="p4:NewTop20Entry" position="0" xmlns:p4="http://api.redbox.com/Top20/v2" />

  </Item>

  <Item productId="productId2" format="format2" websiteUrl="http://uri2">

    <Title xmlns="http://api.redbox.com/Products/MovieList/v2">Title2</Title>

    <SortTitle xmlns="http://api.redbox.com/Products/MovieList/v2">SortTitle2</SortTitle>

    <RedboxReleaseDate xmlns="http://api.redbox.com/Products/MovieList/v2">1900-01-01</RedboxReleaseDate>

    <StreetReleaseDate xmlns="http://api.redbox.com/Products/MovieList/v2">1900-01-01</StreetReleaseDate>

    <ReleaseYear xmlns="http://api.redbox.com/Products/MovieList/v2">0</ReleaseYear>

    <Genres xmlns="http://api.redbox.com/Products/MovieList/v2">

      <Genre>Genre4</Genre>

      <Genre>Genre5</Genre>

      <Genre>Genre6</Genre>

    </Genres>

    <BoxArtImages xmlns="http://api.redbox.com/Products/MovieList/v2">

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Original” xmlns="http://www.w3.org/2005/Atom"></link>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Thumbnail” xmlns="http://www.w3.org/2005/Atom"></link>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Thumbnail(150)” xmlns="http://www.w3.org/2005/Atom"></link>

      <link href="http://uri1" rel="http://api.redbox.com/Links/BoxArt/Full” xmlns="http://www.w3.org/2005/Atom"></link>

    </BoxArtImages>

    <Extensions xmlns="http://api.redbox.com/Products/MovieList/v2" />

    <ExternalIds xmlns="http://api.redbox.com/Products/MovieList/v2">

      <ExternalId source="Baseline" description="description4">ExternalId2</ExternalId>

      <ExternalId source="UPC" description="description5" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

      <ExternalId source="DigitalPartner" description="description6">ExternalId3</ExternalId>

    </ExternalIds>

    <DomesticHomeVideoDistributor xmlns="http://api.redbox.com/Products/MovieList/v2">DomesticHomeVideoDistributor2</DomesticHomeVideoDistributor>

    <DomesticTheatricalDistributor xmlns="http://api.redbox.com/Products/MovieList/v2">DomesticTheatricalDistributor2</DomesticTheatricalDistributor>

    <SynopsisShort xmlns="http://api.redbox.com/Products/MovieList/v2">SynopsisShort2</SynopsisShort>

    <SynopsisLong xmlns="http://api.redbox.com/Products/MovieList/v2">SynopsisLong2</SynopsisLong>

    <RunningLength xmlns="http://api.redbox.com/Products/MovieList/v2">P396DT1H1M1S</RunningLength>

    <MPAARating xmlns="http://api.redbox.com/Products/MovieList/v2">G</MPAARating>

    <Actors xmlns="http://api.redbox.com/Products/MovieList/v2">

      <Person>Person7</Person>

      <Person>Person8</Person>

      <Person>Person9</Person>

    </Actors>

    <Directors xmlns="http://api.redbox.com/Products/MovieList/v2">

      <Person>Person10</Person>

      <Person>Person11</Person>

      <Person>Person12</Person>

    </Directors>

    <Top20Entry xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="p4:NewTop20Entry" position="65535" xmlns:p4="http://api.redbox.com/Top20/v2" />

  </Item>

</Top20>

The following is the response Xml Schema. This schema re-uses the **MovieList** schema defined in 8.1.3 External Movie List.

<xs:schema xmlns:tns="http://api.redbox.com/Products/v2"

           elementFormDefault="qualified"

           targetNamespace="http://api.redbox.com/Products/v2"

           xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:import namespace="http://api.redbox.com/OpenAPI/v1" />

  <xs:import namespace="http://api.redbox.com/Top20/v2" />

  <xs:import namespace="http://api.redbox.com/Products/MovieList/v2" />

  <xs:element name="Top20" nillable="true" type="tns:Top20" />

  <xs:complexType name="Top20">

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="Item" type="tns:Top20MovieProduct" />

    </xs:sequence>

    <xs:attribute name="period" type="xs:unsignedShort" use="required" />

    <xs:attribute name="productType" type="xs:string" />

    <xs:attribute name="lastUpdated" type="xs:dateTime" use="required" />

  </xs:complexType>

  <xs:complexType name="Top20MovieProduct">

    <xs:complexContent mixed="false">

      <xs:extension base="apiml:Movie" xmlns:apiml="http://api.redbox.com/Products/MovieList/v2">

        <xs:sequence>

          <xs:element minOccurs="0" maxOccurs="1" name="Top20Entry" xmlns:q1="http://api.redbox.com/Top20/v2" type="q1:Top20TagBase" />

        </xs:sequence>

      </xs:extension>

    </xs:complexContent>

  </xs:complexType>

</xs:schema>

Additional response Xml Schemas:

<xs:schema id="Top20Entry"

    targetNamespace="http://api.redbox.com/Top20/v2"

    elementFormDefault="qualified"

    xmlns="http://api.redbox.com/Top20/v2"

    xmlns:tns="http://api.redbox.com/Top20/v2"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:complexType abstract="true" name="Top20TagBase" >

    <xs:attribute name="position" type="xs:unsignedShort" use="required" />

  </xs:complexType>

  <xs:simpleType name="DirectionEnum">

    <xs:restriction base="xs:string" >

      <xs:enumeration value="Up" id="up" />

      <xs:enumeration value="Down" id="down" />

      <xs:enumeration value="NoChange" id="nochange" />

    </xs:restriction>

  </xs:simpleType>

  <xs:complexType name="NewTop20Entry" >

    <xs:complexContent mixed="false" >

      <xs:extension base="tns:Top20TagBase" >

      </xs:extension>

    </xs:complexContent>

  </xs:complexType>

  <xs:complexType name="OldTop20Entry" >

    <xs:complexContent mixed="false" >

      <xs:extension base="tns:Top20TagBase" >

        <xs:attribute name="direction" type="DirectionEnum" use="required" />

        <xs:attribute name="amount" type="xs:unsignedShort" use="required" />

      </xs:extension>

    </xs:complexContent>

  </xs:complexType>

  <xs:element name="Top20Entry" type="tns:Top20TagBase" />

</xs:schema>

### Cart Schema

<Cart xmlns="http://api.redbox.com/Reservations/v2">

<UserId>String content</UserId>

<StoreRef>String content</StoreRef>

<CardId>String content</CardId>

<CardCVV>String content</CardCVV>

<Device>String content</Device>

<ProductRef>String content</ProductRef>

<ProductRef>String content</ProductRef>

<DiscountsApplied>

<DiscountApplication>

<DiscountType>WebCredit</DiscountType>

<NumDiscountsApplied>2147483647</NumDiscountsApplied>

</DiscountApplication>

<DiscountApplication>

<DiscountType>WebCredit</DiscountType>

<NumDiscountsApplied>2147483647</NumDiscountsApplied>

</DiscountApplication>

</DiscountsApplied>

</Cart>

<xs:schema xmlns:tns="http://api.redbox.com/Reservations/v2" elementFormDefault="qualified" targetNamespace="http://api.redbox.com/Reservations/v2" xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="Cart" nillable="true" type="tns:Cart" />

<xs:complexType name="Cart">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="1" name="UserId" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="StoreRef" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="CardId" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="CardCVV" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="Device" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="unbounded" name="ProductRef" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="DiscountsApplied" type="tns:ArrayOfDiscountApplication" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="ArrayOfDiscountApplication">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="unbounded" name="DiscountApplication" type="tns:DiscountApplication" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="DiscountApplication">

<xs:sequence>

<xs:element minOccurs="1" maxOccurs="1" name="DiscountType" type="tns:DiscountTypes" />

<xs:element minOccurs="1" maxOccurs="1" name="NumDiscountsApplied" type="xs:int" />

</xs:sequence>

</xs:complexType>

<xs:simpleType name="DiscountTypes">

<xs:restriction base="xs:string">

<xs:enumeration value="WebCredit" />

<xs:enumeration value="PromoCode" />

<xs:enumeration value="DigitalCredit" />

</xs:restriction>

</xs:simpleType>

</xs:schema>

### CartValidation Schema

The ErrorMessages element contains a collection of error messages to be displayed to the end-user.

The Errors element contains a collection of Error elements, each of which indicates the category (such as “InventoryError”), the Error type (such as “InventoryNotAvailable”) and the list of product IDs for which the error applies.

<CartValidation xmlns="http://api.redbox.com/Reservations/v2">

<Cart>

<UserId>String content</UserId>

<StoreRef>String content</StoreRef>

<CardId>String content</CardId>

<CardCVV>String content</CardCVV>

<Device>String content</Device>

<ProductRef>String content</ProductRef>

<ProductRef>String content</ProductRef>

<DiscountsToApply>

<DiscountApplication>

<DiscountType>DigitalCredit</DiscountType>

<NumDiscountsApplied>2147483647</NumDiscountsApplied>

</DiscountApplication>

<DiscountApplication>

<DiscountType>DigitalCredit</DiscountType>

<NumDiscountsApplied>2147483647</NumDiscountsApplied>

</DiscountApplication>

</DiscountsToApply>

</Cart>

<ErrorMessages>

<Message>String content</Message>

<Message>String content</Message>

</ErrorMessages>

<Errors>

<Err Category="String content" Error="String content">

<ProductRef>String content</ProductRef>

<ProductRef>String content</ProductRef>

</Err>

<Err Category="String content" Error="String content">

<ProductRef>String content</ProductRef>

<ProductRef>String content</ProductRef>

</Err>

</Errors>

</CartValidation>

Schema:

<xs:schema xmlns:tns="http://api.redbox.com/Reservations/v2" elementFormDefault="qualified" targetNamespace="http://api.redbox.com/Reservations/v2" xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="CartValidation" type="tns:CartValidation" />

<xs:complexType name="CartValidation">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="1" name="Cart" type="tns:Cart" />

<xs:element minOccurs="0" maxOccurs="1" name="ErrorMessages" type="tns:ArrayOfString" />

<xs:element minOccurs="0" maxOccurs="1" name="Errors" type="tns:ArrayOfErr" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="Cart">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="1" name="UserId" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="StoreRef" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="CardId" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="CardCVV" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="Device" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="unbounded" name="ProductRef" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="DiscountsToApply" type="tns:ArrayOfDiscountApplication" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="ArrayOfDiscountApplication">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="unbounded" name="DiscountApplication" type="tns:DiscountApplication" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="DiscountApplication">

<xs:sequence>

<xs:element minOccurs="1" maxOccurs="1" name="DiscountType" type="tns:DiscountTypes" />

<xs:element minOccurs="1" maxOccurs="1" name="NumDiscountsApplied" type="xs:int" />

</xs:sequence>

</xs:complexType>

<xs:simpleType name="DiscountTypes">

<xs:restriction base="xs:string">

<xs:enumeration value="DigitalCredit" />

</xs:restriction>

</xs:simpleType>

<xs:complexType name="ArrayOfString">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="unbounded" name="Message" nillable="true" type="xs:string" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="ArrayOfErr">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="unbounded" name="Err" type="tns:Err" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="Err">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="unbounded" name="ProductRef" type="xs:string" />

</xs:sequence>

<xs:attribute name="Category" type="xs:string" />

<xs:attribute name="Error" type="xs:string" />

</xs:complexType>

</xs:schema>

### PricedCart Schema

<PricedCart xmlns="http://api.redbox.com/Reservations/v2">

<UserId>String content</UserId>

<StoreRef>String content</StoreRef>

<Online>true</Online>

<CardId>String content</CardId>

<CardCVV>String content</CardCVV>

<Device>String content</Device>

<CartItems>

<CartItem ProductRef="String content" Name="String content" ProductType="String content" Price="12678967.543233" ImageUrl="String content" Discount="12678967.543233" DiscountedPrice="12678967.543233" ExtraPrice="12678967.543233" Format="String content" ItemStatus="Valid" Rating="G" />

<CartItem ProductRef="String content" Name="String content" ProductType="String content" Price="12678967.543233" ImageUrl="String content" Discount="12678967.543233" DiscountedPrice="12678967.543233" ExtraPrice="12678967.543233" Format="String content" ItemStatus="Valid" Rating="G" />

</CartItems>

<SubTotal>12678967.543233</SubTotal>

<DiscountedSubTotal>12678967.543233</DiscountedSubTotal>

<GrandTotal>12678967.543233</GrandTotal>

<Tax>12678967.543233</Tax>

<DiscountsApplied>

<DiscountApplied>

<DiscountType>WebCredit</DiscountType>

<NumDiscountsAvailable>2147483647</NumDiscountsAvailable>

<DiscountExpirationMsg>String content</DiscountExpirationMsg>

<NumDiscountsApplied>2147483647</NumDiscountsApplied>

</DiscountApplied>

<DiscountApplied>

<DiscountType>WebCredit</DiscountType>

<NumDiscountsAvailable>2147483647</NumDiscountsAvailable>

<DiscountExpirationMsg>String content</DiscountExpirationMsg>

<NumDiscountsApplied>2147483647</NumDiscountsApplied>

</DiscountApplied>

</DiscountsApplied>

</PricedCart>

<xs:schema xmlns:tns="http://api.redbox.com/Reservations/v2" elementFormDefault="qualified" targetNamespace="http://api.redbox.com/Reservations/v2" xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="PricedCart" nillable="true" type="tns:PricedCart" />

<xs:complexType name="PricedCart">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="1" name="UserId" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="StoreRef" type="xs:string" />

<xs:element minOccurs="1" maxOccurs="1" name="Online" type="xs:boolean" />

<xs:element minOccurs="0" maxOccurs="1" name="CardId" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="CardCVV" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="Device" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="CartItems" type="tns:ArrayOfCartItem" />

<xs:element minOccurs="1" maxOccurs="1" name="SubTotal" type="xs:decimal" />

<xs:element minOccurs="1" maxOccurs="1" name="DiscountedSubTotal" type="xs:decimal" />

<xs:element minOccurs="1" maxOccurs="1" name="GrandTotal" type="xs:decimal" />

<xs:element minOccurs="1" maxOccurs="1" name="Tax" type="xs:decimal" />

<xs:element minOccurs="0" maxOccurs="1" name="DiscountsApplied" type="tns:ArrayOfDiscountApplied" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="ArrayOfCartItem">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="unbounded" name="CartItem" type="tns:CartItem" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="CartItem">

<xs:attribute name="ProductRef" type="xs:string" />

<xs:attribute name="Name" type="xs:string" />

<xs:attribute name="ProductType" type="xs:string" />

<xs:attribute name="Price" type="xs:decimal" use="required" />

<xs:attribute name="ImageUrl" type="xs:string" />

<xs:attribute name="Discount" type="xs:decimal" use="required" />

<xs:attribute name="DiscountedPrice" type="xs:decimal" use="required" />

<xs:attribute name="ExtraPrice" type="xs:decimal" use="required" />

<xs:attribute name="Format" type="xs:string" />

<xs:attribute name="ItemStatus" type="tns:CartItemStatus" use="required" />

<xs:attribute name="Rating" type="tns:MPAARatings" use="required" />

</xs:complexType>

<xs:simpleType name="CartItemStatus">

<xs:restriction base="xs:string">

<xs:enumeration value="Valid" />

<xs:enumeration value="PricingMismatch" />

<xs:enumeration value="Exclude" />

<xs:enumeration value="InventoryNotAvailable" />

<xs:enumeration value="ProductProfileDataNotFound" />

</xs:restriction>

</xs:simpleType>

<xs:simpleType name="MPAARatings">

<xs:restriction base="xs:string">

<xs:enumeration value="G" />

<xs:enumeration value="PG" />

<xs:enumeration value="PG13" />

<xs:enumeration value="R" />

<xs:enumeration value="NR" />

</xs:restriction>

</xs:simpleType>

<xs:complexType name="ArrayOfDiscountApplied">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="unbounded" name="DiscountApplied" type="tns:DiscountApplied" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="DiscountApplied">

<xs:sequence>

<xs:element minOccurs="1" maxOccurs="1" name="DiscountType" type="tns:DiscountTypes" />

<xs:element minOccurs="1" maxOccurs="1" name="NumDiscountsAvailable" type="xs:int" />

<xs:element minOccurs="0" maxOccurs="1" name="DiscountExpirationMsg" type="xs:string" />

<xs:element minOccurs="1" maxOccurs="1" name="NumDiscountsApplied" type="xs:int" />

</xs:sequence>

</xs:complexType>

<xs:simpleType name="DiscountTypes">

<xs:restriction base="xs:string">

<xs:enumeration value="WebCredit" />

<xs:enumeration value="PromoCode" />

<xs:enumeration value="DigitalCredit" />

</xs:restriction>

</xs:simpleType>

</xs:schema>

### PricedCartResponse Schema

<PricedCartResponse xmlns="http://api.redbox.com/Reservations/v2">

<Cart>

<UserId>String content</UserId>

<StoreRef>String content</StoreRef>

<Online>true</Online>

<CardId>String content</CardId>

<CardCVV>String content</CardCVV>

<Device>String content</Device>

<CartItems>

<CartItem ProductRef="String content" Name="String content" ProductType="String content" Price="12678967.543233" ImageUrl="String content" Discount="12678967.543233" DiscountedPrice="12678967.543233" ExtraPrice="12678967.543233" Format="String content" ItemStatus="Valid" Rating="G" />

<CartItem ProductRef="String content" Name="String content" ProductType="String content" Price="12678967.543233" ImageUrl="String content" Discount="12678967.543233" DiscountedPrice="12678967.543233" ExtraPrice="12678967.543233" Format="String content" ItemStatus="Valid" Rating="G" />

</CartItems>

<SubTotal>12678967.543233</SubTotal>

<DiscountedSubTotal>12678967.543233</DiscountedSubTotal>

<GrandTotal>12678967.543233</GrandTotal>

<Tax>12678967.543233</Tax>

<DiscountsApplied>

<DiscountApplied>

<DiscountType>WebCredit</DiscountType>

<NumDiscountsAvailable>2147483647</NumDiscountsAvailable>

<DiscountExpirationMsg>String content</DiscountExpirationMsg>

<NumDiscountsApplied>2147483647</NumDiscountsApplied>

</DiscountApplied>

<DiscountApplied>

<DiscountType>WebCredit</DiscountType>

<NumDiscountsAvailable>2147483647</NumDiscountsAvailable>

<DiscountExpirationMsg>String content</DiscountExpirationMsg>

<NumDiscountsApplied>2147483647</NumDiscountsApplied>

</DiscountApplied>

</DiscountsApplied>

</Cart>

<Validations>

<ErrorMessages>

<Message>String content</Message>

<Message>String content</Message>

</ErrorMessages>

<Errors>

<Err Category="String content" Error="String content">

<ProductRef>String content</ProductRef>

<ProductRef>String content</ProductRef>

</Err>

<Err Category="String content" Error="String content">

<ProductRef>String content</ProductRef>

<ProductRef>String content</ProductRef>

</Err>

</Errors>

</Validations>

</PricedCartResponse>

<xs:schema xmlns:tns="http://api.redbox.com/Reservations/v2" elementFormDefault="qualified" targetNamespace="http://api.redbox.com/Reservations/v2" xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="PricedCartResponse" nillable="true" type="tns:PricedCartResponse" />

<xs:complexType name="PricedCartResponse">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="1" name="Cart" type="tns:PricedCart" />

<xs:element minOccurs="1" maxOccurs="1" name="Validations" type="tns:CartValidation" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="PricedCart">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="1" name="UserId" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="StoreRef" type="xs:string" />

<xs:element minOccurs="1" maxOccurs="1" name="Online" type="xs:boolean" />

<xs:element minOccurs="0" maxOccurs="1" name="CardId" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="CardCVV" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="Device" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="CartItems" type="tns:ArrayOfCartItem" />

<xs:element minOccurs="1" maxOccurs="1" name="SubTotal" type="xs:decimal" />

<xs:element minOccurs="1" maxOccurs="1" name="DiscountedSubTotal" type="xs:decimal" />

<xs:element minOccurs="1" maxOccurs="1" name="GrandTotal" type="xs:decimal" />

<xs:element minOccurs="1" maxOccurs="1" name="Tax" type="xs:decimal" />

<xs:element minOccurs="0" maxOccurs="1" name="DiscountsApplied" type="tns:ArrayOfDiscountApplied" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="ArrayOfCartItem">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="unbounded" name="CartItem" type="tns:CartItem" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="CartItem">

<xs:attribute name="ProductRef" type="xs:string" />

<xs:attribute name="Name" type="xs:string" />

<xs:attribute name="ProductType" type="xs:string" />

<xs:attribute name="Price" type="xs:decimal" use="required" />

<xs:attribute name="ImageUrl" type="xs:string" />

<xs:attribute name="Discount" type="xs:decimal" use="required" />

<xs:attribute name="DiscountedPrice" type="xs:decimal" use="required" />

<xs:attribute name="ExtraPrice" type="xs:decimal" use="required" />

<xs:attribute name="Format" type="xs:string" />

<xs:attribute name="ItemStatus" type="tns:CartItemStatus" use="required" />

<xs:attribute name="Rating" type="tns:MPAARatings" use="required" />

</xs:complexType>

<xs:simpleType name="CartItemStatus">

<xs:restriction base="xs:string">

<xs:enumeration value="Valid" />

<xs:enumeration value="PricingMismatch" />

<xs:enumeration value="Exclude" />

<xs:enumeration value="InventoryNotAvailable" />

<xs:enumeration value="ProductProfileDataNotFound" />

</xs:restriction>

</xs:simpleType>

<xs:simpleType name="MPAARatings">

<xs:restriction base="xs:string">

<xs:enumeration value="G" />

<xs:enumeration value="PG" />

<xs:enumeration value="PG-13" />

<xs:enumeration value="R" />

<xs:enumeration value="NR" />

</xs:restriction>

</xs:simpleType>

<xs:complexType name="ArrayOfDiscountApplied">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="unbounded" name="DiscountApplied" type="tns:DiscountApplied" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="DiscountApplied">

<xs:sequence>

<xs:element minOccurs="1" maxOccurs="1" name="DiscountType" type="tns:DiscountTypes" />

<xs:element minOccurs="1" maxOccurs="1" name="NumDiscountsAvailable" type="xs:int" />

<xs:element minOccurs="0" maxOccurs="1" name="DiscountExpirationMsg" type="xs:string" />

<xs:element minOccurs="1" maxOccurs="1" name="NumDiscountsApplied" type="xs:int" />

</xs:sequence>

</xs:complexType>

<xs:simpleType name="DiscountTypes">

<xs:restriction base="xs:string">

<xs:enumeration value="WebCredit" />

<xs:enumeration value="PromoCode" />

<xs:enumeration value="DigitalCredit" />

</xs:restriction>

</xs:simpleType>

<xs:complexType name="CartValidation">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="1" name="ErrorMessages" type="tns:ArrayOfString" />

<xs:element minOccurs="0" maxOccurs="1" name="Errors" type="tns:ArrayOfErr" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="ArrayOfString">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="unbounded" name="Message" nillable="true" type="xs:string" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="ArrayOfErr">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="unbounded" name="Err" type="tns:Err" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="Err">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="unbounded" name="ProductRef" type="xs:string" />

</xs:sequence>

<xs:attribute name="Category" type="xs:string" />

<xs:attribute name="Error" type="xs:string" />

</xs:complexType>

</xs:schema>

### ReservationResponse Schema

Example:

<ReservationResponse xmlns="http://api.redbox.com/Reservations/v2">

<Cart>

<UserId>String content</UserId>

<StoreRef>String content</StoreRef>

<Online>true</Online>

<CardId>String content</CardId>

<CardCVV>String content</CardCVV>

<Device>String content</Device>

<CartItems>

<CartItem ProductRef="String content" Name="String content" ProductType="String content" Price="12678967.543233" ImageUrl="String content" Discount="12678967.543233" DiscountedPrice="12678967.543233" ExtraPrice="12678967.543233" Format="String content" ItemStatus="Valid" Rating="G" />

<CartItem ProductRef="String content" Name="String content" ProductType="String content" Price="12678967.543233" ImageUrl="String content" Discount="12678967.543233" DiscountedPrice="12678967.543233" ExtraPrice="12678967.543233" Format="String content" ItemStatus="Valid" Rating="G" />

</CartItems>

<SubTotal>12678967.543233</SubTotal>

<DiscountedSubTotal>12678967.543233</DiscountedSubTotal>

<GrandTotal>12678967.543233</GrandTotal>

<Tax>12678967.543233</Tax>

<DiscountsApplied>

<DiscountApplied>

<DiscountType>WebCredit</DiscountType>

<NumDiscountsAvailable>2147483647</NumDiscountsAvailable>

<DiscountExpirationMsg>String content</DiscountExpirationMsg>

<NumDiscountsApplied>2147483647</NumDiscountsApplied>

</DiscountApplied>

<DiscountApplied>

<DiscountType>WebCredit</DiscountType>

<NumDiscountsAvailable>2147483647</NumDiscountsAvailable>

<DiscountExpirationMsg>String content</DiscountExpirationMsg>

<NumDiscountsApplied>2147483647</NumDiscountsApplied>

</DiscountApplied>

</DiscountsApplied>

</Cart>

<Validations>

<ErrorMessages>

<Message>String content</Message>

<Message>String content</Message>

</ErrorMessages>

<Errors>

<Err Category="String content" Error="String content">

<ProductRef>String content</ProductRef>

<ProductRef>String content</ProductRef>

</Err>

<Err Category="String content" Error="String content">

<ProductRef>String content</ProductRef>

<ProductRef>String content</ProductRef>

</Err>

</Errors>

</Validations>

<ReservationId>String content</ReservationId>

<PickupBy>String content</PickupBy>

</ReservationResponse>

Schema:

<xs:schema xmlns:tns="http://api.redbox.com/Reservations/v2" elementFormDefault="qualified" targetNamespace="http://api.redbox.com/Reservations/v2" xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="ReservationResponse" nillable="true" type="tns:ReservationResponse" />

<xs:complexType name="ReservationResponse">

<xs:complexContent mixed="false">

<xs:extension base="tns:PricedCartResponse">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="1" name="ReservationId" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="PickupBy" type="xs:string" />

</xs:sequence>

</xs:extension>

</xs:complexContent>

</xs:complexType>

<xs:complexType name="PricedCartResponse">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="1" name="Cart" type="tns:PricedCart" />

<xs:element minOccurs="1" maxOccurs="1" name="Validations" type="tns:CartValidation" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="PricedCart">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="1" name="UserId" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="StoreRef" type="xs:string" />

<xs:element minOccurs="1" maxOccurs="1" name="Online" type="xs:boolean" />

<xs:element minOccurs="0" maxOccurs="1" name="CardId" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="CardCVV" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="Device" type="xs:string" />

<xs:element minOccurs="0" maxOccurs="1" name="CartItems" type="tns:ArrayOfCartItem" />

<xs:element minOccurs="1" maxOccurs="1" name="SubTotal" type="xs:decimal" />

<xs:element minOccurs="1" maxOccurs="1" name="DiscountedSubTotal" type="xs:decimal" />

<xs:element minOccurs="1" maxOccurs="1" name="GrandTotal" type="xs:decimal" />

<xs:element minOccurs="1" maxOccurs="1" name="Tax" type="xs:decimal" />

<xs:element minOccurs="0" maxOccurs="1" name="DiscountsApplied" type="tns:ArrayOfDiscountApplied" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="ArrayOfCartItem">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="unbounded" name="CartItem" type="tns:CartItem" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="CartItem">

<xs:attribute name="ProductRef" type="xs:string" />

<xs:attribute name="Name" type="xs:string" />

<xs:attribute name="ProductType" type="xs:string" />

<xs:attribute name="Price" type="xs:decimal" use="required" />

<xs:attribute name="ImageUrl" type="xs:string" />

<xs:attribute name="Discount" type="xs:decimal" use="required" />

<xs:attribute name="DiscountedPrice" type="xs:decimal" use="required" />

<xs:attribute name="ExtraPrice" type="xs:decimal" use="required" />

<xs:attribute name="Format" type="xs:string" />

<xs:attribute name="ItemStatus" type="tns:CartItemStatus" use="required" />

<xs:attribute name="Rating" type="tns:MPAARatings" use="required" />

</xs:complexType>

<xs:simpleType name="CartItemStatus">

<xs:restriction base="xs:string">

<xs:enumeration value="Valid" />

<xs:enumeration value="PricingMismatch" />

<xs:enumeration value="Exclude" />

<xs:enumeration value="InventoryNotAvailable" />

<xs:enumeration value="ProductProfileDataNotFound" />

</xs:restriction>

</xs:simpleType>

<xs:simpleType name="MPAARatings">

<xs:restriction base="xs:string">

<xs:enumeration value="G" />

<xs:enumeration value="PG" />

<xs:enumeration value="PG-13" />

<xs:enumeration value="R" />

<xs:enumeration value="NR" />

</xs:restriction>

</xs:simpleType>

<xs:complexType name="ArrayOfDiscountApplied">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="unbounded" name="DiscountApplied" type="tns:DiscountApplied" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="DiscountApplied">

<xs:sequence>

<xs:element minOccurs="1" maxOccurs="1" name="DiscountType" type="tns:DiscountTypes" />

<xs:element minOccurs="1" maxOccurs="1" name="NumDiscountsAvailable" type="xs:int" />

<xs:element minOccurs="0" maxOccurs="1" name="DiscountExpirationMsg" type="xs:string" />

<xs:element minOccurs="1" maxOccurs="1" name="NumDiscountsApplied" type="xs:int" />

</xs:sequence>

</xs:complexType>

<xs:simpleType name="DiscountTypes">

<xs:restriction base="xs:string">

<xs:enumeration value="WebCredit" />

<xs:enumeration value="PromoCode" />

<xs:enumeration value="DigitalCredit" />

</xs:restriction>

</xs:simpleType>

<xs:complexType name="CartValidation">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="1" name="ErrorMessages" type="tns:ArrayOfString" />

<xs:element minOccurs="0" maxOccurs="1" name="Errors" type="tns:ArrayOfErr" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="ArrayOfString">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="unbounded" name="Message" nillable="true" type="xs:string" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="ArrayOfErr">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="unbounded" name="Err" type="tns:Err" />

</xs:sequence>

</xs:complexType>

<xs:complexType name="Err">

<xs:sequence>

<xs:element minOccurs="0" maxOccurs="unbounded" name="ProductRef" type="xs:string" />

</xs:sequence>

<xs:attribute name="Category" type="xs:string" />

<xs:attribute name="Error" type="xs:string" />

</xs:complexType>

</xs:schema>

### BasicProfile Schema

Example BasicProfile:

<CustomerAccountProfile xmlns="http://api.redbox.com/CustomerAccount/v2">

  <AccountCreationSource>AccountCreationSource1</AccountCreationSource>

  <AnniversaryDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

  <BirthDay xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

  <BirthMonth xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

  <CityName>CityName1</CityName>

  <ContactEmailAddress>ContactEmailAddress1</ContactEmailAddress>

  <CustomerNumber>CustomerNumber1</CustomerNumber>

  <DefaultAccountNumber>DefaultAccountNumber1</DefaultAccountNumber>

  <DisplayName>DisplayName1</DisplayName>

  <FirstName>FirstName1</FirstName>

  <IsActive xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

  <IsMobileUser xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

  <LastName>LastName1</LastName>

  <LastUpdatedTime>1900-01-01T01:01:01</LastUpdatedTime>

  <MobilePhoneNumber>MobilePhoneNumber1</MobilePhoneNumber>

  <NotificationEmailAddress>NotificationEmailAddress1</NotificationEmailAddress>

  <StateCd>StateCd1</StateCd>

  <StreetAddressLine1>StreetAddressLine11</StreetAddressLine1>

  <StreetAddressLine2>StreetAddressLine21</StreetAddressLine2>

  <ZipPostalCode>ZipPostalCode1</ZipPostalCode>

  <Extensions />

</CustomerAccountProfile>

Schema:

<xs:schema id="CustomerAccountProfile"

    targetNamespace="http://api.redbox.com/CustomerAccount/v2"

    elementFormDefault="qualified"

    xmlns="http://api.redbox.com/CustomerAccount/v2"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

    xmlns:oai="http://api.redbox.com/OpenAPI/v1"

>

  <xs:import namespace="http://api.redbox.com/OpenAPI/v1" />

  <xs:complexType name="CA\_CustomerAccountProfile" >

    <xs:sequence>

      <xs:element name="AccountCreationSource" type="xs:string" />

      <xs:element name="AnniversaryDate" type="xs:date"  nillable="true"/>

      <xs:element name="BirthDay" type="xs:unsignedInt" nillable="true"/>

      <xs:element name="BirthMonth" type="xs:unsignedInt" nillable="true"/>

      <xs:element name="CityName" type="xs:string"/>

      <xs:element name="ContactEmailAddress" type="xs:string"/>

      <xs:element name="CustomerNumber" type="xs:string" />

      <xs:element name="DefaultAccountNumber" type="xs:string" />

      <xs:element name="DisplayName" type="xs:string"/>

      <xs:element name="FirstName" type="xs:string" />

      <xs:element name="IsActive" type="xs:boolean" nillable="true"/>

      <xs:element name="IsMobileUser" type="xs:boolean" nillable="true" />

      <xs:element name="LastName" type="xs:string"/>

      <xs:element name="LastUpdatedTime" type="xs:dateTime"/>

      <xs:element name="MobilePhoneNumber" type="xs:string" />

      <xs:element name="NotificationEmailAddress" type="xs:string"/>

      <xs:element name="StateCd" type="xs:string" />

      <xs:element name="StreetAddressLine1" type="xs:string"/>

      <xs:element name="StreetAddressLine2" type="xs:string" />

      <xs:element name="ZipPostalCode" type="xs:string" />

      <xs:element name="Extensions" type="oai:ExtensionData" />

    </xs:sequence>

  </xs:complexType>

  <xs:element name="CustomerAccountProfile" type="CA\_CustomerAccountProfile" />

</xs:schema>

### Preferences

Example:

<Preferences lastUpdated="1900-01-01T01:01:01" xmlns="http://api.redbox.com/CustomerAccount/v2">

  <OptInForBetaTesting>true</OptInForBetaTesting>

  <OptInForEmailReceipts>true</OptInForEmailReceipts>

  <OptInForSmsMessages>true</OptInForSmsMessages>

  <FavoriteStores lastUpdated="1900-01-01T01:01:01">

    <StoreId>StoreId1</StoreId>

    <StoreId>StoreId2</StoreId>

    <StoreId>StoreId3</StoreId>

  </FavoriteStores>

  <GamePreferences preferredFormatType="preferredFormatType1" lastUpdated="1900-01-01T01:01:01">

    <Genre>Genre1</Genre>

    <Genre>Genre2</Genre>

    <Genre>Genre3</Genre>

  </GamePreferences>

  <MoviePreferences preferredFormatType="preferredFormatType1" lastUpdated="1900-01-01T01:01:01">

    <Genre>Genre1</Genre>

    <Genre>Genre2</Genre>

    <Genre>Genre3</Genre>

  </MoviePreferences>

  <InterestedComingSoonTitles lastUpdated="1900-01-01T01:01:01">

    <ProductId>ProductId1</ProductId>

    <ProductId>ProductId2</ProductId>

    <ProductId>ProductId3</ProductId>

  </InterestedComingSoonTitles>

  <Extensions />

</Preferences>

Schema:

<xs:schema id="CA\_Preferences\_V2"

    targetNamespace="http://api.redbox.com/CustomerAccount/v2"

    elementFormDefault="qualified"

    xmlns="http://api.redbox.com/CustomerAccount/v2"

    xmlns:tns="http://api.redbox.com/CustomerAccount/v2"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

    xmlns:oai="http://api.redbox.com/OpenAPI/v1"

>

  <xs:import namespace="http://api.redbox.com/OpenAPI/v1" />

  <xs:complexType name="CA\_Preferences" >

    <xs:sequence>

      <xs:element name="OptInForBetaTesting" type="xs:boolean"/>

      <xs:element name="OptInForEmailReceipts" type="xs:boolean"/>

      <xs:element name="OptInForSmsMessages" type="xs:boolean"/>

      <xs:element name="FavoriteStores" type="tns:ArrayOfStoreIds" />

      <xs:element name="GamePreferences" type="tns:ProductPreferencesType" />

      <xs:element name="MoviePreferences" type="tns:ProductPreferencesType"/>

      <xs:element name="InterestedComingSoonTitles" type="ArrayOfProductIds"/>

      <xs:element name="Extensions" type="oai:ExtensionData" />

    </xs:sequence>

    <xs:attribute name="lastUpdated" type="xs:dateTime" />

  </xs:complexType>

  <xs:complexType name="ArrayOfGenres"  >

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="Genre" type="xs:string" />

    </xs:sequence>

    <xs:attribute name="lastUpdated" type="xs:dateTime"/>

  </xs:complexType>

  <xs:complexType name="ProductPreferencesType" >

    <xs:complexContent >

      <xs:extension base="tns:ArrayOfGenres" >

        <xs:attribute name="preferredFormatType" type="xs:string"/>

      </xs:extension>

    </xs:complexContent>

  </xs:complexType>

  <xs:complexType name="ArrayOfStoreIds"  >

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="StoreId" type="xs:string" />

    </xs:sequence>

    <xs:attribute name="lastUpdated" type="xs:dateTime"/>

  </xs:complexType>

  <xs:complexType name="ArrayOfProductIds"  >

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="ProductId" type="xs:string" />

    </xs:sequence>

    <xs:attribute name="lastUpdated" type="xs:dateTime"/>

  </xs:complexType>

  <xs:element name="Preferences" type="tns:CA\_Preferences" />

</xs:schema>

### RentalHistory

Example:

<RentalHistory lastUpdated="1900-01-01T01:01:01" xmlns="http://api.redbox.com/CustomerAccount/v2">

  <RentalRecord productId="productId1">

    <ProductTitle>ProductTitle1</ProductTitle>

    <RentedDate>1900-01-01</RentedDate>

    <ReturnedDate>1900-01-01</ReturnedDate>

    <Detail>Detail1</Detail>

    <FormatType>FormatType1</FormatType>

    <Extensions />

  </RentalRecord>

  <RentalRecord productId="productId2">

    <ProductTitle>ProductTitle2</ProductTitle>

    <RentedDate>0001-01-01</RentedDate>

    <ReturnedDate>0001-01-01</ReturnedDate>

    <Detail>Detail2</Detail>

    <FormatType>FormatType2</FormatType>

    <Extensions />

  </RentalRecord>

  <RentalRecord productId="productId3">

    <ProductTitle>ProductTitle3</ProductTitle>

    <RentedDate>9999-12-31</RentedDate>

    <ReturnedDate>9999-12-31</ReturnedDate>

    <Detail>Detail3</Detail>

    <FormatType>FormatType3</FormatType>

    <Extensions />

  </RentalRecord>

</RentalHistory>

Schema:

<xs:schema id="CA\_RentalHistory\_V2"

    targetNamespace="http://api.redbox.com/CustomerAccount/v2"

    elementFormDefault="qualified"

    xmlns="http://api.redbox.com/CustomerAccount/v2"

    xmlns:tns="http://api.redbox.com/CustomerAccount/v2"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

    xmlns:oai="http://api.redbox.com/OpenAPI/v1"

>

  <xs:import namespace="http://api.redbox.com/OpenAPI/v1" />

  <xs:complexType name="CA\_RentalHistory" >

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="RentalRecord" type="tns:RentalRecordType" />

    </xs:sequence>

    <xs:attribute name="lastUpdated" type="xs:dateTime" />

  </xs:complexType>

  <xs:complexType name="RentalRecordType" >

    <xs:sequence>

      <xs:element name="ProductTitle" type="xs:string"/>

      <xs:element name="RentedDate" type="xs:date"/>

      <xs:element name="ReturnedDate" type="xs:date" />

      <xs:element name="Detail" type="xs:string"/>

      <xs:element name="FormatType" type="xs:string" />

      <xs:element name="Extensions" type="oai:ExtensionData" />

    </xs:sequence>

    <xs:attribute name="productId" type="xs:string"/>

  </xs:complexType>

  <xs:element name="RentalHistory" type="CA\_RentalHistory" />

</xs:schema>

### QueuesListResponse

Example:

<QueueList xmlns="http://api.redbox.com/CustomerAccount/v2">

  <Queue>

    <QueueName>QueueName1</QueueName>

    <QueueId>QueueId1</QueueId>

    <ProductTypes>ProductTypes1</ProductTypes>

    <Extensions />

  </Queue>

  <Queue>

    <QueueName>QueueName2</QueueName>

    <QueueId>QueueId2</QueueId>

    <ProductTypes>ProductTypes2</ProductTypes>

    <Extensions />

  </Queue>

  <Queue>

    <QueueName>QueueName3</QueueName>

    <QueueId>QueueId3</QueueId>

    <ProductTypes>ProductTypes3</ProductTypes>

    <Extensions />

  </Queue>

</QueueList>

Schema:

<xs:schema id="CA\_QueueList\_V2"

    targetNamespace="http://api.redbox.com/CustomerAccount/v2"

    elementFormDefault="qualified"

    xmlns="http://api.redbox.com/CustomerAccount/v2"

    xmlns:tns="http://api.redbox.com/CustomerAccount/v2"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:include schemaLocation="CA\_QueueInfo\_V2.xsd" />

  <xs:complexType name="CA\_QueueList" mixed="false" >

    <xs:sequence >

      <xs:element minOccurs="0" maxOccurs="unbounded" name="Queue" type="tns:CA\_QueueInfo" />

    </xs:sequence>

  </xs:complexType>

  <xs:element name="QueueList" type="CA\_QueueList" />

</xs:schema>

Additional Schema:

See QueueInfo in the next section.

### QueueInfo

Example:

<Queue xmlns="http://api.redbox.com/CustomerAccount/v2">

  <QueueInfo>

    <QueueName>QueueName1</QueueName>

    <QueueId>QueueId1</QueueId>

    <ProductTypes>ProductTypes1</ProductTypes>

    <Extensions />

  </QueueInfo>

  <QueueContent>

    <QueuedProductId position="0">QueuedProductId1</QueuedProductId>

    <QueuedProductId position="4294967295">QueuedProductId2</QueuedProductId>

    <QueuedProductId position="1">QueuedProductId3</QueuedProductId>

  </QueueContent>

</Queue>

Schema:

<xs:schema id="CA\_QueueInfo\_V2"

    targetNamespace="http://api.redbox.com/CustomerAccount/v2"

    elementFormDefault="qualified"

    xmlns="http://api.redbox.com/CustomerAccount/v2"

    xmlns:tns="http://api.redbox.com/CustomerAccount/v2"

    xmlns:oai="http://api.redbox.com/OpenAPI/v1"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:import namespace="http://api.redbox.com/OpenAPI/v1" />

  <xs:complexType name="CA\_Queue" >

    <xs:sequence >

      <xs:element name="QueueInfo" type="CA\_QueueInfo" />

      <xs:element name="QueueContent" type="tns:ArrayOfQueueObject" />

    </xs:sequence>

  </xs:complexType>

  <xs:complexType name="CA\_QueueInfo" >

    <xs:sequence>

      <xs:element name="QueueName" type="xs:string" />

      <xs:element name="QueueId" type="xs:string" />

      <xs:element name="ProductTypes" type="xs:string" />

      <xs:element name="Extensions" type="oai:ExtensionData" />

    </xs:sequence>

  </xs:complexType>

  <xs:complexType name="ArrayOfQueueObject" >

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="QueuedProductId" type="tns:QueueObject" />

    </xs:sequence>

  </xs:complexType>

  <xs:complexType name="QueueObject">

    <xs:simpleContent >

      <xs:extension base="xs:string" >

        <xs:attribute name="position" type="xs:unsignedInt" />

      </xs:extension>

    </xs:simpleContent>

  </xs:complexType>

  <xs:element name="Queue" type="CA\_Queue" />

</xs:schema>

### RecommendedProductsResponse

Example:

<RecommendedProductsResponse generatedAt="1900-01-01T01:01:01-06:00" productType="productType1" xmlns="http://api.redbox.com/CustomerAccount/v2">

  <ProductId>ProductId1</ProductId>

  <ProductId>ProductId2</ProductId>

  <ProductId>ProductId3</ProductId>

</RecommendedProductsResponse>

Schema:

<xs:schema id="CA\_Recommendations\_V2"

    targetNamespace="http://api.redbox.com/CustomerAccount/v2"

    elementFormDefault="qualified"

    xmlns="http://api.redbox.com/CustomerAccount/v2"

    xmlns:mstns="http://api.redbox.com/CustomerAccount/v2"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:complexType name="CA\_RecommendedProductsResponse" >

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="ProductId" type="xs:string" />

    </xs:sequence>

    <xs:attribute name="generatedAt" type="xs:dateTime" />

    <xs:attribute name="productType" type="xs:string" />

  </xs:complexType>

  <xs:element name="RecommendedProductsResponse" type="CA\_RecommendedProductsResponse" />

</xs:schema>

### AccountListResponse

Example:

<AccountList lastUpdated="1900-01-01T01:01:01xmlns="http://api.redbox.com/CustomerAccount/v2">

  <Account>

    <AccountNumber>AccountNumber1</AccountNumber>

    <AccountAlias>AccountAlias1</AccountAlias>

    <LastFour>LastFour1</LastFour>

    <ExpirationDate>ExpirationDate1</ExpirationDate>

    <Extensions />

  </Account>

  <Account>

    <AccountNumber>AccountNumber2</AccountNumber>

    <AccountAlias>AccountAlias2</AccountAlias>

    <LastFour>LastFour2</LastFour>

    <ExpirationDate>ExpirationDate2</ExpirationDate>

    <Extensions />

  </Account>

  <Account>

    <AccountNumber>AccountNumber3</AccountNumber>

    <AccountAlias>AccountAlias3</AccountAlias>

    <LastFour>LastFour3</LastFour>

    <ExpirationDate>ExpirationDate3</ExpirationDate>

    <Extensions />

  </Account>

</AccountList>

Schema:

<xs:schema id="CA\_AccountList\_v2"

    targetNamespace="http://api.redbox.com/CustomerAccount/v2"

    elementFormDefault="qualified"

    xmlns="http://api.redbox.com/CustomerAccount/v2"

    xmlns:tns="http://api.redbox.com/CustomerAccount/v2"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:include schemaLocation="CA\_AccountInfo\_V2.xsd" />

  <xs:complexType name="CA\_AccountList" >

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="Account" type="tns:CA\_AccountInfo" />

    </xs:sequence>

    <xs:attribute name="lastUpdated" type="xs:dateTime"/>

  </xs:complexType>

  <xs:element name="AccountList" type="tns:CA\_AccountList" />

</xs:schema>

### AccountInfo

Example:

<AccountInfo xmlns="http://api.redbox.com/CustomerAccount/v2">

  <AccountNumber>AccountNumber1</AccountNumber>

  <AccountAlias>AccountAlias1</AccountAlias>

  <LastFour>LastFour1</LastFour>

  <ExpirationDate>ExpirationDate1</ExpirationDate>

  <Extensions />

</AccountInfo>

Schema:

<xs:schema id="CA\_AccountInfo\_V2"

    targetNamespace="http://api.redbox.com/CustomerAccount/v2"

    elementFormDefault="qualified"

    xmlns="http://api.redbox.com/CustomerAccount/v2"

    xmlns:oai="http://api.redbox.com/OpenAPI/v1"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:import namespace="http://api.redbox.com/OpenAPI/v1"/>

  <xs:complexType name="CA\_AccountInfo" >

    <xs:sequence>

      <xs:element name="AccountNumber" type="xs:string" />

      <xs:element name="AccountAlias" type="xs:string"/>

      <xs:element name="LastFour" type="xs:string"/>

      <xs:element name="ExpirationDate" type="xs:string" />

      <xs:element name="Extensions" type="oai:ExtensionData"/>

    </xs:sequence>

  </xs:complexType>

  <xs:element name="AccountInfo" type="CA\_AccountInfo" />

</xs:schema>

### CreditBalance

Example:

<CreditBalances lastUpdated="1900-01-01T01:01:01" xmlns="http://api.redbox.com/CustomerAccount/v2">

  <Balance>0</Balance>

</CreditBalances>

Schema:

<xs:schema id="CA\_CreditBalances\_V2"

    targetNamespace="http://api.redbox.com/CustomerAccount/v2"

    elementFormDefault="qualified"

    xmlns="http://api.redbox.com/CustomerAccount/v2"

    xmlns:mstns="http://api.redbox.com/CustomerAccount/v2"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:complexType name="CA\_CreditBalances" >

    <xs:sequence>

      <xs:element name="Balance" type="xs:unsignedInt" />

    </xs:sequence>

    <xs:attribute name="lastUpdated" type="xs:dateTime" />

  </xs:complexType>

  <xs:element name="CreditBalances" type="CA\_CreditBalances" />

</xs:schema>

### CreditsAvailable

Example:

<CreditsAvailable lastUpdated="1900-01-01T01:01:01" xmlns="http://api.redbox.com/CustomerAccount/v2">

  <Credit>

    <CreditId>CreditId1</CreditId>

    <CreditName>CreditName1</CreditName>

    <EffectiveDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    <ExpirationDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    <Extensions />

  </Credit>

  <Credit>

    <CreditId>CreditId2</CreditId>

    <CreditName>CreditName2</CreditName>

    <EffectiveDate>1900-01-01</EffectiveDate>

    <ExpirationDate>1900-01-01</ExpirationDate>

    <Extensions />

  </Credit>

  <Credit>

    <CreditId>CreditId3</CreditId>

    <CreditName>CreditName3</CreditName>

    <EffectiveDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    <ExpirationDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    <Extensions />

  </Credit>

</CreditsAvailable>

Schema:

<xs:schema id="CA\_CreditsAvailable\_V2"

    targetNamespace="http://api.redbox.com/CustomerAccount/v2"

    elementFormDefault="qualified"

    xmlns="http://api.redbox.com/CustomerAccount/v2"

    xmlns:tns="http://api.redbox.com/CustomerAccount/v2"

    xmlns:oai="http://api.redbox.com/OpenAPI/v1"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:import namespace="http://api.redbox.com/OpenAPI/v1"/>

  <xs:complexType name="CA\_CreditsAvailable" >

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="Credit" type="tns:CreditType" />

    </xs:sequence>

    <xs:attribute name="lastUpdated" type="xs:dateTime" />

  </xs:complexType>

  <xs:complexType name="CreditType" >

    <xs:sequence>

      <xs:element name="CreditId" type="xs:string" />

      <xs:element name="CreditName" type="xs:string" />

      <xs:element name="EffectiveDate" type="xs:date" nillable="true" />

      <xs:element name="ExpirationDate" type="xs:date" nillable="true" />

      <xs:element name="Extensions" type="oai:ExtensionData"/>

    </xs:sequence>

  </xs:complexType>

  <xs:element name="CreditsAvailable" type="CA\_CreditsAvailable" />

</xs:schema>

### CreditUseHistory

Example:

<CreditUseHistory lastUpdated="1900-01-01T01:01:01" xmlns="http://api.redbox.com/CustomerAccount/v2">

  <CreditHistory>

    <CreditId>CreditId1</CreditId>

    <CreditName>CreditName1</CreditName>

    <CreditState>Redeemed</CreditState>

    <EffectiveDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    <ExpirationDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    <RedemptionDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    <RedeemProductId>RedeemProductId1</RedeemProductId>

    <Extensions />

  </CreditHistory>

  <CreditHistory>

    <CreditId>CreditId2</CreditId>

    <CreditName>CreditName2</CreditName>

    <CreditState>Expired</CreditState>

    <EffectiveDate>1900-01-01</EffectiveDate>

    <ExpirationDate>1900-01-01</ExpirationDate>

    <RedemptionDate>1900-01-01</RedemptionDate>

    <RedeemProductId>RedeemProductId2</RedeemProductId>

    <Extensions />

  </CreditHistory>

  <CreditHistory>

    <CreditId>CreditId3</CreditId>

    <CreditName>CreditName3</CreditName>

    <CreditState>Deleted</CreditState>

    <EffectiveDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    <ExpirationDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    <RedemptionDate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />

    <RedeemProductId>RedeemProductId3</RedeemProductId>

    <Extensions />

  </CreditHistory>

</CreditUseHistory>

Schema:

<xs:schema id="CA\_CreditUseHistory\_V2"

    targetNamespace="http://api.redbox.com/CustomerAccount/v2"

    elementFormDefault="qualified"

    xmlns="http://api.redbox.com/CustomerAccount/v2"

    xmlns:tns="http://api.redbox.com/CustomerAccount/v2"

    xmlns:oai="http://api.redbox.com/OpenAPI/v1"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:import namespace="http://api.redbox.com/OpenAPI/v1" />

  <xs:complexType name="CA\_CreditUseHistory" >

    <xs:sequence>

          <xs:element minOccurs="0" maxOccurs="unbounded" name="CreditHistory" type="tns:CreditUseRecordType" />

    </xs:sequence>

    <xs:attribute name="lastUpdated" type="xs:dateTime" />

  </xs:complexType>

   <xs:complexType name="CreditUseRecordType" >

    <xs:sequence>

      <xs:element name="CreditId" type="xs:string" />

      <xs:element name="CreditName" type="xs:string" />

      <xs:element name="CreditState" type="tns:CreditStates" />

      <xs:element name="EffectiveDate" type="xs:date"  nillable="true" />

      <xs:element name="ExpirationDate" type="xs:date" nillable="true" />

      <xs:element name="RedemptionDate" type="xs:date" nillable="true" />

      <xs:element name="RedeemProductId" type="xs:string" />

      <xs:element name="Extensions" type="oai:ExtensionData" />

    </xs:sequence>

  </xs:complexType>

  <xs:simpleType name="CreditStates" >

    <xs:restriction base="xs:string">

      <xs:enumeration value="Redeemed" />

      <xs:enumeration value="Expired" />

      <xs:enumeration value="Deleted" />

    </xs:restriction>

  </xs:simpleType>

  <xs:element name="CreditUseHistory" type="CA\_CreditUseHistory" />

</xs:schema>

### CreditCardInfo

Example:

<CreditCardInfo preferred="true" xmlns="http://api.redbox.com/CustomerAccount/v2">

  <Type>Type1</Type>

  <Number>Number1</Number>

  <Name>Name1</Name>

  <Zip>Zip1</Zip>

  <Month>Month1</Month>

  <Year>Year1</Year>

  <CVV>CVV1</CVV>

  <Alias>Alias1</Alias>

</CreditCardInfo>

Schema:

<xs:schema id="CA\_CreditCardInfo\_V2"

    targetNamespace="http://api.redbox.com/CustomerAccount/v2"

    elementFormDefault="qualified"

    xmlns="http://api.redbox.com/CustomerAccount/v2"

    xmlns:mstns="http://api.redbox.com/CustomerAccount/v2"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:complexType name="CreditCardInfo">

    <xs:sequence>

      <xs:element name="Type" type="xs:string" />

      <xs:element name="Number" type="xs:string" />

      <xs:element name="Name" type="xs:string" />

      <xs:element name="Zip" type="xs:string" />

      <xs:element name="Month" type="xs:string" />

      <xs:element name="Year" type="xs:string" />

      <xs:element name="CVV" type="xs:string" />

      <xs:element name="Alias" type="xs:string" />

    </xs:sequence>

    <xs:attribute name="preferred" type="xs:boolean" use="optional" />

  </xs:complexType>

  <xs:element name="CreditCardInfo" type="CreditCardInfo" />

</xs:schema>

### CacheRefreshCommand Message Schema

<?xml version="1.0" encoding="utf-8"?>

<xs:schema id="CacheRefreshCommand"

    targetNamespace="http://www.redbox.com/OpenAPI/CacheRefreshCommands/v2"

    elementFormDefault="qualified"

    xmlns="http://www.redbox.com/OpenAPI/CacheRefreshCommands/v2"

    xmlns:mstns="http://www.redbox.com/OpenAPI/CacheRefreshCommands/v2"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:complexType name="CacheRefreshCommand" >

         <xs:attribute name="dateTimeIssued" use="required" type="xs:dateTime" />

  </xs:complexType>

</xs:schema>

### ProductsCacheRefreshCommand Message Schema

<?xml version="1.0" encoding="utf-8"?>

<xs:schema id="ProductsCacheRefreshCommand"

    targetNamespace="http://www.redbox.com/OpenAPI/CacheRefreshCommands/v2"

    elementFormDefault="qualified"

    xmlns="http://www.redbox.com/OpenAPI/CacheRefreshCommands/v2"

    xmlns:mstns="http://www.redbox.com/OpenAPI/CacheRefreshCommands/v2"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:include schemaLocation="CacheRefreshCommand.xsd" />

  <xs:complexType name="ProductsCacheRefreshCommand" >

    <xs:complexContent >

      <xs:extension base="mstns:CacheRefreshCommand" >

      </xs:extension>

    </xs:complexContent>

  </xs:complexType>

  <xs:element name="ProductsCacheRefreshCommand" type="ProductsCacheRefreshCommand" />

</xs:schema>

### StoresCacheRefreshCommand Message Schema

<?xml version="1.0" encoding="utf-8"?>

<xs:schema id="StoresCacheRefreshCommand"

    targetNamespace="http://www.redbox.com/OpenAPI/CacheRefreshCommands/v2"

    elementFormDefault="qualified"

    xmlns="http://www.redbox.com/OpenAPI/CacheRefreshCommands/v2"

    xmlns:mstns="http://www.redbox.com/OpenAPI/CacheRefreshCommands/v2"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:include schemaLocation="CacheRefreshCommand.xsd" />

  <xs:complexType name="StoresCacheRefreshCommand" >

    <xs:complexContent >

      <xs:extension base="mstns:CacheRefreshCommand" >

      </xs:extension>

    </xs:complexContent>

  </xs:complexType>

  <xs:element name="StoresCacheRefreshCommand" type="StoresCacheRefreshCommand" />

</xs:schema>

### Top20CacheRefreshCommand Message Schema

<?xml version="1.0" encoding="utf-8"?>

<xs:schema id="Top20CacheRefreshCommand"

    targetNamespace="http://www.redbox.com/OpenAPI/CacheRefreshCommands/v2"

    elementFormDefault="qualified"

    xmlns="http://www.redbox.com/OpenAPI/CacheRefreshCommands/v2"

    xmlns:mstns="http://www.redbox.com/OpenAPI/CacheRefreshCommands/v2"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:include schemaLocation="CacheRefreshCommand.xsd" />

  <xs:complexType name="Top20CacheRefreshCommand" >

    <xs:complexContent >

      <xs:extension base="mstns:CacheRefreshCommand" >

      </xs:extension>

    </xs:complexContent>

  </xs:complexType>

  <xs:element name="Top20CacheRefreshCommand" type="Top20CacheRefreshCommand" />

</xs:schema>

### InventoryCacheRefreshCommandMessage Schema

<?xml version="1.0" encoding="utf-8"?>

<xs:schema id="InventoryCacheRefreshCommand"

    targetNamespace="http://www.redbox.com/OpenAPI/CacheRefreshCommands/v2"

    elementFormDefault="qualified"

    xmlns="http://www.redbox.com/OpenAPI/CacheRefreshCommands/v2"

    xmlns:mstns="http://www.redbox.com/OpenAPI/CacheRefreshCommands/v2"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:include schemaLocation="CacheRefreshCommand.xsd" />

  <xs:complexType name="InventoryCacheRefreshCommand" >

    <xs:complexContent >

      <xs:extension base="mstns:CacheRefreshCommand" >

      </xs:extension>

    </xs:complexContent>

  </xs:complexType>

  <xs:element name="InventoryCacheRefreshCommand" type="InventoryCacheRefreshCommand" />

</xs:schema>

### ItemChangedEventMessage Schema

<?xml version="1.0" encoding="utf-8"?>

<xs:schema id="SourceItemChangedEventMessage"

    targetNamespace="http://api-source.redbox.com/OpenAPI/SourceItemChangeMessages/v2"

    elementFormDefault="qualified"

    xmlns="http://api-source.redbox.com/OpenAPI/SourceItemChangeMessages/v2"

    xmlns:tns="http://api-source.redbox.com/OpenAPI/SourceItemChangeMessages/v2"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:complexType name="SourceItemChangedEventMessage" >

    <xs:attribute name="dateTimeIssued" use="required" type="xs:dateTime" />

    <xs:attribute name="source" type="xs:string" use="required" />

  </xs:complexType>

</xs:schema>

### StoreChangedEventMessage Schema

Example:

<StoreChangedEventMessage lat="1" long="1" storeId="0" storeType="Indoor" dateTimeIssued="1900-01-01T01:01:01" source="source1" xmlns="http://api-source.redbox.com/OpenAPI/SourceItemChangeMessages/v2">

  <KioskLabel>KioskLabel1</KioskLabel>

  <KioskBanner>KioskBanner1</KioskBanner>

  <KioskStatus>Online</KioskStatus>

  <Channel>Channel1</Channel>

  <Address>Address1</Address>

  <Address2>Address21</Address2>

  <City>City1</City>

  <State>State1</State>

  <Zipcode>Zipcode1</Zipcode>

</StoreChangedEventMessage>

Schema:

<?xml version="1.0" encoding="utf-8"?>

<xs:schema id="SourceStoreChangedEventMessage"

    targetNamespace="http://api-source.redbox.com/OpenAPI/SourceItemChangeMessages/v2"

    elementFormDefault="qualified"

    xmlns="http://api-source.redbox.com/OpenAPI/SourceItemChangeMessages/v2"

    xmlns:tns="http://api-source.redbox.com/OpenAPI/SourceItemChangeMessages/v2"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:include schemaLocation="ItemChangedEventMessage.xsd" />

  <xs:simpleType name="KioskType">

    <xs:restriction base="xs:string">

      <xs:enumeration value="Indoor" />

      <xs:enumeration value="Outdoor" />

    </xs:restriction>

  </xs:simpleType>

  <xs:simpleType name="KioskCommStatus">

    <xs:restriction base="xs:string">

      <xs:enumeration value="Online" />

      <xs:enumeration value="Offline" />

    </xs:restriction>

  </xs:simpleType>

  <xs:complexType name="StoreChangedEventMessage" >

    <xs:complexContent >

      <xs:extension base="tns:SourceItemChangedEventMessage" >

        <xs:sequence >

          <xs:element minOccurs="0" maxOccurs="1" name="KioskLabel" type="xs:string" />

          <xs:element minOccurs="0" maxOccurs="1" name="KioskBanner" type="xs:string" />

          <xs:element minOccurs="0" maxOccurs="1" name="KioskStatus" type="tns:KioskCommStatus" />

          <xs:element minOccurs="0" maxOccurs="1" name="Channel" type="xs:string" />

          <xs:element minOccurs="0" maxOccurs="1" name="Address" type="xs:string" />

          <xs:element minOccurs="0" maxOccurs="1" name="Address2" type="xs:string" />

          <xs:element minOccurs="0" maxOccurs="1" name="City" type="xs:string" />

          <xs:element minOccurs="0" maxOccurs="1" name="State" type="xs:string" />

          <xs:element minOccurs="0" maxOccurs="1" name="Zipcode" type="xs:string" />

        </xs:sequence>

        <xs:attribute name="lat" type="xs:decimal" use="required" />

        <xs:attribute name="long" type="xs:decimal" use="required" />

        <xs:attribute name="storeId" type="xs:unsignedInt" use="required" />

        <xs:attribute name="storeType" type="tns:KioskType" use="required" />

      </xs:extension>

    </xs:complexContent>

  </xs:complexType>

  <xs:element name="StoreChangedEventMessage" type="StoreChangedEventMessage" />

</xs:schema>

### InventoryChangedEventMessage Schema

Examples:

<InventoryEventMessage xmlns="http://api-source.redbox.com/OpenAPI/SourceItemChangeMessages/v2">

  <InventoryChangedEventMessage inventoryAmount="0" storeId="0" productNumber="0" dateTimeIssued="1900-01-01T01:01:01" source="source1" />

</InventoryEventMessage>

<InventoryEventMessage xmlns="http://api-source.redbox.com/OpenAPI/SourceItemChangeMessages/v2">

  <InventoryRemovalEventMessage storeId="0" productNumber="0" dateTimeIssued="1900-01-01T01:01:01" source="source1" />

</InventoryEventMessage>

Schema:

<?xml version="1.0" encoding="utf-8"?>

<xs:schema id="SourceInventoryChangedEventMessage"

    targetNamespace="http://api-source.redbox.com/OpenAPI/SourceItemChangeMessages/v2"

    elementFormDefault="qualified"

    xmlns="http://api-source.redbox.com/OpenAPI/SourceItemChangeMessages/v2"

    xmlns:tns="http://api-source.redbox.com/OpenAPI/SourceItemChangeMessages/v2"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:include schemaLocation="ItemChangedEventMessage.xsd" />

  <xs:complexType name="InventoryRemovalEventMessage">

    <xs:complexContent >

      <xs:extension base="tns:SourceItemChangedEventMessage" >

        <xs:attribute name="storeId" type="xs:unsignedInt" use="required" />

        <xs:attribute name="productNumber" type="xs:unsignedInt" use="required" />

      </xs:extension>

    </xs:complexContent>

  </xs:complexType>

  <xs:complexType name="InventoryChangedEventMessage" >

    <xs:complexContent >

      <xs:extension base="tns:SourceItemChangedEventMessage" >

        <xs:attribute name="inventoryAmount" type="xs:unsignedInt" use="required" />

        <xs:attribute name="storeId" type="xs:unsignedInt" use="required" />

        <xs:attribute name="productNumber" type="xs:unsignedInt" use="required" />

      </xs:extension>

    </xs:complexContent>

  </xs:complexType>

  <xs:complexType name="InventoryEventMessage">

    <xs:choice>

      <xs:element name="InventoryChangedEventMessage" type="InventoryChangedEventMessage" />

      <xs:element name="InventoryRemovalEventMessage" type="InventoryRemovalEventMessage" />

    </xs:choice>

  </xs:complexType>

  <xs:element name="InventoryEventMessage" type="InventoryEventMessage" />

</xs:schema>

### ItemChangesMessage Schema

<?xml version="1.0" encoding="utf-8"?>

<xs:schema id="ExternalItemChangesMessage"

    targetNamespace="http://api.redbox.com/OpenAPI/ItemChangeMessages/v2"

    elementFormDefault="qualified"

    xmlns="http://api.redbox.com/OpenAPI/ItemChangeMessages/v2"

    xmlns:tns="http://api.redbox.com/OpenAPI/ItemChangeMessages/v2"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:complexType name="ItemChangesMessageBase" >

    <xs:attribute name="dateTimeIssued" use="required" type="xs:dateTime" />

    <xs:attribute name="source" type="xs:string" use="required" />

  </xs:complexType>

</xs:schema>

### StoreChangesMessage Schema

Example:

<StoreChangesMessage dateTimeIssued="1900-01-01T01:01:01" source="source1" xmlns="http://api.redbox.com/OpenAPI/ItemChangeMessages/v2">

  <StoreChange storeId="storeId1" storeType="Indoor" lastUpdated="1900-01-01T01:01:01">

    <Location lat="1" long="1">

      <Address>Address1</Address>

      <Address2>Address21</Address2>

      <City>City1</City>

      <State>State1</State>

      <Zipcode>Zipcode1</Zipcode>

    </Location>

    <KioskLabel>KioskLabel1</KioskLabel>

    <KioskBanner>KioskBanner1</KioskBanner>

    <KioskStatus>Online</KioskStatus>

    <Channel>Channel1</Channel>

  </StoreChange>

  <StoreRemoval storeId="storeId1" lastUpdated="1900-01-01T01:01:01" />

  <StoreChange storeId="storeId2" storeType="Outdoor" lastUpdated="0001-01-01T00:00:00">

    <Location lat="-79228162514264337593543950335" long="-79228162514264337593543950335">

      <Address>Address2</Address>

      <Address2>Address22</Address2>

      <City>City2</City>

      <State>State2</State>

      <Zipcode>Zipcode2</Zipcode>

    </Location>

    <KioskLabel>KioskLabel2</KioskLabel>

    <KioskBanner>KioskBanner2</KioskBanner>

    <KioskStatus>Offline</KioskStatus>

    <Channel>Channel2</Channel>

  </StoreChange>

</StoreChangesMessage>

Schema:

<?xml version="1.0" encoding="utf-8"?>

<xs:schema id="ExternalStoreChangesMessage"

    targetNamespace="http://api.redbox.com/OpenAPI/ItemChangeMessages/v2"

    elementFormDefault="qualified"

    xmlns="http://api.redbox.com/OpenAPI/ItemChangeMessages/v2"

    xmlns:tns="http://api.redbox.com/OpenAPI/ItemChangeMessages/v2"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:include schemaLocation="ExternalItemChangesMessage.xsd" />

  <xs:simpleType name="KioskTypeEnum">

    <xs:restriction base="xs:string">

      <xs:enumeration value="Indoor" />

      <xs:enumeration value="Outdoor" />

    </xs:restriction>

  </xs:simpleType>

  <xs:simpleType name="KioskCommStatus">

    <xs:restriction base="xs:string">

      <xs:enumeration value="Online" />

      <xs:enumeration value="Offline" />

    </xs:restriction>

  </xs:simpleType>

  <xs:complexType name="StoreRemoval">

    <xs:attribute name="storeId"  type="xs:string" use="required" />

    <xs:attribute name="lastUpdated" type="xs:dateTime" use="required" />

  </xs:complexType>

  <xs:complexType name="StoreChange" >

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="1" name="Location" type="tns:Location" />

      <xs:element minOccurs="0" maxOccurs="1" name="KioskLabel" type="xs:string" />

      <xs:element minOccurs="0" maxOccurs="1" name="KioskBanner" type="xs:string" />

      <xs:element minOccurs="0" maxOccurs="1" name="KioskStatus" type="tns:KioskCommStatus" />

      <xs:element minOccurs="0" maxOccurs="1" name="Channel" type="xs:string" />

    </xs:sequence>

    <xs:attribute name="storeId" type="xs:string" use="required" />

    <xs:attribute name="storeType" type="tns:KioskTypeEnum" use="required" />

    <xs:attribute name="lastUpdated" type="xs:dateTime" use="required"/>

  </xs:complexType>

  <xs:complexType name="Location">

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="1" name="Address" type="xs:string" />

      <xs:element minOccurs="0" maxOccurs="1" name="Address2" type="xs:string" />

      <xs:element minOccurs="0" maxOccurs="1" name="City" type="xs:string" />

      <xs:element minOccurs="0" maxOccurs="1" name="State" type="xs:string" />

      <xs:element minOccurs="0" maxOccurs="1" name="Zipcode" type="xs:string" />

    </xs:sequence>

    <xs:attribute name="lat" type="xs:decimal" use="required" />

    <xs:attribute name="long" type="xs:decimal" use="required" />

  </xs:complexType>

  <xs:complexType name="StoreChanges" >

    <xs:complexContent >

      <xs:extension base="tns:ItemChangesMessageBase" >

        <xs:sequence >

          <xs:choice  minOccurs="1" maxOccurs="unbounded">

            <xs:element name="StoreChange" type="StoreChange" />

            <xs:element name="StoreRemoval" type="StoreRemoval" />

          </xs:choice>

        </xs:sequence>

      </xs:extension>

    </xs:complexContent>

  </xs:complexType>

  <xs:element name="StoreChangesMessage" type="StoreChanges" />

</xs:schema>

### InventoryChangesMessage Schema

Example:

<InventoryChangesMessage dateTimeIssued="1900-01-01T01:01:01" source="source1" xmlns="http://api.redbox.com/OpenAPI/ItemChangeMessages/v2">

  <InventoryRemoval storeId="storeId1" productId="productId1" lastUpdated="1900-01-01T01:01:01" />

  <InventoryChange inventoryAmountCode="inventoryAmountCode1" storeId="storeId1" productId="productId2" lastUpdated="1900-01-01T01:01:01" />

  <InventoryRemoval storeId="storeId2" productId="productId2" lastUpdated="0001-01-01T00:00:00" />

</InventoryChangesMessage>

Schema:

<?xml version="1.0" encoding="utf-8"?>

<xs:schema id="ExternalInventoryChangesMessage"

    targetNamespace="http://api.redbox.com/OpenAPI/ItemChangeMessages/v2"

    elementFormDefault="qualified"

    xmlns="http://api.redbox.com/OpenAPI/ItemChangeMessages/v2"

    xmlns:tns="http://api.redbox.com/OpenAPI/ItemChangeMessages/v2"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:include schemaLocation="ExternalItemChangesMessage.xsd" />

  <xs:complexType name="InventoryChange" >

    <xs:attribute name="inventoryAmountCode" type="xs:string" use="required" />

    <xs:attribute name="storeId"  type="xs:string" use="required" />

    <xs:attribute name="productId"  type="xs:string" use="required" />

    <xs:attribute name="lastUpdated" type="xs:dateTime" use="required"/>

  </xs:complexType>

  <xs:complexType name="InventoryRemoval">

    <xs:attribute name="storeId"  type="xs:string" use="required" />

    <xs:attribute name="productId"  type="xs:string" use="required" />

    <xs:attribute name="lastUpdated" type="xs:dateTime" use="required" />

  </xs:complexType>

  <xs:complexType name="InventoryChangesMessage" >

    <xs:complexContent >

      <xs:extension base="tns:ItemChangesMessageBase" >

        <xs:sequence >

          <xs:choice  minOccurs="1" maxOccurs="unbounded">

            <xs:element name="InventoryRemoval" type="InventoryRemoval"  />

            <xs:element name="InventoryChange"  type="InventoryChange" />

          </xs:choice>

        </xs:sequence>

      </xs:extension>

    </xs:complexContent>

  </xs:complexType>

  <xs:element name="InventoryChangesMessage" type="InventoryChangesMessage" />

</xs:schema>

### MovieTrailers Schema

Example:

<MovieTrailers xmlns="http://api.redbox.com/Products/MovieTrailers/v2">

  <MovieTrailer productId="productId1">

    <Rendition>

      <Url>http://uri1</Url>

      <EncodingRate>1</EncodingRate>

      <FrameHeight>1</FrameHeight>

      <FrameWidth>1</FrameWidth>

      <Size>1</Size>

      <Duration>P396DT1H1M1S</Duration>

      <Codec>Codec1</Codec>

      <Container>Container1</Container>

    </Rendition>

    <Rendition>

      <Url>http://uri2</Url>

      <EncodingRate>-2147483647</EncodingRate>

      <FrameHeight>-2147483647</FrameHeight>

      <FrameWidth>-2147483647</FrameWidth>

      <Size>-9223372036854775807</Size>

      <Duration>-P10675199DT2H48M5.477S</Duration>

      <Codec>Codec2</Codec>

      <Container>Container2</Container>

    </Rendition>

    <Rendition>

      <Url>http://uri3</Url>

      <EncodingRate>2147483647</EncodingRate>

      <FrameHeight>2147483647</FrameHeight>

      <FrameWidth>2147483647</FrameWidth>

      <Size>9223372036854775807</Size>

      <Duration>P10675199DT2H48M5.477S</Duration>

      <Codec>Codec3</Codec>

      <Container>Container3</Container>

    </Rendition>

  </MovieTrailer>

  <MovieTrailer productId="productId2">

    <Rendition>

      <Url>http://uri4</Url>

      <EncodingRate>0</EncodingRate>

      <FrameHeight>0</FrameHeight>

      <FrameWidth>0</FrameWidth>

      <Size>0</Size>

      <Duration>P31DT1H1M1S</Duration>

      <Codec>Codec4</Codec>

      <Container>Container4</Container>

    </Rendition>

    <Rendition>

      <Url>http://uri5</Url>

      <EncodingRate>2</EncodingRate>

      <FrameHeight>2</FrameHeight>

      <FrameWidth>2</FrameWidth>

      <Size>2</Size>

      <Duration>P761DT1H1M1S</Duration>

      <Codec>Codec5</Codec>

      <Container>Container5</Container>

    </Rendition>

    <Rendition>

      <Url>http://uri6</Url>

      <EncodingRate>-2147483646</EncodingRate>

      <FrameHeight>-2147483646</FrameHeight>

      <FrameWidth>-2147483646</FrameWidth>

      <Size>-9223372036854775806</Size>

      <Duration>-P10674834DT2H48M5.477S</Duration>

      <Codec>Codec6</Codec>

      <Container>Container6</Container>

    </Rendition>

  </MovieTrailer>

</MovieTrailers>

Schema:

<?xml version="1.0" encoding="utf-8"?>

<xs:schema id="MovieTrailers"

         xmlns:tns="http://api.redbox.com/Products/MovieTrailers/v2"

         elementFormDefault="qualified"

         targetNamespace="http://api.redbox.com/Products/MovieTrailers/v2"

         xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

  <xs:complexType name="ArrayOfMovieTrailers">

    <xs:sequence>

      <xs:element minOccurs="0" maxOccurs="unbounded" name="MovieTrailer" nillable="false" type="tns:MovieTrailer" />

    </xs:sequence>

  </xs:complexType>

  <xs:complexType name="MovieTrailer">

    <xs:sequence>

      <xs:element minOccurs="1" maxOccurs="unbounded" name="Rendition" type="tns:TrailerRendition" />

    </xs:sequence>

    <xs:attribute name="productId" type="xs:string"  use="required" />

  </xs:complexType>

  <xs:complexType name="TrailerRendition" >

    <xs:sequence >

      <xs:element name="Url" type="xs:anyURI" />

      <xs:element name="EncodingRate" type="xs:int" />

      <xs:element name="FrameHeight" type="xs:int" />

      <xs:element name="FrameWidth" type="xs:int"/>

      <xs:element name="Size" type="xs:long" />

      <xs:element name="Duration" type="xs:duration" />

      <xs:element name="Codec" type="xs:string" />

      <xs:element name="Container" type="xs:string" />

    </xs:sequence>

  </xs:complexType>

  <xs:element name="MovieTrailers" type="tns:ArrayOfMovieTrailers" />

 </xs:schema>

# Change Log

| **Date** | **Version** | **Author** | **Description** |
| --- | --- | --- | --- |
| 1/13/11 | 1.0 | Chris Rudolphi | Initial Draft |
| 1/20/11 | 1.0.1 | Chris Rudolphi | Revised to reflect feedback from Development.   * Revised Products Endpoint URL structure to be /Products/Movies and changed schema of returned data to match (<Products> <Movie>) in anticipation of support for Games * Revised Source Products URL structure to be /Products/Movies and changed schema to match * Moved the Top20 external endpoint to be an operation on the Products endpoint; changed URL structure to be Products/Movies/Top20 * Revised Top20 schemas to embed Top20 information into Movie metadata, rather than embedding Movie metadata inside a Top20 entry. * Added Risk item on the maturity of MQ in the Redbox environment. |
| 2/4/11 | 1.02 |  | Corrected execution frequency of Products Service Refresh. |
| 2/4/11 | 2.0 Draft 2 | Chris Rudolphi | First draft of Store Search, Inventory, and Reservation services. |
| 2/11/11 | 2.0 Draft 4 | Chris Rudolphi | Revisions to Stores, Inventory & Reservations to fix issues found in review by Architecture, Dev. |
| 3/16/11 | 2.0 Draft 8 | Chris Rudolphi | First complete draft; split into two separate documents. |
| 3/28/11 | 2.0 Draft 9 | Chris Rudolphi | Corrected Source & External Top20 schemas to include a ‘NoChange’ directional enumeration value.  Corrected all schema samples that use the ‘lastUpdated’ attribute to eliminate the use of the “-06:00” suffix as all times will be rendered as GMT and thus no suffix is required.  Corrected MPAARatings to include an enumeration value of “NR” for use when a product has no MPAARating value.  Revised the internal and external ItemChangeMessage schemas, and CacheRefreshCommand message schemas to make the namespaces more distinct.  Revised structure of ItemChangeMessage schemas to include the concept of an item being removed (for when Stores are removed and when a Product is no longer kept in Inventory at a Store).  Corrected the Titles schema to include a Titles element for each ProductRef. This will be useful when the titles of each Product vary from the title given the collapsed title-product.  Marked as ‘Out-of-scope’ the Item-changed sequence for Stores. |
| 4/8/11 | 2.0 Draft 10 | Chris Rudolphi | Finalized how Trailers would be supported:   * Added Trailer Gateway that will call BrightCove when Products are bulk refreshed * Storage of bulk trailer metadata in Trailer cache at Apigee Layer * Added explicit Products Endpoint operation to retrieve trailer metadata for a given product.   Corrected description of how Inventory Gateway works; deleted mention of a cache of Inventory data in the gateway.  Fixed table in Customer Profile section 2.7.6 that list the CP schemas.  Corrected mention of use of SAML token to a SWT token for customer authentication. |
| 4/27/2011 | 2.0 Draft 11 | Chris Rudolphi | Fixed confusing names of references to Schema 6.1.3 in the Products operation descriptions. They all now refer to ‘ProductList’.  Corrected Products Source schema by making RunningLength datatype to xs:duration.  Corrected MPAARating valid values to be PG-13 from PG13 (all Products,Top20 and Cart schemas).  Revised Source Inventory schema to move the timestamp from the Store level to the ProductInventory level. Each ProductInventory item now has its own timestamp. |
| 5/5/11 | 2.0 Draft 12 | Chris Rudolphi | Fixed Titles sample to show that TitleIdentifier source attribute will always be “Baseline”.  Added lastUpdated timestamp to the <Titles> element.  Modified the CartValidation schema to include a “Cart” as an element. This supports returning a validated/modified Cart object from Apigee when operations R1 – R4 are called.  Corrected an error in the detailed description of CP19 (Create Customer) to reflect that the operation returns a new CustomerNumber, not an AccountNumber.  Eliminated Reservation operation R4 (Apply Discount) and renumbered R5 and R6 to be R4 and R5 respectively. Apply Discount was eliminated because there is no corresponding Credits or Pipeline operation. Discounts that are applicable to a cart are identified by the Pipeline when the Cart is priced.  Eliminated use of schema 6.1.31 (Credits Balance) because it provides only a single number as an answer, which no longer makes sense given the diversity of credit types which can be applied. References to 6.1.31 are replaced with schema 6.1.32 (Available Credits) which provides a detailed listing. |
|  | 2.0 Draft 13 | Chris Rudolphi | Added RedboxComingSoon date as xs:date field of the Products schemas. This date value will be calculated by OpenServices layer based upon ComingSoonDays value from EPC (RedboxComingSoonDate = RedboxReleaseDate – ComingSoonDays)  Added Status Monitoring operations to each Apigee Endpoint and internal Gateway. These will be called by Sitescope. |