****

**Redbox**

OpenAPI – Release 10

Architecture Design

Part II – Design

**Author**: *Chris Rudolphi*

**Document Title**: OpenAPI R10 Architecture Design v1 DRAFT 6

**Updated**: *4/10/2012 1:55 PM*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 |  |  |
| Scott Sisil | Product Manager | Reviewer |  |  |
| Viju Hullur | SA for OpenAPI | Reviewer |  |  |
| Imran Maskatia | Digital Product Manager | Reviewer |  |  |

[1. Introduction 6](#_Toc321734088)

[1.1 Project Description 6](#_Toc321734089)

[1.2 Description and Scope of this Document 6](#_Toc321734090)

[1.3 New In the Design of Release 6 6](#_Toc321734091)

[2. Design 7](#_Toc321734092)

[2.1 Products Service 7](#_Toc321734093)

[2.2 Stores Service 11](#_Toc321734094)

[2.3 Inventory Service 13](#_Toc321734095)

[2.4 Reservation Service 15](#_Toc321734096)

[2.5 Customer Account Service 19](#_Toc321734097)

[2.6 Major Solution Components 23](#_Toc321734098)

[2.6.1 Products Gateway 23](#_Toc321734099)

[2.6.2 Kiosk (“Stores”) Gateway 25](#_Toc321734100)

[2.6.3 Inventory Gateway 26](#_Toc321734101)

[2.6.4 Reservations Gateway 27](#_Toc321734102)

[2.6.5 Customer Account Gateway 29](#_Toc321734103)

[2.6.6 CreditCard Services Gateway 32](#_Toc321734104)

[2.6.7 Trailers Gateway 33](#_Toc321734105)

[2.6.8 Data Sources 33](#_Toc321734106)

[2.6.9 Cache Refresh Batch Jobs 34](#_Toc321734107)

[2.6.10 Change Notifiers (Out of Scope for R2 & R3) 34](#_Toc321734108)

[2.6.11 Source System Change Service 35](#_Toc321734109)

[2.6.12 Cache Refresh Message Handlers 35](#_Toc321734110)

[2.6.13 Item Changed Message Handlers *(Out of Scope for R2 & R3)* 35](#_Toc321734111)

[2.6.14 Apigee Components 36](#_Toc321734112)

[2.6.15 EPC Web Service 81](#_Toc321734113)

[2.6.16 Batch Scheduler 81](#_Toc321734114)

[2.6.17 Web Reservation Pipeline 81](#_Toc321734115)

[2.6.18 Caching Services 81](#_Toc321734116)

[2.6.19 Web Services Monitoring 81](#_Toc321734117)

[2.7 Physical Deployment 82](#_Toc321734118)

[2.8 Database Logical Design 84](#_Toc321734119)

[2.9 Open Systems Layer: Infrastructure Components & Dependencies 85](#_Toc321734120)

[2.9.1 Custom-built .NET Web Services (Product, Store, Top20 proxies) 85](#_Toc321734121)

[2.9.2 Source System Change Service 85](#_Toc321734122)

[2.9.3 Logging & Instrumentation 85](#_Toc321734123)

[2.10 Design Approaches to Non-Functional Qualities (availability, performance, scaling) 87](#_Toc321734124)

[2.10.1 Availability 87](#_Toc321734125)

[2.10.2 Performance 87](#_Toc321734126)

[2.10.3 Scale 87](#_Toc321734127)

[2.11 Security 88](#_Toc321734128)

[2.11.1 OpenAPI Service Endpoint Security 88](#_Toc321734129)

[2.11.2 Open Services Layer – Source System Gateway Security 88](#_Toc321734130)

[2.12 Dependencies & Assumptions 89](#_Toc321734131)

[3. Open Issues 90](#_Toc321734132)

[4. Risks 92](#_Toc321734133)

[5. Glossary 93](#_Toc321734134)

[6. Appendices 94](#_Toc321734135)

[6.1 Appendix A: Message Contracts & Schemas 94](#_Toc321734136)

[6.1.1 Common Schemas 94](#_Toc321734137)

[6.1.2 Source Movies Schema 96](#_Toc321734138)

[6.1.3 External Products Schema 100](#_Toc321734139)

[6.1.4 External Products Browse Request Schema 104](#_Toc321734140)

[6.1.5 Product Search Request 105](#_Toc321734141)

[6.1.6 Paginated Product List 107](#_Toc321734142)

[6.1.7 Titles 110](#_Toc321734143)

[6.1.8 RecommendedProductsResponse 115](#_Toc321734144)

[6.1.9 Source Stores Schema 116](#_Toc321734145)

[6.1.10 External Stores Schema 118](#_Toc321734146)

[6.1.11 Common External Store Schema 120](#_Toc321734147)

[6.1.12 Source Inventory List 121](#_Toc321734148)

[6.1.13 External Store Lookup Response Schema 122](#_Toc321734149)

[6.1.14 InventoryLookup Schema 124](#_Toc321734150)

[6.1.15 Source Top20 Schema 126](#_Toc321734151)

[6.1.16 Store and Inventory Search Result Schema 127](#_Toc321734152)

[6.1.17 External Top20 Schema 129](#_Toc321734153)

[6.1.18 Cart Schema 132](#_Toc321734154)

[6.1.19 CartValidation Schema 134](#_Toc321734155)

[6.1.20 PricedCart Schema 136](#_Toc321734156)

[6.1.21 PricedCartResponse Schema 139](#_Toc321734157)

[6.1.22 ReservationResponse Schema 141](#_Toc321734158)

[6.1.23 BasicProfile Schema 143](#_Toc321734159)

[6.1.24 Preferences 145](#_Toc321734160)

[6.1.25 RentalHistory 148](#_Toc321734161)

[6.1.26 QueuesListResponse 150](#_Toc321734162)

[6.1.27 QueueInfo 151](#_Toc321734163)

[6.1.28 RecommendedProductsResponse 152](#_Toc321734164)

[6.1.29 AccountListResponse 153](#_Toc321734165)

[6.1.30 AccountInfo 155](#_Toc321734166)

[6.1.31 CreditBalance 156](#_Toc321734167)

[6.1.32 CreditsAvailable 157](#_Toc321734168)

[6.1.33 CreditUseHistory 159](#_Toc321734169)

[6.1.34 CreditCardInfo 161](#_Toc321734170)

[6.1.35 CacheRefreshCommand Message Schema 162](#_Toc321734171)

[6.1.36 ProductsCacheRefreshCommand Message Schema 163](#_Toc321734172)

[6.1.37 StoresCacheRefreshCommand Message Schema 164](#_Toc321734173)

[6.1.38 Top20CacheRefreshCommand Message Schema 165](#_Toc321734174)

[6.1.39 InventoryCacheRefreshCommandMessage Schema 166](#_Toc321734175)

[6.1.40 ItemChangedEventMessage Schema 167](#_Toc321734176)

[6.1.41 StoreChangedEventMessage Schema 168](#_Toc321734177)

[6.1.42 InventoryChangedEventMessage Schema 170](#_Toc321734178)

[6.1.43 ItemChangesMessage Schema 171](#_Toc321734179)

[6.1.44 StoreChangesMessage Schema 172](#_Toc321734180)

[6.1.45 InventoryChangesMessage Schema 174](#_Toc321734181)

[6.1.46 MovieTrailers Schema 176](#_Toc321734182)

[6.1.47 Customer Identity Schema 178](#_Toc321734183)

[6.1.48 New Customer Creation Response Schema 179](#_Toc321734184)

[6.1.49 V3 External Products Schema 180](#_Toc321734185)

[6.1.50 V3 External Top20 Schema 182](#_Toc321734186)

[6.1.51 ProductRatings Schema 186](#_Toc321734187)

[6.1.52 Product Metrics List Schema 187](#_Toc321734188)

[6.1.53 Invoice 188](#_Toc321734189)

[6.1.54 Partner Preferences 191](#_Toc321734190)

[6.1.55 Partner Profile 194](#_Toc321734191)

[6.1.56 AssignPartnerRelationshipRequest 196](#_Toc321734192)

[6.1.57 AssignPartnerRelationshipResponse 197](#_Toc321734193)

[6.1.58 LoginChange 198](#_Toc321734194)

[6.1.59 PasswordChange 199](#_Toc321734195)

[6.1.60 Password Confirmation Request 200](#_Toc321734196)

[6.1.61 Notification Preferences 202](#_Toc321734197)

[6.1.62 PasswordResetRequest 203](#_Toc321734198)

[6.1.63 PasswordConfirmationResponseToken 204](#_Toc321734199)

[6.1.64 CreditCardVerificationRequest 205](#_Toc321734200)

[6.2 Schema Files 206](#_Toc321734201)

[7. Change Log 207](#_Toc321734202)

# 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

## New In the Design of Release 6

Implementation of the v3 Products design (originally included in the R3 design doc but eventually cut from scope of R3), which includes Games and re-design of the Apigee caching approach for Product metadata.

New: product metadata includes “Flags” that specify the start- and end-dates of key business windows for Movie and Game products.

New: product metadata includes “Metrics”; an extensible mechanism for Redbox to share business metrics about products with affiliates. The first set of metrics will include “Popularity” metrics. These can be considered an expansion of the concept of “Top20” to include the popularity ranking of all products.

# Design

## Products Service

The Products Service will provide information about Movies and Games in the Redbox catalog, as depicted in Figure 2.1‑1 Products Service**Error! Reference source not found.**. 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 (operations shown in grey text are out-of-scope):

|  |  |  |  |
| --- | --- | --- | --- |
| **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 | Movie metadata (Bulk list, younger than 250 days with ‘Display On Website’ set) |
| P2 | Retrieves a bulk listing of all Movie product metadata | None | Metadata for all products |
| 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 |
| P10 | Retrieve a Top20 list | Period (7 or 30) | Top20List |
| P13 | Retrieve Trailer Information | Product ID | Trailer metadata |
| P15 | Retrieve a bulk listing of all ‘current’ Game product metadata. |  | Products (Bulk list, younger than 250 days with ‘Display On Website’ set) (aka, the ‘default’ set) |
| P16 | Retrieves a bulk listing of all Game product metadata | None | Metadata for all games |
| P17 | Retrieve movie product metrics (popularity, sales volume, etc) | None | ProductMetricsList |
| P18 | Retrieve game product metrics | None | ProductMetricsList |

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. Apigee will cache product information in different caches to accommodate the various sub-sets and sort orders of data that is served by the Products operations.

The operations will use the caches in the following way:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cache #** | **Cached Data Type** | **Cached Data Subset** | **Sorting**  **(RRD = RedboxReleaseDate)** | **To be used in Operation(s)** |
| 1 | Movies | Default | RRD desc | P3-1 /movies/default?includecomingsoon=false |
| 2 | Movies | ALL Movies | RRD desc | P3-2 /movies |
| 3 | Movies | ComingSoon | RRD asc | P3-3 /movies/comingsoon |
| 4 | Games | ALL Games | RRD desc | P3-16 /games |
| 5 | Games | Default | RRD desc | P3-15 /games/default?includecomingsoon=false |
| 6 | Titles | Default | RRD desc | P3-7 /titles/default |
| 7 | Titles | ComingSoon | RRD asc | P3-8 /titles/comingsoon |
| 8 | Top20 | 7 Day | Position asc | P3-10 /movies/top20?p=7 |
| 9 | Top20 | 30 Day | Position asc | P3-10 /movies/top20?p=30 |
| 10 | Trailers | ALL Trailer data | RRD desc | P3-13 /products/{ID}/trailers |
| 11 | Products | ALL products | RRD desc | P3-4 /products?productIDs={ids,}  P3-5 /products/{ID}  P3-6 (Search) /products? where sortField = RRD |
| 12 | Products | ALL products | Title asc | P3-6 (Search) /products?  where sortField = Title |
| 13 | Movie Metrics | ALL Movie |  | P3-17 /products/movies/metrics |
| 14 | Game Metrics | ALL Game |  | P3-18 /products/games/metrics |
| 15 | Movies | Default WITH ComingSoon | RRD desc | P3-15 /movies/default?includecomingsoon=true |
| 16 | Games | Default WITH ComingSoon | RRD desc | P3-16 /games/default?includecomingsoon=true |

The following Products Cache operations will cause cache refreshes to occur:

|  |  |
| --- | --- |
| **ID** | **Description** |
| PC1 | Refresh Movies-Default |
| PC2 | Refresh Movies-ALL |
| PC3 | Refresh ComingSoon Movies |
| PC4 | Refresh Games-ALL |
| PC5 | Refresh Games – default browsable list |
| PC6 | Refresh Titles – default browsable list |
| PC7 | Refresh Titles – Coming Soon |
| PC8 | Refresh 7-day Top20 |
| PC9 | Refresh 30-day Top20 |
| PC10 | Refresh Trailer Information |
| PC11 | Refresh Products in RedboxReleaseDate sort order |
| PC12 | Refresh Products in Title sort order |
| PC13 | Refresh Products Movie Metrics |
| PC14 | Refresh Products Games Metrics |
| PC15 | Refresh Movies-Default WITH comingsoon |
| PC16 | Refresh Games-Default WITH comingsoon |



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 2.7.1.1)
* The Inventory Gateway (see 2.7.4 )
* The Stores Cache Refresh Job (see 2.7.10.2)
* The Stores Cache Refresh Message Handler (see 2.7.13.3)
* 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
* Store Search by ZipCode

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 |
| S6 | Retrieve paged Store metadata given a ZipCode. | ZipCode;  Radius to search;  Count of number of Stores to return;  Banner name to filter;  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:

* 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 (2.7.1.1)



Figure ‑ Stores Service

## 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 Gateway (described further in 2.7.4)
* The Inventory Cache Refresh Job (see 2.7.10.4)
* The Inventory Cache Refresh Message Handler (see 2.7.13.5)
* The Inventory Endpoint Service in the Apigee Layer (see 2.7.15.1.3)

Inventory data changes frequently. After consultation with Apigee, it has been deterimined that the optimal mechanism to use to keep Apigee updated with inventory data is to partition Inventory updates by Store and to send complete Store inventory updates on a periodic basis. Apigee will implement the Inventory service as a pass-through proxy and will cache Store-level inventory for a short period of time (configurable).

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 single store. | Store Id for which Inventory is to be retrieved;  List of Product IDs for which Inventory status is to be retrieved (optional);  Page Number and Page Size | Inventory Lookup |
| I5 | Retrieve a paged listing of Inventory for a set of stores by Lat/Long location. | Lat/Long location for which Inventory is to be retrieved;  List of Product IDs for which Inventory status is to be retrieved (optional);  Page Number and Page Size;  Radius to search;  Count of number of Stores to return;  Banner name to filter;  Sorting options | Inventory Lookup |
| I6 | Retrieve a paged listing of Inventory for a set of stores by PostalCode. | ZipCode for stores for which Inventory is to be retrieved;  List of Product IDs for which Inventory status is to be retrieved (optional);  Page Number and Page Size;  Radius to search;  Count of number of Stores to return;  Banner name to filter;  Sorting options | Inventory Lookup |

The output of the first operation, an inventory lookup, is a list 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.



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 | Validates a user's cart. The body of the message contains a Cart to be validated. | Validated by Apigee Layer and Reservation Gateway | CartValidation |
| R2 | Deprecated – not used. | Validated by Apigee Layer | CartValidation |
| R3 | Deprecated – not used. | 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 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) |
| Rental Invoices | Invoice & Payment Information | Merchant Services |
| 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* |
| Partner Profile | Varies by partner | CustomerProfile Service |
| Partner Preferences | Digital program customer preferences | CustomerProfile 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).

*Note: in the following table the term “Customer Identifier” is used in the INPUT column. A Customer Identifier is any form of unique identifier that identifies a customer. Two forms are possible: a CustomerNumber and a PartnerCustomerIdentifier (which will take the form of a VendorCode+CustomerParnerNumber).*

| **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 | Customer Identifier | Basic Profile |
| CP2 | Saves the basic profile | Customer Identifier BasicProfile | None |
| CP3 | Retrieve all preference data | Customer Identifier | Preferences |
| CP4 | Saves preferences | Customer Identifier Preferences |  |
| CP5 | Retrieve rental history | Customer Identifier | Rental History info |
| CP6 | Retrieve the customer’s list of Queues | Customer Identifier | List of Queue Info |
| CP7 | Retrieve a single Queue | Customer Identifier QueueId | Queue Info (ordered list of Product Ids) |
| CP8 | Replace a Queue | Customer Identifier QueueId  Queue Info |  |
| CP9 | Delete a Queue | Customer Identifier QueueId |  |
| CP10 | Add to or Move a Product in a Queue to the given position | Customer Identifier QueueId  Position number  ProductId | Queue Info |
| CP11 | Remove an item from a Queue | Customer Identifier QueueId  ProductId | Queue Info |
| CP12 | Retrieve recommendations | Customer Identifier ProductType  ProductId  UseRentalHistory flag | List of Product Ids |
| CP13 | Retrieve the customer’s accounts | Customer Identifier | List of Account Info objects |
| CP14 | Retrieve a single account | Customer Identifier AccountNumber | Account Info |
| CP15 | Updates attributes of an account | Customer Identifier AccountNumber  AccountInfo |  |
| CP16 | Retrieve the Rental Credits balance for a customer | Customer Identifier | Credits Balances |
| CP17 | Retrieve the Rental Credits history for a customer | Customer Identifier | Credits History |
| CP18 | Add a CreditCard | Customer Identifier CreditCard | Account Info |
| CP19 | Create a New Customer | Email Address | CustomerNumber |
| CP20 | Retrieve Customer Partner Profile | Customer Identifier | PartnerProfile |
| CP21 | Assign CustomerPartnerNumber to Customer | CustomerNumber  PartnerId | CustomerPartnerNumber |
| CP22 | Retrieve Customer Partner Preferences | Customer Identifier PartnerId | Customer Partner Preferences |
| CP23 | Update Customer Partner Preferences | Customer Identifier PartnerId  Partner Preferences | Customer Partner Preferences |
| CP24 | Retrieve Customer Invoice | Customer Identifier TransactionNumber | Invoice |
| CP25 | Change Login Information | Customer Identifier LoginChange | Customer Login Information |
| CP26 | Reset Password | Email | status |
| CP27 | Change Password | Customer Identifier  PasswordChange | status |
| CP28 | Delete a credit card account from a customer’s account | Customer Identifier  AccountNumber | status |
| CP29 | Password Confirmation | Customer Identifier  Password | PasswordConfirmationResponseToken |
| CP30 | – not used - |  |  |
| CP31 | Update Customer Partner Profile | Customer Identifier  Partner Profile |  |
| CP32 | Verify a credit card account (AVS Check and CVV Check) | Customer Identifier  Partner ID  AccountNumber | Status |
| CP33 | Set Account for Recurring Partner Billing | Customer Identifier  Partner ID  AccountNumber | status |

## Major Solution Components

New Components

The following components will be built for OpenAPI:

### Products Gateway

This will be a 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 information about Products (Movies and/or Games) and Titles. The output formats are described in schemas 6.1.3 and 6.1.7 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 (operations in grey text are out-of-scope):

* /products/fullRefresh
  + Input: None
  + Output: None
  + Description: Invoked by the **Products Cache Refresh Command Handler**, this method will refresh internal Products caches in the Open Services Layer in preparation for handling cache refresh calls from the Apigee layer.
* /products/statusmonitor
  + Input: None
  + Output: Status
  + Description: Confirm connectivity with EPC and report back status.
* /products/movies/default?includecomingsoon={icsFlag} (PC1 and PC15)
  + Input:

icsFlag: indicator of whether to include the coming soon data (ie, contents of PC1 or PC15)

* + Output: Products, see schema V3 External Products Schema6.1.49
  + Description: Retrieve all Movies from EPC that fit within the “Default” window (250-days or less) (do NOT include coming soon)
* /products/movies (PC2)
  + Input: None
  + Output: Products, see schema V3 External Products Schema6.1.49
  + Description: Retrieve all Movies from EPC
* /products/movies/comingsoon (PC3)
  + Input: None
  + Output: Products, see schema V3 External Products Schema6.1.49
  + Description: Retrieve all Movies from EPC that are not yet released but fit within the “ComingSoon” window
* /products/games (PC4)
  + Input: None
  + Output: Products, see schema V3 External Products Schema6.1.49
  + Description: Retrieve all Games from EPC
* /products/games/default?includecomingsoon={icsFlag} (PC5 & PC16)
  + Input:

icsFlag: indicator of whether to include the coming soon data (ie, contents of PC5 or PC16)

* + Output: Products, see schema V3 External Products Schema6.1.49
  + Description: Retrieve all Games from EPC that are fit within the “Default” window without ComingSoon
* /Titles/Default (PC6)
  + Input: None
  + Output: Titles, see schema 6.1.7
  + Description: Retrieve Titles representation of Movies from EPC that are fit within the “Default” window
* /Titles/ComingSoon (PC7)
  + Input: None
  + Output: Titles, see schema 6.1.7
  + Description: Retrieve Titles representation of Movies from EPC that are Coming Soon
* /products/movies/top20?period=p (PC8 and PC9)
  + Input: Period
  + Output: External Top20, see schema 6.1.17
  + Description: Retrieve Top20 of Movies from EPC
* /products?sortField=fieldName (PC11 & PC12)
  + Input: SortField
  + Output: Products, see schema V3 External Products Schema6.1.49
  + Description: Retrieve all Products from EPC and sort on the given field (RedboxReleaseDate or Title)
* /products/movies/metrics (PC13)
  + Input: None
  + Output: ProductsMetrics, see schema 6.1.526.1.3
  + Description: Retrieve Product Movie Metrics from ODS
* /products/games/metrics (PC14)
  + Input: None
  + Output: ProductsMetrics, see schema 6.1.3
  + Description: Retrieve Product Games Metrics from ODS
* /products/{productId}/similar
  + Input: productId
  + Output: RecommendedProductResponse, see schema 6.1.8 in Appendix A
  + Description: Uses the same logic as the RB 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. The response should set the productType attribute to be the productType of the product from which the recommendations were generated and set the output productId attribute to the input value. The output attribute “basedUponRentalHistory” should be set to ‘false’.

#### 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:

* (S1): /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**.
* ~~/Stores/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.~~
* /stores/storechanged *(Out of scope)*
  + 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.
* (S0): /stores/statusmonitor
  + Input: None
  + Output: Status
  + Description: Confirm connectivity with ODS and report back status.
* (S6): /stores/postalcode/{zipCode}
  + Input: ZipCode
  + Output: StoreLookupList (schema: 6.1.13)
  + Description: Return 50 stores closest to the given zipcode, sorted by ascending distance

**S1: /stores::** This operation will retrieve Store metadata from the ODS, transform the structure of the data to fit the Store Source Format and return it to Apigee. It will store in AppFabric two ID maps: one that maps from internal-to-external storeID format, and one that maps from Externa to Internal ID format. It will also iterate through all AppFabric entries created by operation S6 and invalidate/delete them.

**S6: /stores/postalcode/{zipCode}:** This operation will lookup in the AppFabric cache by ZipCode to find the nearest stores and return to Apigee in external StoreLookupList schema format. If a storelist is not found in AppFabric it will be computed by finding the 50 stores nearest the zipcode, sort them by distance, and store in AppFabric; keyed by zipcode.

### 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.

The ODS will periodically extract Inventory information from the Redbox production database. When an extract is completed, the ODS will invoke **the Inventory Cache Refresh Job**, thus notifiying the OpenAPI that new inventory data is available. The **Inventory Cache Refresh Command Handler** will invoke the Inventory Gateway’s FullRefresh operation.

The FullRefresh operation will:

1. Read the Inventory data from ODS
2. Split it up by Store
3. Save each Store Inventory into AppFabric Cache (keyed by external Store ID)
4. For each Store, invoke the Apigee Inventory endpoint and cause it to refresh its copy of that Store’s inventory

Step d will be accomplished by using an HTTP GET to the Apigee Inventory operation along with a query string parameter that forces the Inventory Endpoint to call back to the Inventory Gateway to pull a fresh copy of the Inventory information for the given store.

To the extent practical, steps c & d should be parallelized. {Suggest using Microsoft’s Task Parallel Library (TPL) as the basis for the development}

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 6.1.14 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/stores/{StoreId}
  + Input: StoreId – external StoreId of the store for which inventory data should be retrieved from the AppFabric Cache.
  + Output: Inventory List; see schema 6.1.140 in Appendix A
  + Description: Called by the Apigee layer, this operation responds to a request for a single Store’s worth of Inventory.
* /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.
* /inventory/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.

The customer is identified in the cart schemas as a ‘User ID’. This can be either the CustomerNumber of the user or the concatenation of a VendorCode and CustomerPartnerNumber (aka {VendorCode}.{CustomerPartnerNumber} )

Operations:

* /cart/price
  + Input: Cart (see schema 6.1.18)
  + Output: PricedCartResponse (see schema 6.1.21)
  + Description: Called by the Apigee layer, this operation calls the Pipeline service to execute the CartView pipeline (which validates and prices the cart).
* /reservations
  + Input: Cart (see schema 6.1.18)
  + 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).
* /reservations/statusmonitor
  + Input: None
  + Output: Status
  + Description: Confirm connectivity with the Reservation Pipeline & Customer Profile Service and report back status.
* ~~/reservations /validate~~ 
  + ~~Method: POST~~
  + ~~Input :~~

~~Cart (see schema: 6.1.18)~~

* + ~~Output: CartValidation (see schema: 6.1.19)~~
  + ~~Description: Reservation Gateway will validate the following:~~

~~The UserID, if provided, in the cart corresponds to an existing, active, customer;~~

~~That each ProductID in the cart is a valid ProductID;~~

~~That the number of items in the cart passes preliminary business rule checks;~~

~~That the StoreID, if provided, is a valid StoreID;~~

~~That the cardID, if provided, corresponds to an Account with the current customer.~~

### 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:

Note: In this table the URL structures are prefixed with either:

**/customers/{CustomerNumber}**

**/customers/{VendorCode}.{CustomerPartnerNumber}**

| **API ID** | **HTTP Method** | **URL structure** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| CP0 | GET | /customers/statusmonitor *(does not use the URL prefix noted above)* | 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}&basedUponProductId={pId}&includeUserHistory={true/false}&numRecommendations={maxRecomm} | CustomerNumber  ProductType  ProductId  IncludeUserHistory Flag  Max Number of Recommendation | Recommended 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  CVV |  |
| CP16 | GET | /credits/balance | CustomerNumber | Credits Balances |
| CP17 | GET | /credits/history | CustomerNumber | Credits History |
| CP19 | POST | /customers *{does not use the /Customers/{CustomerNumber} prefix}* | CustomerIdentity | Customer Number |
| CP20 | GET | /partner/profile?partnerId={PartnerId} | CustomerNumber  PartnerId | CustomerPartnerProfile |
| CP21 | POST | /partner/relationship?partnerId={PartnerId} | CustomerNumber  PartnerId  AssignPartnerRelationshipRequest | AssignPartnerRelationshipResponse |
| CP22 | GET | /partner/preferences?partnerId={PartnerId} | CustomerNumber  PartnerId | CustomerPartnerPreferences |
| CP23 | PUT | /partner/preferences?partnerId={PartnerId} | CustomerNumber  PartnerId  CustomerPartnerPreferences | CustomerPartnerPreferences |
| CP24 | GET | /invoices/{TransactionNumber} | CustomerNumber  TransactionNumber | Invoice |
| CP25 | PUT | /identity | CustomerNumber  LoginChange | 200: Success  400: RequiredField validation Failure  500: Other Error |
| CP26 | POST | /customers/identity/resetpassword  (without use of the /customers/{id}/ prefix) | PasswordResetRequest | 200: Success  400: Address Not found  500: Other Error |
| CP27 | PUT | /identity/password | CustomerNumber  PasswordChange | 200: Success  400: RequiredField validation Failure  500: Other Error |
| CP28 | DELETE | /accounts/{AccountNumber} | CustomerNumber  AccountNumber | 200: Success  400: AccountNotFound  500: Other Error |
| CP29 | POST | /identity/password/confirm | CustomerNumber  PasswordConfirmationRequest | PasswordConfirmationResponseToken |
| CP30 | POST | -not used - |  |  |
| CP31 | PUT | /partner/profile?partnerId={PartnerId} | CustomerNumber  PartnerId  PartnerProfile | 200: Success  400: RequiredField validation Failure  500: Other Error |
| CP32 | POST | /accounts/{AccountNumber}/verify | CustomerNumber  AccountNumber  CVVCode | Status  200: Success with no payload when verification succeeds  200: with Error payload when card fails verification  400: validation error or missing data  500: Other error |
| CP33 | GET | /accounts/{acccountNumber}/subscriptionbilling?partnerId={partnerID} | CustomerNumber  AccountNumber  PartnerId | 200: Success  400: Validation Failure  500: Other Error |

The data contracts used and provided by this service include:

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Contract** | **Output From** | **Input To** | **Schema Location** |
| Basic Profile | CP1 | CP2 | 0 |
| Preferences | CP3 | CP4 | 0 |
| RentalHistory | CP5 |  | 6.1.25 |
| QueuesListResponse | CP6 |  | 0 |
| QueueInfo | CP7, CP10, CP11 | CP8 | 6.1.27 |
| RecommendedProductsResponse | CP12 |  | 6.1.8 |
| AccountListResponse | CP13 |  | 6.1.29 |
| AccountInfo | CP14 | CP15 | 6.1.30 |
| CreditBalances | CP16 |  | 6.1.32 |
| CreditUseHistory | CP17 |  | 6.1.33 |
| CustomerPartnerProfile | CP20 |  | 6.1.55 |
| AssignPartnerRelationshipRequest |  | CP21 | 6.1.56 |
| AssignPartnerRelationshipResponse | CP21 |  | 6.1.57 |
| Preferences\_Digital | CP22 | CP23 | 6.1.54 |
| Invoice | CP24 |  | 6.1.53 |
| CustomerIdentity |  | CP19 | 6.1.47 |
| PasswordResetRequest |  | CP26 | 6.1.62 |
| LoginChange |  | CP25 | 6.1.58 |
| PasswordChange |  | CP27 | 6.1.59 |
| PasswordConfirmationRequest |  | CP29 | 6.1.60 |
| PasswordConfrmationResponseToken | CP29 |  | 6.1.63 |

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 /Customers. 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  CVV | 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.

If the calling client includes a CVV code, then (before associating the card account to the user’s profile) this operation will also invoke the card verification logic (same logic as used with CP32). If the

Verification succeeds, the card will be allowed to be associated to the user profile (if so requested). If the card verification fails, the error code/message output should provide the successfully added card Account Number, but indicate that verification has failed.

### 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:

* /Trailers/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 cacheRefresh=’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).
* /Trailers/Movies
  + 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.

##### ZipCode Location Table

(Assumption for R8): ODS will be enhanced with a table that contains lat/long location information for each known zipcode in the US. This data will be updated manually as the source data is updated. The data will be read by OpenServices when a store search by zipcode is requested. OpenServices will search the ODS by zipcode to find the lat/long and this location will be used to find the nearest kiosks to that location.



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**.

#### **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 (Out of Scope for R2 & R3)

#### **Store Change Notifier**

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 WCF Web service (hosted in IIS). 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 (R2)**

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.

#### **Products Cache Refresh Message Handler (R3)**

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 (for R2 caches) and will directly invoke the Apigee Gateway to cause refresh of the 12 R3 Products caches.

#### **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 *(Out of Scope for R2 & R3)*

#### **Store ChangedEventMessage Handler**

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.

***Operations with a ‘strike-through’ font are out-of-scope.***

| **ID** | **HTTP Method** | **URL structure** | **Input** | **Output** |
| --- | --- | --- | --- | --- |
| P0 | GET | /products/statusMonitor | None | Combined Status of Products, Top20 and Trailers gateways. |
| **V3 Operations** | | | | |
| P3-1 | GET | /v3/products/movies/default?pageNum={pN}&pageSize={pS}&includeComingSoon={true/false} | includeComingSoon flag  Page Number  Page Size | Products (Bulk list, younger than 250 days with ‘Display On Website’ set); will include coming soon products when the ‘includeComingSoon’ flag is set to true |
| P3-2 | GET | /v3/products/movies?pageNum={pN}&pageSize={pS} | Page Number  Page Size | Products |
| P3-3 | GET | /v3/products/movies/comingsoon | None | Products |
| P3-4 | GET | /v3/products?pageNum={pN}&pageSize={pS}&productIds={list} | ProductIds (list of Product Ids)  Page Number  Page Size | Products |
| P3-5 | GET | /v3/products/{ProductID} | ProductId ID | Products |
| P3-6 | GET | /v3/products?pageNum={pN}&pageSize={pS}&q={searchText}&productTypes={pt}&sortField={sort}&operator={searchOperator}&field={searchField} | Search Text  SearchField  SearchOperator  Sort Field  Product Types  Page Number  Page Size | Products with Paging element |
| ~~P3-7~~ | ~~GET~~ | ~~/v3/titles/movies/default~~ | ~~Page Number~~  ~~Page Size~~ | ~~Titles~~ |
| ~~P3-8~~ | ~~GET~~ | ~~/v3/titles/movies/comingsoon~~ | ~~None~~ | ~~Titles~~ |
| P3-9 | GET | /v3/products/{pid}/similar | Product ID | Products |
| P3-10 | GET | /v3/products/movies/top20?period={period} | Period: 7 or 30 | Top20 |
| ~~P3-13~~ | ~~GET~~ | ~~/v3/products/{ProductId}/trailers~~ | ~~Product ID~~ | ~~Trailers~~ |
| P3-15 | GET | /v3/products/games/default?pageNum={pN}&pageSize={pS}&includeComingSoon={true/false} | includeComingSoon flag  Page Number  Page Size | Products (games Bulk list, younger than 250 days with ‘Display On Website’ set); |
| P3-16 | GET | /v3/products/games?pageNum={pN}&pageSize={pS} | Page Number  Page Size | Product Metadata for all games products |
| P3-17 | GET | /v3/products/movies/metrics?pageNum={pN}&pageSize={pS} | Page Number  Page Size | ProductMetrics |
| P3-18 | GET | /v3/products/games/metrics?pageNum={pN}&pageSize={pS} | Page Number  Page Size | ProductMetrics |
| **V3 Cache Update Operations** | | | | |
| PC3-1 | GET | /v3/products/movies/default?cacheRefresh=true | None | Products |
| PC3-2 | GET | /v3/products/movies?cacheRefresh=true | None | Products |
| PC3-3 | GET | /v3/products/movies/comingsoon?cacheRefresh=true | None | Products |
| PC3-4 | GET | /v3/products/games?cacheRefresh=true | None | Products |
| PC3-5 | GET | /v3/products/games/default?cacheRefresh=true | None | Products |
| ~~PC3-6~~ | ~~GET~~ | ~~/v3/titles/default?cacheRefresh=true~~ | ~~None~~ | ~~Titles~~ |
| ~~PC3-7~~ | ~~GET~~ | ~~/v3/titles/comingsoon?cacheRefresh=true~~ | ~~None~~ | ~~Titles~~ |
| PC3-8 | GET | /v3/products/movies/top20?period=7&cacheRefresh=true | Period=7 | Top20 |
| PC3-9 | GET | /v3/products/movies/top20?period=30&cacheRefresh=true | Period=30 | Top20 |
| ~~PC3-10~~ | ~~GET~~ | ~~/v3/products/trailers?cacheRefresh=true~~ | ~~None~~ | ~~Trailers~~ |
| PC3-11 | GET | /v3/products?sortField=RedboxReleaseDate&cacheRefresh=true | sortField | Products |
| PC3-12 | GET | /v3/products?sortField=Title &cacheRefresh=true | sortField | Products |
| PC3-13 | GET | /v3/products/movies/metrics?cacheRefresh=true |  | ProductMetrics |
| PC3-14 | GET | /v3/products/games/metrics?cacheRefresh=true |  | ProductMetrics |
| PC3-15 | GET | /v3/products/movies/default?comingSoon=true&cacheRefresh=true | comingSoon flag | Products |
| PC3-16 | GET | /v3/products/games/default?comingSoon=true&cacheRefresh=true | comingSoon flag | Products |

*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.

**V3 versions of Operations:**

* P3-1: Get the Default Movie Browse List
  + URL: /v3/products/movies/default?apiKey={key}&includeComingSoon={true/false}&pageNum{pN}&pageSize={pS}
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
    - includeComingSoon: optional (default: false)
    - 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 products matching the other criteria.
    - PageSize: optional; when specified along with the PageNumber, sets the number of products to return per-page. Default: 10
  + Output: Products, see schema 6.1.49 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 (PC1)**.
    - * + When the ‘includeComingSoon’ flag is true, the output will be provided by the contents of the **Products Default ComingSoon Catalog Cache (PC15)**.
  + 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 Last-Modified header will be set to the value of the ‘lastUpdated’ attribute in the Products output.
    - 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-Modified-Since header that equals or is later than the most recent update timestamp in the **Products Default Catalog Cache**.
* P3-2: Get the Full Movie Catalog
  + URL: /v3/products/movies?apiKey={key}&pageNum{pN}&pageSize={pS}
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
    - 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 products matching the other criteria.
    - PageSize: optional; when specified along with the PageNumber, sets the number of products to return per-page. Default: 10
  + Output: Products, see schema 6.1.49 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 (PC2)**.
  + 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 Last-Modified header will be set to the value of the ‘lastUpdated’ attribute in the Products output.
    - 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-Modified-Since header that equals or is later than the most recent update timestamp in the **Products In-Circulation Cache**.
* P3-3: Get the Coming Soon Movie List
  + URL: /v3/products/movies/comingsoon?apiKey={key}
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
  + Output: Products, see schema 6.1.49 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 ComingSoon Cache (PC3)**.
  + 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 Last-Modified header will be set to the value of the ‘lastUpdated’ attribute in the Products output.
    - 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-Modified-Since header that equals or is later than the most recent update timestamp in the **Products Default Catalog Cache**.
* P3-4: Get metadata about a set of Movies by ID
  + URL: /v3/products?apiKey={key}&pageNum={pN}&pageSize={pS}&productIds={pids}
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
    - List of Product IDs: optional comma separated list of Product IDs
    - 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 products matching the other criteria.
    - PageSize: optional; when specified along with the PageNumber, sets the number of products to return per-page. Default: 10
  + Output: Products, see schema 6.1.49 in Appendix A
  + Description: This is a request to retrieve specific products from the product catalog. The Products Service will fulfill the request by fetching the contents from the Full **Products Cache (PC11)**. If the productIds query parameter value is omitted or empty, the operation will return ALL the contents of the 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 Last-Modified header will be set to the value of the ‘lastUpdated’ attribute in the Products output.
    - 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-Modified-Since header that equals or is later than the most recent update timestamp in the **Products Default Catalog Cache**.
* P3-5: Get metadata of a single Product (Movie or Game)
  + URL: /v3/products/{ProductId}?apiKey={key}
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
    - ProductID: mandatory; the id of a single product
  + Output: Products, see schema 6.1.49 in Appendix A
  + Description: This is a request to retrieve a single product from the catalog. The Products Service will fulfill the request by fetching the contents from the **Full Products Cache (PC5)**.
  + 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 Last-Modified header will be set to the value of the ‘lastUpdated’ attribute in the Products output.
    - 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-Modified-Since header that equals or is later than the most recent update timestamp in the **Products In-Circulation Cache**.
* P3-6: Search for Products
  + URL: /v3/products? q={searchText}&searchField={field}&searchOperator={op}&productTypes={pt}&sortField={sort}&pageNum={pN}&pageSize={pS}
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
    - q: mandatory; the text to search for
    - searchField: optional: the field to search; Default=UpperCaseTitle;
    - searchOperator: optional: the operator to use (StartsWith, Contains);
    - productTypes: optional; comma delimited list of product types to search for (default: when omitted the search will return products of both types) (valid values: ‘Movies’, ‘Games’)
    - sortField: optional; the results will be sorted by this field, valid values are Title and RedboxReleaseDate, default is RedboxReleaseDate.
    - 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 products matching the other criteria.
    - PageSize: optional; when specified along with the PageNumber, sets the number of products to return per-page. Default: 10
  + Output: Products, see schema 6.1.49 in Appendix A
  + Description: This operation will search through one of the Full product caches. When the sort field is RedboxReleaseDate, this operation will use cache **PC11** to find the products that match the search criteria. When the sort field is Title, this operation will use cache **PC12** to find the products that match the search criteria. The caches are presorted. The result set can optionally be paginated.

This version of the Search restricts searching to the Title only, using the <UppercaseTitle> element (so that searches are case insensitive).

* + 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 Last-Modified header will be set to the value of the ‘lastUpdated’ attribute in the Products output.
    - 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-Modified-Since header that equals or is later than the most recent update timestamp in the **Products Default Catalog Cache**.
* (OUT\_OF\_SCOPE) P3-7: Get the Titles Feed
  + URL: /v3/titles/movies/default?apiKey={key}
  + 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 by using cache PC6.
  + 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-Modified-Since header that equals or is later than the most recent update timestamp in the **Titles Default Cache**.
* (OUT\_OF\_SCOPE) P3-8: Get the Titles Coming Soon list
  + URL: /v3/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 by using cache PC7.
  + 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-Modified-Since header that equals or is later than the most recent update timestamp in the **Titles Default Cache**.
* P3-9: Recommend Similar Titles
  + URL: /v3/products/{pid}/similar?apiKey={key}
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
    - ProductId: mandatory; the product ID upon which to base recommendations
  + Output: RecommendedProductsResponse, see schema 6.1.8 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 the result.
  + 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.

The Last-Modified header will be set to the value of the ‘lastUpdated’ attribute in the Products output.

* + - 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-Modified-Since header that equals or is later than the most recent update timestamp in the **Products Default Catalog Cache**.
* P3-10: Get the Top20 list
  + URL: /v3/products/movies/top20?period={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 a **Top20 External Format Cache,** either cache PC8 or PC9, depending upon the value of the period query string.
  + 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 Last-Modified header will be set to the value of the ‘lastUpdated’ attribute in the Top20 output.
    - 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-Modified-Since header that equals or is later than the most recent update timestamp in the **Top20 External Format Cache**.
* (OUT\_OF\_SCOPE) P3-13: Get Trailer metadata of a single Product
  + URL: /v3/products/{ProductId}/trailers/?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).
* P3-15: Get the Default Game Browse List
  + URL: /v3/products/games/default?apiKey={key}&includeComingSoon={true/false}&pageNum{pN}&pageSize={pS}
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
    - includeComingSoon: optional (default: false)
    - 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 products matching the other criteria.
    - PageSize: optional; when specified along with the PageNumber, sets the number of products to return per-page. Default: 10
  + Output: Products, see schema 6.1.49 in Appendix A
  + Description: This represents a request for the default list of Games (as displayed on the website) in Redbox External format by providing the contents of the **Games Default Catalog Cache (PC5)**. When the ‘includeComingSoon’ flag is true, the output will be provided by the contents of the **Games Default Coming Soon Catalog Cache (PC16)**.
  + 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 Last-Modified header will be set to the value of the ‘lastUpdated’ attribute in the Products output.
    - 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-Modified-Since header that equals or is later than the most recent update timestamp in the **Products Default Catalog Cache**.
* P3-16: Get the Full Games Catalog
  + URL: /v3/products/games?apiKey={key}&pageNum{pN}&pageSize={pS}
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
    - 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 products matching the other criteria.
    - PageSize: optional; when specified along with the PageNumber, sets the number of products to return per-page. Default: 10
  + Output: Products, see schema 6.1.49 in Appendix A
  + Description: This represents a request for the full list of Games in the catalog in Redbox External format by providing the contents of the **Games Cache (PC4)**.
  + 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 Last-Modified header will be set to the value of the ‘lastUpdated’ attribute in the Products output.
    - 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-Modified-Since header that equals or is later than the most recent update timestamp in the **Games Cache.**
* P3-17: Get the Movie Product Metrics
  + URL: /v3/products/movies/metrics?apiKey={key}&pageNum{pN}&pageSize={pS}
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
    - 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 products matching the other criteria.
    - PageSize: optional; when specified along with the PageNumber, sets the number of products to return per-page. Default: 10
  + Output: ProductsMetrics, see schema 6.1.52 in Appendix A
  + Description: This represents a request for the list of Movie Metrics by providing the contents of the **Movie Metrics Cache (Cache 13)**.
  + 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 Last-Modified header will be set to the value of the ‘lastUpdated’ attribute in the Products output.

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-Modified-Since header that equals or is later than the most recent update timestamp in the **Movies ProductsMetrics Cache**.

* P3-18: Get the Games Product Metrics
  + URL: /v3/products/games/metrics?apiKey={key}&pageNum{pN}&pageSize={pS}
  + Method: GET
  + Input:
    - ApiKey: mandatory; the security key
    - 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 products matching the other criteria.
    - PageSize: optional; when specified along with the PageNumber, sets the number of products to return per-page. Default: 10
  + Output: Products, see schema 6.1.52 in Appendix A
  + Description: This represents a request for the full list of Games in the catalog in Redbox External format by providing the contents of the **Games Product Metrics Cache (Cache 14)**.
  + 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 Last-Modified header will be set to the value of the ‘lastUpdated’ attribute in the Products output.

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-Modified-Since header that equals or is later than the most recent update timestamp in the **Games Products Metrics Cache**.

V3 Cache Refresh Operations:

* PC3-1: Refresh the Movies Default Cache
  + URL: /v3/products/movies/default?apiKey={key}&includeComingSoon={comingsoonFlag}&cacheRefresh=’true’
  + Input :
    - ApiKey: mandatory; the security key
    - comingsoonFlag: mandatory
  + Output: The contents of the newly refreshed cache
  + Description: Refresh cache content by calling /v3/Products/Movies/Default on the Products Source System gateway and replace the appropriate cache (either the one without comingsoon or the one with).
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this key.
* PC3-2: Refresh the Movies Cache
  + URL: /v3/products/movies?apiKey={key}&cacheRefresh=’true’
  + Input :
    - ApiKey: mandatory; the security key
  + Output: The contents of the newly refreshed cache
  + Description: Refresh cache content by calling /v3/Products/Movies on the Products Source System gateway and replace the cache.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this key.
* PC3-3: Refresh the ComingSoon Movies Cache
  + URL: /v3/products/movies/comingsoon?apiKey={key}&cacheRefresh=’true’
  + Input :
    - ApiKey: mandatory; the security key
  + Output: The contents of the newly refreshed cache
  + Description: Refresh cache content by calling /v3/Products/Movies/ComingSoon on the Products Source System gateway and replace the cache.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this key.
* PC3-4: Refresh the Games Cache
  + URL: /v3/products/games?apiKey={key}&cacheRefresh=’true’
  + Input :
    - ApiKey: mandatory; the security key
  + Output: The contents of the newly refreshed cache
  + Description: Refresh cache content by calling /v3/Products/Games on the Products Source System gateway and replace the cache.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this key.
* PC3-5: Refresh the Games Default Cache
  + URL: /v3/products/games/default?apiKey={key}&includeComingSoon={comingsoonFlag}&cacheRefresh=’true’
  + Input :
    - ApiKey: mandatory; the security key
    - comingsoonFlag: mandatory
  + Output: The contents of the newly refreshed cache
  + Description: Refresh cache content by calling /v3/Products/Games/Default on the Products Source System gateway and replace the appropriate cache.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this key.
* (OUT\_OF\_SCOPE) PC3-6: Refresh the Titles Default Cache
  + URL: /v3/titles/default?apiKey={key}&cacheRefresh=’true’
  + Input :
    - ApiKey: mandatory; the security key
  + Output: The contents of the newly refreshed cache
  + Description: Refresh cache content by calling /v3/Titles/Default on the Products Source System gateway and replace the cache.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this key.
* (OUT\_OF\_SCOPE) PC3-7: Refresh the Titles Coming Soon Cache
  + URL: /v3/titles/comingsoon?apiKey={key}&cacheRefresh=’true’
  + Input :
    - ApiKey: mandatory; the security key
  + Output: The contents of the newly refreshed cache
  + Description: Refresh cache content by calling /v3/Titles/ComingSoon on the Products Source System gateway and replace the cache.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this key.
* PC3-8: Refresh the Top20 7-day Period Cache
  + URL: /v3/top20?apiKey={key}&cacheRefresh=’true’&period=7
  + Input :
    - ApiKey: mandatory; the security key
    - Period: mandatory; identifies which Top20 cache to update (7 or 30 day cache)
  + Output: The contents of the newly refreshed cache
  + Description: Refresh cache content by calling /v3/Products/Movies/Top20?period=7 on the Products Source System gateway and replace the cache.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this key.
* PC3-9: Refresh the Top20 30-day Period Cache
  + URL: /v3/top20?apiKey={key}&cacheRefresh=’true’&period=30
  + Input :
    - ApiKey: mandatory; the security key
    - Period: mandatory; identifies which Top20 cache to update (7 or 30 day cache)
  + Output: The contents of the newly refreshed cache
  + Description: Refresh cache content by calling /v3/Products/Movies/Top20?period=30 on the Products Source System gateway and replace the cache.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this key.
* (OUT\_OF\_SCOPE) PC3-10: Refresh the Trailers Cache
  + URL: /v3/trailers?apiKey={key}&cacheRefresh=’true’
  + Input :
    - ApiKey: mandatory; the security key
  + Output: The contents of the newly refreshed cache
  + Description: Refresh cache content by calling /v3/Products/Movies/Trailers on the Trailers Source System gateway and replace the cache.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this key.
* PC3-11: Refresh the presorted (RedboxReleaseDate) Products Cache
  + URL: /v3/products?apiKey={key}&cacheRefresh=’true’&sortField=RedboxReleaseDate
  + Input :
    - ApiKey: mandatory; the security key
    - sortField: mandatory; identifies which presorted cache to update (Title or RedboxReleaseDate)
  + Output: The contents of the newly refreshed cache
  + Description: Refresh cache content by calling /v3/Products?sortField=RedboxReleaseDate on the Products Source System gateway and replace the cache.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this key.
* PC3-12: Refresh the presorted (Title order) Products Cache
  + URL: /v3/products?apiKey={key}&cacheRefresh=’true’&sortField=Title
  + Input :
    - ApiKey: mandatory; the security key
    - sortField: mandatory; identifies which presorted cache to update (Title or RedboxReleaseDate)
  + Output: The contents of the newly refreshed cache
  + Description: Refresh cache content by calling /v3/Products?sortField=Title on the Products Source System gateway and replace the cache.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this key.
* PC3-13: Refresh the Movies Products Metrics Cache
  + URL: /v3/products/movies/metrics?apiKey={key}&cacheRefresh=’true’
  + Input :
    - ApiKey: mandatory; the security key
  + Output: The contents of the newly refreshed cache
  + Description: Refresh cache content by calling /v3/Products/Movies/Metrics on the Products Source System gateway and replace the cache.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this key.
* PC3-14: Refresh the Games Products Metrics Cache
  + URL: /v3/products/games/metrics?apiKey={key}&cacheRefresh=’true’
  + Input :
    - ApiKey: mandatory; the security key
  + Output: The contents of the newly refreshed cache
  + Description: Refresh cache content by calling /v3/Products/Games/Metrics on the Products Source System gateway and replace the cache.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this 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?cacheRefresh=’true’ | None | Stores |
| ~~S5~~ | ~~POST~~ | ~~/stores/storechanged~~ | ~~StoreChanges~~ |  |
| S6 | GET | /stores/postalcode/{zipCode}}?radius={r}&count={c}&retailer={ret}&sortBy={s}&sortDir={asc/desc}&pageNum={pN}&pageSize={pS} | ZipCode 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 |

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 Last-Modified header will be set to the value of the ‘lastUpdated’ attribute in the Stores output.

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-Modified-Since header that equals or is later than the most recent update timestamp 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: optional; 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 Last-Modified header will be set to the value of the ‘lastUpdated’ attribute in the Stores output.

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-Modified-Since header that equals or is later than the most recent update timestamp 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 (Asc/Desc). Default: Asc.
    - 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 Last-Modified header will be set to the value of the ‘lastUpdated’ attribute in the Stores External Format Cache.

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-Modified-Since header that equals or is later than the most recent update timestamp in the **Stores External Format Cache**.

* S4: Refresh the Stores Cache
  + URL: /stores?apiKey={key}&cacheRefresh=’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.
* S6: Find Stores by ZipCode
  + URL: /stores/postalcode/{zipCode}?apiKey={key}&radius={r}&count={c}&retailer={ret}& sortBy={sB}&sortDir={asc/desc}&pageNum={pN}&pageSize={pS}
  + Input:
    - zipCode: location expressed as a US Postal ZipCode (6 digits)
    - Radius: optional; maximum distance in miles from the location, default: 10
    - Count: optional; the maximum number of stores to return, default:10, 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 (Asc/Desc). Default: Asc.
    - 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 ZipCode.
  + Description: Respond to a request for a list of Stores that are located near a given ZipCode location.
  + Apigee Caching: Responses from the Stores Gateway target may be cached on the Apigee side for 60 minutes (configurable).
  + 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 Last-Modified header will be set to the value of the ‘lastUpdated’ attribute in the Stores External Format Cache.

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-Modified-Since header that equals or is later than the most recent update timestamp in the **Stores External Format Cache**.

##### 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 | /v3/inventory/stores/{storeID}?products={listOfProductIDs}&pageNum={pN}&pageSize={pS} | StoreId 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 |
| I4 | GET | /v3/inventory/stores/{storeId}?cacheRefresh=’true’ | StoreId  Inventory | None |
| I5 | GET | /v3/inventory/stores/latlong/{lat,long}?products={listOfProductIDs}&pageNum={pN}&pageSize={pS}&radius={r}&count={c}&retailer={retailer}&sortBy={s}&sortDir={asc/desc} | Lat/Long;  List of Product IDs for which Inventory status is to be retrieved;  Page Number and Page Size;  Search radius (in miles);  Count of number of Stores to return;  Retailer name to filter;  SortBy (Distance or Retailer);  SortDirection | InventoryLookup |
| I6 | GET | /v3/inventory/stores/postalcode/{zipcode}?products={listOfProductIDs}&pageNum={pN}&pageSize={pS}&radius={r}&count={c}&retailer={retailer}&sortBy={s}&sortDir={asc/desc} | ZipCode;  List of Product IDs for which Inventory status is to be retrieved;  Page Number and Page Size;  Search radius (in miles);  Count of number of Stores to return;  Retailer name to filter;  SortBy (Distance or Retailer);  SortDirection | InventoryLookup |

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: /v3/inventory/stores/{storeId}?apiKey={key}&products={listOfProductIDs}&pageNum={pN}&pageSize={pS}
  + Method: GET
  + Input:
    - StoreId: mandatory; a StoreId indicating which store for which information should be retrieved.
    - Products: 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 store, optionally filtered to a set of products. If no ProductIDs are provided, this operation returns all the Inventory records for the identified 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 allow this response to be cached by downstream clients.
    - The Cache-Control: max-age header will be set to a configurable value (eg: 1 minute).
    - The Last-Modified header will be set to the value of the ‘lastUpdated’ attribute in the StoreInventory output.

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-Modified-Since header that equals or is later than the most recent update timestamp in the Inventory cache for that store.

* I4: Force Update to a Store’s Inventory Cache
  + URL: /v3/inventory/stores/{storeId}?apiKey={key}&cacheRefresh=’true’
  + Method: GET
  + Input :
    - StoreId: mandatory; the StoreId of the store to be refreshed3
  + Output: None
  + Description: The Inventory Endpoint will call the Redbox Inventory OpenServices layer gateway and refresh the cache content by replacing the cache with the provided Inventory payload.
  + Security: The caller will provide an ApiKey which will be validated by Apigee. Only Redbox will have this key.
* I5: Get Inventory For Store(s) by Location (Lat/Long)
  + URL: /v3/inventory/stores/latlong/{lat,long}?products={listOfProductIDs}&pageNum={pN}&pageSize={pS}&radius={r}&count={c}&retailer={retailer}&sortBy={s}&sortDir={asc/desc}
  + Method: GET
  + Input:
    - Location: mandatory; a latitude/longitude location around which stores will be searched;
    - Products: 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
    - 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 (Asc/Desc). Default: Asc.
  + 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 Stores.
  + 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: 1 minute).
    - Because there is no feasible way to determine whether inventory has changed for a search of stores’ inventory, the Apigee server will not support conditional GET requests. The response for these requests will not include a Last-Modified Header.
* I6: Get Inventory For Store(s) by ZipCode
  + URL: /v3/inventory/stores/postalcode/{zipcode}?products={listOfProductIDs}&pageNum={pN}&pageSize={pS}&radius={r}&count={c}&retailer={retailer}&sortBy={s}&sortDir={asc/desc}
  + Method: GET
  + Input:
    - Location: mandatory; a PostalCode (zipCode) location around which stores will be searched;
    - Products: 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
    - 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 (Asc/Desc). Default: Asc.
  + 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 Stores.
  + 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: 1 minute).
    - Because there is no feasible way to determine whether inventory has changed for a search of stores’ inventory, the Apigee server will not support conditional GET requests. The response for these requests will not include a Last-Modified Header.

##### 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 | /v4/reservation/statusmonitor | None | Status |
| R1 | POST | /v4/cart/validate | User ID of customer  Cart | CartValidation |
| R2 |  | Deprecated |  |  |
| R3 |  | Deprecated |  |  |
| R4 | POST | /v4/cart/price | User ID  Cart | PricedCartResponse |
| R5 | POST | /v4/reservations | Cart | ReservationResponse |

**See Section 2.12.1 on Security for more information on how these sensitive operations will be secured.**

Operation Detail:

* R0: Get the Status of the Service
  + URL: /v4/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 I 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 I will have this ApiKey.
* R1: Validate a Cart
  + URL: /v4/cart/validate?access\_token={token}
  + Method: POST
  + Input :
    - UserID: CustomerNumber or VendorCode.CustomerPartnerNumber
    - Cart (see schema: 6.1.18)
  + Output: CartValidation (see schema: 6.1.19)
  + Description:
    - * + Apigee will also validate that the UserID in the Cart matches the UserID provided in the Authentication token, if a token is available.
        + Apigee will pass-through the call to the Reservation Gateway by calling /v3/cart (using a POST method and passing along the Cart object).
        + After the Reservation Gateway has returned a CartValidation, if the CartValidation object indicates no errors, the Apigee Layer will further validate the cart by validating that the ProductIDs in the cart are “InStock” in the inventory of the StoreID (provided that the cart contains both a StoreID and at least one ProductID). This validation can be performed by calling out to Inventory operation I1.
  + Security: The caller will provide an Oauth access\_token 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: /v4/cart/price?access\_token={token}
  + Method: POST
  + Input :
    - UserID: (optional) CustomerNumber or VendorCode.CustomerPartnerNumber
    - 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 PricedCartResponse schema(see schema: 6.1.21)
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a UserID, it MUST match the UserID in the Cart object (if a UserID is present in the cart).
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* R5: Check-out the Cart
  + URL: /v4/reservations?access\_token={token}
  + Method: POST
  + Input :
    - Cart (see schema: 6.1.18 )
  + 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 Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a UserID, it MUST match the UserID in the Cart object.
  + 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:

Note: In this table, these URLs are prefixed with one of the following to identify the customer:

**/v5/customers/{CustomerNumber}**

**/v5/customers/{VendorCode}.{CustomerPartnerNumber}**

| **API ID** | **HTTP Method** | **URL structure** | **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}&basedUponProductId={pId}&includeUserHistory={true/false}&numRecommendations={maxRecomm} | CustomerNumber  ProductType  ProductId  IncludeUserHistory Flag  Max Number of Recommendation | 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  CVV |  |
| CP16 | GET | /credits/balance | CustomerNumber | Credits Balances |
| CP17 | GET | /credits/history | CustomerNumber | Credits History |
| CP18 | POST | /creditcard | CustomerNumber  CreditCard  CVV | Account Info |
| CP19 | POST | /customers *{without the use of a customerNumber}* | NewCustomer  CustomerIdentity | NewCustomer  CreationResponse |
| CP20 | GET | /partner/profile | CustomerNumber  PartnerId | PartnerProfile |
| CP21 | POST | /partner/relationship | CustomerNumber  PartnerId  AssignPartnerRelationshipRequest | AssignPartnerRelationshipResponse |
| CP22 | GET | /partner/preferences?partnerId={partnerId} | CustomerNumber  PartnerId | PartnerPreferences |
| CP23 | PUT | /partner/preferences?partnerId={partnerId} | CustomerNumber  PartnerId  PartnerPreferences |  |
| CP24 | GET | /invoices/{transactionNumber} | CustomerNumber  TransactionNumber | Invoice |
| CP25 | PUT | /identity | CustomerNumber  LoginChange | Status  200: Success  400: CustomerEmail Not Found  500: Other Error |
| CP26 | POST | /customers/identity/resetpassword  *{without the use of a customerNumber}* | PasswordResetRequest | status |
| CP27 | PUT | /identity/password | CustomerNumber  PasswordChange | status |
| CP28 | DELETE | /accounts/{AccountNo} | CustomerNumber  AccountNumber | Status  200: Success  400: Account Not Found  500: Other Error |
| CP29 | POST | /identity/password/confirm | CustomerNumber  PasswordConfirmation | PasswordConfirmationResponseToken |
| CP30 | POST | -not used- |  |  |
| CP31 | PUT | /partner/profile | CustomerNumber  PartnerId  PartnerProfile | Status  200: Success  400: NoMatch, validation error or missing data  500: Other error |
| CP32 | POST | /accounts/{accountNumber}/verify | CustomerNumber  AccountNumber  cvvCode | Status  200: Success with no payload when verification succeeds  200: with Error payload when card fails verification  400: validation error or missing data  500: Other error |
| CP33 | GET | /accounts/{acccountNumber}/subscriptionbilling?partnerId={partnerID} | CustomerNumber  AccountNumber  PartnerId | Status  200: Success  400: Validation error or missing data  500: Other error |

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: /v5/customers/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 I 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 I will have this ApiKey.
* CP1: Get Profile
  + URLs: /v5/customers/{CustomerNumber}/profile?access\_token={token}

Or /v5/customers/{VendorCode}.{CustomerPartnerNumber}/profile?access\_token={token}

* + Method: GET
  + Input :
    - CustomerNumber: ID that identifies the customer

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of the customer
  + Output: BasicProfile (see schema: 0)
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP2: Save Profile
  + URLs: /v5/customers/{CustomerNumber}/profile?access\_token={token}

Or /v5/customers/{VendorCode}.{CustomerPartnerNumber}/profile?access\_token={token}

* + Method: PUT
  + Input :
    - CustomerNumber: ID that identifies the customer

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of the customer
    - BasicProfile (see schema: 0)
  + Output: None
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP3: Get Preferences
  + URLs: /v5/customers/{CustomerNumber}/preferences?access\_token={token}

Or /v5/customers/{VendorCode}.{CustomerPartnerNumber}/preferences?access\_token={token}

* + Method: GET
  + Input :
    - CustomerNumber: ID that identifies the customer

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of the customer
  + Output: Preferences (see schema: 0)
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP4: Save Preferences
  + URL: /v5/customers/{CustomerNumber}/preferences?access\_token={token}

Or /v5/customers/{VendorCode}.{CustomerPartnerNumber}/preferences?access\_token={token}

* + Method: PUT
  + Input :
    - CustomerNumber: ID that identifies the customer

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of the customer
    - Preferences (see schema: 0)
  + Output: None
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP5: Get Rental History
  + URL: /v5/customers/{CustomerNumber}/rentalhistory?access\_token={token}

Or /v5/customers/{VendorCode}.{CustomerPartnerNumber}/rentalhistory?access\_token={token}

* + Method: GET
  + Input :
    - CustomerNumber: ID that identifies the customer

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of the customer
  + Output: RentalHistory (see schema: 6.1.25)
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP6: Get Queues
  + URL: /v5/customers/{CustomerNumber}/queues?access\_token={token}

Or /v5/customers/{VendorCode}.{CustomerPartnerNumber}/queues?access\_token={token}

* + Method: GET
  + Input :
    - CustomerNumber: ID that identifies the customer

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of the customer
  + Output: List of Queue Info (see schema: 0)
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP7: Get a Queue
  + URL: /v5/customers/{CustomerNumber}/queues/{queueId}?access\_token={token}

Or /v5/customers/{VendorCode}.{CustomerPartnerNumber}/queues/{queueId}?access\_token={token}

* + Method: GET
  + Input :
    - CustomerNumber: ID that identifies the customer

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of 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 Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP8: Update Queue
  + URL: /v5/customers/{CustomerNumber}/queues/{queueId}?access\_token={token}

Or /v5/customers/{VendorCode}.{CustomerPartnerNumber}/queues/{queueId}?access\_token={token}

* + Method: PUT
  + Input :
    - CustomerNumber: ID that identifies the customer

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of the customer
    - QueueId
    - QueueInfo
  + Output: None
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP9: Delete a Queue
  + URL: /v5/customers/{CustomerNumber}/queues/{queueId}?access\_token={token}

Or /v5/customers/{VendorCode}.{CustomerPartnerNumber}/queues/{queueId}?access\_token={token}

* + Method: DELETE
  + Input :
    - CustomerNumber: ID that identifies the customer

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of the customer
    - QueueId
  + Output: None
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP10: Add/Update a Product to a Queue
  + URL: /v5/customers/{CustomerNumber}/queues/{queueId}/{position},{productId}?access\_token={token}

Or /v5/customers/{VendorCode}.{CustomerPartnerNumber}/queues/{queueId}/{position}, {productId}?access\_token={token}

* + Method: PUT
  + Input :
    - CustomerNumber: ID that identifies the customer

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of 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 Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP11: Remove a Product from a Queue
  + URL: /v5/customers/{CustomerNumber}/queues/{queueId}/{productId}?access\_token={token}

Or /v5/customers/{VendorCode}.{CustomerPartnerNumber}/queues/{queueId}/{productId}?access\_token={token}

* + Method: DELETE
  + Input :
    - CustomerNumber: ID that identifies the customer

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of 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 Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP12: Get Recommendations
  + URLs: /v5/customers/{CustomerNumber}/recommendedproducts?access\_token={token}&productType={type}&basedUponProductId={pId}&includeUserHistory={true/false}&numRecommendations={maxRecomm}

Or /v5/customers/{VendorCode}.{CustomerPartnerNumber}/recommendedproducts?access\_token={token}&productType={type}&basedUponProductId={pId}&includeUserHistory={true/false}&numRecommendations={maxRecomm}

* + Method: GET
  + Input :
    - CustomerNumber: ID that identifies the customer

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of the customer
    - type: optional, one of the product type values (eg, “movie”)
    - basedUponProductId: optional, ProductID upon which recommendations should be made
    - includeUserHistory: optional, Boolean to indicate whether the customer’s rental history should influence the recommendations. Default: True
    - numRecommendations: optional, integer; maximum number of recommendations to return. Max: 10, Default: 10
  + Output: Recommendations (list of Product IDs) (see schema: 6.1.8)
  + Description: Pass-thru the request to the CustomerAccount Gateway. The value of the ‘includeUserHistory’ flag is Ignored if the caller does not have the “RENTALHISTORY” scope. This must be enforced at the OpenServices layer. The return schema includes a flag: “basedUponRentalHistory” which should be populated based upon whether the rental history was used or not (reflecting whether the caller had the correct Scope).
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP13: Get Accounts
  + URLs: /v5/customers/{CustomerNumber}/accounts?access\_token={token}

Or /v5/customers/{VendorCode}.{CustomerPartnerNumber}/accounts?access\_token={token}

* + Method: GET
  + Input :
    - CustomerNumber: ID that identifies the customer

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of 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 Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP14: Get Account
  + URLs:

/v5/customers/{CustomerNumber}/accounts/{AccountNo}?access\_token={token}

/v5/customers/{VendorCode}.{CustomerPartnerNumber}/accounts/{AccountNo}?access\_token={token}

* + Method: GET
  + Input :
    - CustomerNumber: ID that identifies the customer

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of 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 Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP15: Update Account
  + URLs:

/v5/customers/{CustomerNumber}/accounts/{AccountNo}?access\_token={token} }

/v5/customers/{VendorCode}.{CustomerPartnerNumber}/accounts/{AccountNo}?access\_token={token}

* + Method: PUT
  + Input :
    - CustomerNumber: ID that identifies the customer

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of the customer
    - AccountNo: identifier of the account
    - Account Info: an object describing an account (schema 6.1.30)
  + Output: None
  + Description: Pass-thru the request to the CustomerAccount Gateway to update the attributes of an account.
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP16: Get Credit Balance
  + URLs:

/v5/customers/{CustomerNumber}/credits/balance?access\_token={token}

/v5/customers/{VendorCode}.{CustomerPartnerNumber}/credits/balance?access\_token={token}

* + Method: GET
  + Input :
    - CustomerNumber: ID that identifies the customer

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of the customer
  + Output: CreditsBalance (see schema: 6.1.32)
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP17: Get Credit Use History
  + URLs:

/v5/customers/{CustomerNumber}/credits/history?access\_token={token}

/v5/customers/{VendorCode}.{CustomerPartnerNumber}/credits/history?access\_token={token}

* + Method: GET
  + Input :
    - CustomerNumber: ID that identifies the customer

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of the customer
  + Output: CreditsHistory (see schema: 6.1.33)
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP18: Add a Credit Card
  + URLs:

/v5/customers/{CustomerNumber}/creditcard?access\_token={token}

/v5/customers/{VendorCode}.{CustomerPartnerNumber}/creditcard?access\_token={token}

* + Method: POST
  + Input :
    - CustomerNumber: ID that identifies the customer

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of 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 Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP19: Create A New Customer
  + URL: /v5/customers?access\_token={token}
  + Method: POST
  + Input :
    - NewCustomerCreationRequest (see schema 6.1.47)
  + Output: NewCustomerCreationResponse (see schema 6.1.48)
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present.
  + 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 NewCustomerCreationResponse message 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).

* CP20: Get Customer Partner Profile
  + URLs:

/v5/customers/{CustomerNumber}/partner/profile?partnerId={partner}&access\_token={token}

/v5/customers/{VendorCode}.{CustomerPartnerNumber}/partner/profile?access\_token={token}

* + Method: GET
  + Input :
    - CustomerNumber: ID that identifies the customer
    - PartnerId: Identifier for the Partner

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of the customer
  + Output: Partner Profile (see schema: 6.1.55)
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP21: AssignPartnerRelationship
  + URL: /v5/customers/{CustomerNumber}/partner/relationship?partnerId={partner}&access\_token={token}
  + Method: POST
  + Input :
    - CustomerNumber: ID that identifies the customer
    - PartnerId: Identifier for the Partner
    - AssignPartnerRelationshipRequest
  + Output: AssignPartnerRelationshipResponse (see schema: 6.1.57)
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP22: Get Customer Partner Preferences
  + URLs:

/v5/customers/{CustomerNumber}/partner/preferences?partnerId={partner}&access\_token={token}

/v5/customers/{VendorCode}.{CustomerPartnerNumber}/partner/preferences?access\_token={token}

* + Method: GET
  + Input :
    - CustomerNumber: ID that identifies the customer
    - PartnerId: Identifier for the Partner

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of the customer
  + Output: Partner Preferences (see schema: 6.1.54)
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP23: Update Customer Partner Preferences
  + URLs:

/v5/customers/{CustomerNumber}/partner/preferences?partnerId={partner}&access\_token={token}

/v5/customers/{VendorCode}.{CustomerPartnerNumber}/partner/preferences?access\_token={token}

* + Method: PUT
  + Input :
    - CustomerNumber: ID that identifies the customer
    - PartnerId: Identifier for the Partner

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of the customer
    - PartnerPreferences (in body of the HTTP message)
  + Output: None
  + Description: Pass-thru the request to the CustomerAccount Gateway
    - The PartnerPreferences schema includes the basic Preferences information as a child element. Updates made via this method may affect both the partner specific preferences data and basic preferences data.
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP24: Get Invoice
  + URLs:

/v5/customers/{CustomerNumber}/invoices/{transactionNumber}?access\_token={token}

/v5/customers/{VendorCode}.{CustomerPartnerNumber}/invoices/{transactionNumber}?access\_token={token}

* + Method: GET
  + Input :
    - CustomerNumber: ID that identifies the customer

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of the customer
    - TransactionNumber: Identifier for the transaction
  + Output: Invoice (see schema: 6.1.53)
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP25: Update Customer Login
  + URLs:

/v5/customers/{CustomerNumber}/identity?access\_token={token}

/v5/customers/{VendorCode}.{CustomerPartnerNumber}/identity?access\_token={token}

* + Method: PUT
  + Input :
    - CustomerNumber: ID that identifies the customer

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of the customer
    - LoginChange: updated email, name, postal code (Schema: 6.1.58)
  + Output: Return an HTTP status code to describe the outcome of the operation:
    - 200: Login Change operation successful.
    - 400: Field validation failed (required field missing or customer not found).
    - 500: Change not successful; other error.
  + Description: Pass-thru the request to the CustomerAccount Gateway.
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
    - * + The input schema requires a PasswordConfirmationResponseToken as input. This value is the output of CP29, which must be called before the client calls this operation.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP26: Reset Password
  + URL: /v5/customers/identity/resetpassword? Access\_token={token}
  + Method: POST
  + Input :
    - PasswordResetRequest: the email address of the customer (schema: 6.1.62)
  + Output: Return an HTTP status code to describe the outcome of the operation:
    - 200: Password Reset operation successful; reset email sent to address.
    - 400: Reset not successful; email address not found.
    - 500: Reset not successful; other error.
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP27: Change Password
  + URLs:

/v5/customers/{CustomerNumber}/identity/password?access\_token={token}

/v5/customers/{VendorCode}.{CustomerPartnerNumber}/identity/password?access\_token={token}

* + Method: PUT
  + Input :
    - CustomerNumber: ID that identifies the customer

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of the customer
    - PasswordChange: old and new password (Schema: 6.1.59)
  + Output: Return an HTTP status code to describe the outcome of the operation:
    - 200: Password Change operation successful.
    - 400: Field validation failed (required field missing or password complexity validation).
    - 500: Change not successful; other error.
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
    - * + The input schema requires a PasswordConfirmationResponseToken as input. This value is the output of CP29, which must be called before the client calls this operation.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP28: Delete Account
  + URLs:

/v5/customers/{CustomerNumber}/accounts/{AccountNo}?access\_token={token}

/v5/customers/{VendorCode}.{CustomerPartnerNumber}/accounts/{AccountNo}?access\_token={token}

* + Method: DELETE
  + Input :
    - CustomerNumber: ID that identifies the customer

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of the customer
    - AccountNo: Identifier of an account
  + Output:
    - 200: Delete operation successful.
    - 400: Field validation failed (required field missing or AccountNotFound).
    - 500: Change not successful; other error.
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP29: Confirm Password
  + URLs:

/v5/customers/{CustomerNumber}/identity/password/confirm?access\_token={token}

/v5/customers/{VendorCode}.{CustomerPartnerNumber}/identity/password/confirm?access\_token={token}

* + Method: POST
  + Input :
    - CustomerNumber: ID that identifies the customer

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of the customer
    - PasswordConfirmationRequest: password (Schema: 6.1.60)
  + Output:
    - * + PasswordConfirmationResponseToken (Schema: 6.1.63)
        + Return an HTTP status code to describe the outcome of the operation:
    - 200: Password confirmation operation successful.
    - 400: Field validation failed (required field missing or password doesn’t match).
    - 500: Change not successful; other error.
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP30: - NOT USED —
* CP31: Update Partner Profile
  + URLs:

/v5/customers/{CustomerNumber}/partner/profile?access\_token={token}&partnerId={VendorCode}

/v5/customers/{VendorCode}.{CustomerPartnerNumber}/partner/profile?access\_token={token}

* + Method: PUT
  + Input :
    - CustomerNumber: ID that identifies the customer
    - PartnerId: Identifier for the Partner

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of the customer
    - PartnerProfile: (Schema: 6.1.55)
  + Output: Return an HTTP status code to describe the outcome of the operation:
    - 200: Operation successful.
    - 400: Field validation failed (required field missing, date out-of-range, customer or partner not found).
    - 500: Change not successful; other error.
  + Description: Pass-thru the request to the CustomerAccount Gateway
    - The PartnerProfile schema includes the Basic Profile information as a child element. Updates made via this method may affect both the partner specific profile data and basic profile data.
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP32: Verify Credit Card
  + URLs:

/v5/customers/{CustomerNumber}/accounts/{accountNumber}/verify?access\_token={token}

/v5/customers/{VendorCode}.{CustomerPartnerNumber}/ accounts/{accountNumber}/verify?access\_token={token}

* + Method: POST
  + Input :
    - CustomerNumber: ID that identifies the customer

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of the customer
    - CreditCardVerificationRequest: mandatory; schema 6.1.64
  + Output: Return an HTTP status code to describe the outcome of the operation:
    - 200: With no response payload means that all card checks succeeded
    - 200: With an Error payload: means that one or more card verification checks failed.
    - 400: Field validation failed (required field missing, customer or partner not found, account not found).
    - 500: Not successful; other error.
  + Description: Pass-thru the request to the CustomerAccount Gateway.
    - * + This operation will perform an AVSCheck for the card account. When a CVV code is provided the CVV Code will be also be verified.
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.
* CP33: Set Account for Recurring Partner Billing
  + URLs:

/v5/customers/{CustomerNumber}/accounts/{accountNumber}/subscriptionbilling?access\_token={token}&partnerId={VendorCode}

/v5/customers/{VendorCode}.{CustomerPartnerNumber}/accounts/{accountNumber}/subscriptionbilling?access\_token={token}

* + Method: GET
  + Input :
    - CustomerNumber: ID that identifies the customer
    - PartnerId: Identifier for the Partner

OR

* + - VendorCode: Identifier of the partner
    - CustomerPartnerNumber: Partner-specific ID of the customer
    - AccountNumber: identifier of the credit card account that is to be used for recurring billing by the identified partner
  + Output: Return an HTTP status code to describe the outcome of the operation:
    - 200: Operation successful.
    - 400: Field validation failed (required field missing, date out-of-range, customer or partner not found).
    - 500: Change not successful; other error.
  + Description: Pass-thru the request to the CustomerAccount Gateway
  + Security: The caller will provide an Oauth access\_token which will be validated by Apigee. The authentication token MUST be present. IF the token is associated with a user, that user MUST match the CustomerNumber given in the operation URL and/or payload.
  + Caching Control: The Apigee layer will set cache-control headers on the response to Private.

#### JSON Support

By default, all Apigee endpoint operations will accept input and render output in XML format using the schemas defined for each data contract as defined in Appendix 6.1. Each operation will also support input and output in JSON data format. The following conventions will apply:

* A request for JSON output is signaled by appending “.json” to the URL (before the query string)
  + Example: /v3/products/movies/default.json?includeComingSoon={Boolean}
* A request that output be in jSON-P (Json with padding) is signaled by appending “.json” to the URL and including a callback querystring parameter.
  + Example: /v3/products/movies/default.json?includeComingSoon={false}&callback=’appCallback’
* Reservations and Customers will accept JSON as input.
  + This is signaled by appending “.json” to the URL that the data is posted to

OR

* + Setting the Content-Type header to ‘application/json’
    - Example: CP2 POST /v5/Customers/{id}/profile.json

The implementation of the JSON support features will be implemented via:

* Output will be generated using the Apigee JSON output policy. It will be configured with the following defaults:
  + Recognize Numbers: true
  + NULL equivalent: NULL
  + Suppress Namespace Prefixes: true
  + Prefix string for attributes: @
* JSON-P support will be a custom developed policy
* JSON input will be supported via a custom input policy that transforms the json input into XML and then passes the XML onto the regular processing pipeline.

JSON output is supported using an out-of-the box Apigee policy. The output follows industry conventions of converting XML into JSON, therefore this document will not separately define JSON output schemas for each data contract.

JSON input schemas for the Reservations and Customers operations that take input are defined using JSON Schema notation and are included in Appendix 6.2.

#### 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)

Built-in Apigee Policy Components

The attached files define which off-the-shelf Apigee policy components will be used by each Operation.





Apigee provides a configurable set of XML protection settings & configurations. OpenAPI will use the following settings:

|  |  |
| --- | --- |
| **XML Payload Protection Settings** |  |
| Max. Node Depth | 15 |
| Max. Attributes Per Element | 25 |
| Max. Namespace Per Element | 10 |
| Max. Sibling Nodes | 50 |
| **Limit Payload Size (Max Characters)** |  |
| Max. Content Text | 150 |
| Max Attribute Value | 150 |
| Max Comment Size | 1 |
| Max Cdata Size | 1 |
| Max PI (ProcessingInstructions) Size | 100 |
| **Limit on Length of Tag Names (Max Characters)** |  |
| Max Element Name | 50 |
| Max Attribute Name | 50 |
| Max. Namespace Prefix | 50 |
| Max. PI Name | 100 |
| **On Error** |  |
| Send Error Response | Custom |
| Error Message Format | tbd |
| Error Message | Bad Request |
| Response Code | 400 |

#### 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 I {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 I 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 I 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 I by calling the Open Services Layer over the Internet. The external I 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 I 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

### 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.

#### ActivityID tracing

The Apigee gateway proxy will accept an activity ID with all requests that can be used to correlate log entries across affiliate applications, OpenAPI logs and downstream I systems. The ActivityID will be represented as a custom HTTP Header (X-I-ActivityID). If the calling affiliate application fails to provide a value through this header, the Apigee gateway will create one. The ActivityID will be included in all Apigee log and audit entries and in all OpenServices log and audit entries. It will be passed along to downstream I systems to those systems that accept such a correlation value.

## 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 (oolean. 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 two different means.

The Products, Stores, and Inventory operations 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 as a query string parameter. 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 Customers and Reservations operations will be secured using Oauth2.0. This requires the calling client to authenticate itself and optionally the end-user. Upon authentication, the client will be issued a short-lived Oauth access\_token. The access\_token will be provided by the client to Apigee with every request as a query string parameter. Apigee will validate the access\_token on each request.

### 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 I 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 I applications – The firewall will be configured to allow HTTP traffic (port 80) from any I internal server.

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

In addition to firewall security, all calls to the source system proxy (aka, OpenServices) will be authenticated using HTTP BASIC authentication. The credentials presented in the BASIC auth header will be validated against the I Activity Directory. The credentials will identify service accounts which must be a member of an AD Group, to which access to the OpenServices operations is granted.

## 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.
* I.com will be modified to provide Recommendations as an application service.
* I.com will be modified to support web single sign-on.
* I.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, * I/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 I identifiers: I.com exposes the internal movie product identifier (ProductNumber) in several spots on the site (such as thru image tags and data provided via its I 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 I 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 I. 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 I. 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; I’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

Schemas files can be found attached to this document in a zip file located in Section 6.2.

### Common Schemas

#### Links

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)”

#### Paging and Extensions

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” schemaLocation=”atom.xsd” />

  <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>

#### Errors

The Apigee gateway will return the following schema whenever the processing of a request results in an error. The schema is included in the schema files as OpenAPI-Error\_V1.xsd.

Example:

<Error xmlns=”http://api.redbox.com/v1/OpenAPI-Error”>

  <Status>Status Code</Status>

  <Request><https://api.redbox.com/URL> - the URL of the original request/Request>

  <ActivityId>ActivityId1</ActivityId>

  <Datetime>1900-01-01T01:01:01-06:00</Datetime>

  <Code>Code1</Code>

  <Message>Error Message</Message>

  <MoreInfo>Developer information and or links/pointers to documentation</MoreInfo>

</Error>

Schema:

<xs:schema id=”OpenAPI-Error\_V1”

    targetNamespace=”http://api.redbox.com/v1/OpenAPI-Error”

    elementFormDefault=”qualified”

    xmlns=”http://api.redbox.com/v1/OpenAPI-Error”

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

>

  <xs:complexType name=”Error”>

    <xs:sequence>

      <xs:element name=”Status” type=”xs:string” />

      <xs:element name=”Request” type=”xs:anyURI” />

      <xs:element name=”ActivityId” type=”xs:string” />

      <xs:element name=”Datetime” type=”xs:dateTime” />

      <xs:element name=”Code” type=”xs:string” />

      <xs:element name=”Message” type=”xs:string” />

      <xs:element name=”MoreInfo” type=”xs:string” />

    </xs:sequence>

  </xs:complexType>

  <xs:element name=”Error” type=”Error” />

</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:oolean” 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 Products Schema

**This is the Products schema for V2. The V3 version of this schema is at 6.1.49.**

**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:enumeration value=”ALL AGES” />

      <xs:enumeration value=”TVY” />

      <xs:enumeration value=”TVY7” />

      <xs:enumeration value=”TVG” />

      <xs:enumeration value=”TVPG” />

      <xs:enumeration value=”TV14” />

      <xs:enumeration value=”TVMA” />

    </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

**This schema has been deprecated and is no longer used.**

Reference for [http://localhost:56745/RedboxAPIProducts/Products/Movies?apiKey={KEY}](http://localhost:56745/RedboxAPIProducts/Products/Movies?apiKey=%7bKEY%7d)

**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

**This schema has been deprecated and is no longer used.**

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

**This schema has been deprecated and is no longer used.**

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 6.1.3 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=”1900-01-01T01:01:01-06:00” xmlns=”http://api.redbox.com/v3/Products/TitleList”>

  <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” />

        <RedboxDoNotRentDate 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>Title2</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>

        <RedboxDoNotRentDate>1900-01-01</RedboxDoNotRentDate>

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

      </ProductRef>

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

        <Title>Title3</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>

        <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” />

        <RedboxDoNotRentDate 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=”UPC” 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>Title4</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>

        <RedboxComingSoonDate>0001-01-01</RedboxComingSoonDate>

        <RedboxDoNotRentDate>0001-01-01</RedboxDoNotRentDate>

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

      </ProductRef>

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

        <Title>Title5</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” />

        <RedboxDoNotRentDate 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>Title6</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>

        <RedboxDoNotRentDate>9999-12-31</RedboxDoNotRentDate>

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

      </ProductRef>

    </ProductRefs>

    <Extensions />

  </Title>

</Titles>

Schema:

<xs:schema xmlns:tns=”http://api.redbox.com/v3/Products/TitleList”

           elementFormDefault=”qualified”

           targetNamespace=”http://api.redbox.com/v3/Products/TitleList”

           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=”RedboxDoNotRentDate” 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=”PG13” />

      <xs:enumeration value=”R” />

      <xs:enumeration value=”NR” />

      <xs:enumeration value=”ALL AGES” />

      <xs:enumeration value=”TVY” />

      <xs:enumeration value=”TVY7” />

      <xs:enumeration value=”TVG” />

      <xs:enumeration value=”TVPG” />

      <xs:enumeration value=”TV14” />

      <xs:enumeration value=”TVMA” />

    </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>

### RecommendedProductsResponse

Example:

<RecommendedProductsResponse generatedAt=”1900-01-01T01:01:01-06:00” productType=”productType1” basedUponProductId=”basedUponProductId1” basedUponRentalHistory=”true” xmlns=”http://api.redbox.com/v3/Products”>

  <ProductId>ProductId1</ProductId>

  <ProductId>ProductId2</ProductId>

  <ProductId>ProductId3</ProductId>

</RecommendedProductsResponse>

Schema:

<xs:schema id=”RecommendedProductIDList”

    targetNamespace=”http://api.redbox.com/v3/Products”

    elementFormDefault=”qualified”

    xmlns=”http://api.redbox.com/v3/Products”

    xmlns:tns=”http://api.redbox.com/v3/Products”

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

>

  <xs:complexType name=”RecommendedProductsIDList” >

    <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:attribute name=”basedUponProductId” type=”xs:string” />

    <xs:attribute name=”basedUponRentalHistory” type=”xs:oolean” />

  </xs:complexType>

  <xs:element name=”RecommendedProductsResponse” type=”RecommendedProductsIDList” />

</xs:schema>

### Source Stores Schema

In this Source schema, the storeId attribute is the internal I 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:attribute name=”locationId” type=”xs:string” use=”optional” />

</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:

<StoreBulkList 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” locationId=”locationId1” >

      <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” locationId=”locationId1” >

      <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” locationId=”locationId2” >

      <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>

</StoreBulkList>

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”  schemaLocation=”..\..\OpenAPI-Paging.xsd” />

  <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=” optional” />

    <xs:attribute name=”commStatus” type=”tns:CommStatusEnum” use=” optional” />

  </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:attribute name=”locationId” type=”xs:string” use=”optional” />

  </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

**This schema has been deprecated and is no longer used.**

The following is an example Xml body:

<Inventory xmlns=”http://api-source.redbox.com/v3/Inventory”>

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

<ProductInventory productId=”1627aea5-8e0a-4371-9022-9b504344e724” lastUpdated=”1999-05-31T11:20:00” lastUpdatedEpoch=”9223372036854775807” inventoryStatus=”InStock” />

<ProductInventory productId=”1627aea5-8e0a-4371-9022-9b504344e724” lastUpdated=”1999-05-31T11:20:00” lastUpdatedEpoch=”9223372036854775807” inventoryStatus=”InStock” />

</StoreInventory>

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

<ProductInventory productId=”1627aea5-8e0a-4371-9022-9b504344e724” lastUpdated=”1999-05-31T11:20:00” lastUpdatedEpoch=”9223372036854775807” inventoryStatus=”InStock” />

<ProductInventory productId=”1627aea5-8e0a-4371-9022-9b504344e724” lastUpdated=”1999-05-31T11:20:00” lastUpdatedEpoch=”9223372036854775807” inventoryStatus=”InStock” />

</StoreInventory>

</Inventory>

The following is the response Xml Schema:

<xs:schema xmlns:tns=”http://api-source.redbox.com/v3/Inventory”

           elementFormDefault=”qualified”

           targetNamespace=”http://api-source.redbox.com/v3/Inventory”

           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=”xs:string” use=”required” />

  </xs:complexType>

  <xs:complexType name=”ProductInventory”>

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

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

    <xs:attribute name=”lastUpdatedEpoch” type=”xs:long” 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).

The following is an example response Xml body:

<StoreLookupList 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” locationId=”locationId1” >

      <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” locationId=”locationId1” >

      <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” locationId=”locationId2” >

      <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 6.1.11.

<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

The following is an example response Xml body:

<Inventory xmlns=”http://api.redbox.com/v3/Inventory”>

  <StoreInventory storeId=”storeId1” lastUpdated=”1900-01-01T01:01:01-06:00” lastUpdatedEpoch=”1”>

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

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

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

  </StoreInventory>

  <StoreInventory storeId=”storeId2” lastUpdated=”0001-01-01T00:00:00-06:00” lastUpdatedEpoch=”-9223372036854775807”>

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

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

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

  </StoreInventory>

  <StoreInventory storeId=”storeId3” lastUpdated=”9999-12-31T23:59:59.9999999-06:00” lastUpdatedEpoch=”9223372036854775807”>

    <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:

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

<xs:schema xmlns:tns=”http://api.redbox.com/v3/Inventory”

           elementFormDefault=”qualified”

           targetNamespace=”http://api.redbox.com/v3/Inventory”

           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” />

<!—

  This element is the root return element of a call to retrieve Inventory.

🡪

  <xs:element name=”Inventory” nillable=”true” type=”tns:InventoryLookupList” />

<!—

  This element defines the inventory of a single store.  It is used within <Inventory>

  and also used to POST inventory updates to the Apigee cache.

🡪

  <xs:element name=”StoreInventory” nillable=”false” type=”tns:ArrayOfProductInventory” />

  <xs:complexType name=”InventoryLookupList”>

    <xs:sequence>

      <xs:element minOccurs=”0” maxOccurs=”unbounded” ref=”tns:StoreInventory” />

      <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:attribute name=”lastUpdatedEpoch” type=”xs:long” use=”optional” />

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

    <xs:attribute name=”lastUpdatedEpoch” type=”xs:long” use=”optional” />

    🡪

  </xs:complexType>

  <xs:simpleType name=”InventoryStatusEnum”>

    <xs:restriction base=”xs:string”>

      <xs:enumeration value=”InStock” />

      <xs:enumeration value=”OutOfStock” />

      <xs:enumeration value=”LowStock” />

      <!—

      <xs:enumeration value=”Removed” />

      🡪

    </xs:restriction>

  </xs:simpleType>

</xs:schema>

### Source Top20 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=”xs: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

**This schema has been deprecated and is no longer used.**

Example Result from operation I2:

<StoreAndInventorySearchResult xmlns=”http://api.redbox.com/v3/Inventory”>

  <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/v3/Stores”>Label1</Label>

      <Retailer xmlns=”http://api.redbox.com/v3/Stores”>Retailer1</Retailer>

      <Channel xmlns=”http://api.redbox.com/v3/Stores”>Channel1</Channel>

      <DistanceFromSearchLocation xmlns=”http://api.redbox.com/v3/Stores”>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/v3/Stores”>

        <Address>Address2</Address>

        <Address2>Address22</Address2>

        <City>City2</City>

        <State>State2</State>

        <Zipcode>Zipcode2</Zipcode>

      </Location>

      <Label xmlns=”http://api.redbox.com/v3/Stores”>Label2</Label>

      <Retailer xmlns=”http://api.redbox.com/v3/Stores”>Retailer2</Retailer>

      <Channel xmlns=”http://api.redbox.com/v3/Stores”>Channel2</Channel>

      <DistanceFromSearchLocation xmlns=”http://api.redbox.com/v3/Stores”>-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/v3/Stores”>

        <Address>Address3</Address>

        <Address2>Address23</Address2>

        <City>City3</City>

        <State>State3</State>

        <Zipcode>Zipcode3</Zipcode>

      </Location>

      <Label xmlns=”http://api.redbox.com/v3/Stores”>Label3</Label>

      <Retailer xmlns=”http://api.redbox.com/v3/Stores”>Retailer3</Retailer>

      <Channel xmlns=”http://api.redbox.com/v3/Stores”>Channel3</Channel>

      <DistanceFromSearchLocation xmlns=”http://api.redbox.com/v3/Stores”>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/v3/Inventory”

           elementFormDefault=”qualified”

           targetNamespace=”http://api.redbox.com/v3/Inventory”

           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/v3/Stores” />

  <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

**This is the V2 schema, the V3 schema is at: 6.1.50**

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 6.1.3 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

Sample:

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

<Cart xmlns=”http://api.redbox.com/v4/Reservations”>

  <UserId>UserId1</UserId>

  <StoreRef>StoreRef1</StoreRef>

  <Device>Device1</Device>

  <ClientIPAddr>ClientIPAddr1</ClientIPAddr>

  <PaymentMethod applyCreditsIfAvailable=”false”>

    <RegisteredCreditCard>

      <CardId>CardId1</CardId>

      <CardCVV>CardCVV1</CardCVV>

    </RegisteredCreditCard>

  </PaymentMethod>

  <ProductRef>ProductRef1</ProductRef>

  <ProductRef>ProductRef2</ProductRef>

  <ProductRef>ProductRef3</ProductRef>

</Cart>

Schema:

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

<xs:schema

    xmlns:tns=”http://api.redbox.com/v4/Reservations”

    elementFormDefault=”qualified”

    targetNamespace=”http://api.redbox.com/v4/Reservations”

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

>

    <xs:element name=”Cart” nillable=”true” type=”tns:Cart” />

    <xs:complexType name=”CartWithNoProductRefs”>

        <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=”Device” type=”xs:string” />

  <xs:element minOccurs=”0” maxOccurs=”1” name=”ClientIPAddr” type=”xs:string” />

            <xs:element minOccurs=”1” maxOccurs=”1” name=”PaymentMethod” type=”tns:PaymentMethodType” />

            <!—

            <xs:element minOccurs=”0” maxOccurs=”1” name=”DiscountsApplied” type=”tns:ArrayOfDiscountApplication” />

🡪

        </xs:sequence>

    </xs:complexType>

    <xs:complexType name=”PaymentMethodType”>

        <xs:choice>

            <xs:element name=”RegisteredCreditCard” type=”tns:RegisteredCreditCardType” />

        </xs:choice>

        <xs:attribute name=”applyCreditsIfAvailable” type=”xs:oolean” use=”optional” default=”false” />

    </xs:complexType>

    <xs:complexType name=”RegisteredCreditCardType” >

        <xs:sequence >

            <xs:element minOccurs=”0” maxOccurs=”1” name=”CardId” type=”xs:string” />

            <xs:element minOccurs=”0” maxOccurs=”1” name=”CardCVV” type=”xs:string” />

        </xs:sequence>

    </xs:complexType>

    <xs:complexType name=”Cart” >

        <xs:complexContent >

            <xs:extension base=”tns:CartWithNoProductRefs” >

                <xs:sequence >

                    <xs:element minOccurs=”0” maxOccurs=”unbounded” name=”ProductRef” type=”xs:string” />

                </xs:sequence>

            </xs:extension>

        </xs:complexContent>

    </xs:complexType>

    <!--  This section of declarations commented out for now, until we decide how to handle other forms of discount 🡪

    <!—

    <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.

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

<CartValidation xmlns=”http://api.redbox.com/v4/Reservations”>

  <UserId>UserId1</UserId>

  <StoreRef>StoreRef1</StoreRef>

  <Device>Device1</Device>

  <PaymentMethod applyCreditsIfAvailable=”false”>

    <RegisteredCreditCard>

      <CardId>CardId1</CardId>

      <CardCVV>CardCVV1</CardCVV>

    </RegisteredCreditCard>

  </PaymentMethod>

  <ProductRef>ProductRef1</ProductRef>

  <ProductRef>ProductRef2</ProductRef>

  <ProductRef>ProductRef3</ProductRef>

  <ValidationErrors>

    <Message Category=”Category1” Error=”Error1”>

      <ProductRef>ProductRef1</ProductRef>

      <ProductRef>ProductRef2</ProductRef>

      <ProductRef>ProductRef3</ProductRef>

    </Message>

    <Message Category=”Category2” Error=”Error2”>

    </Message>

    <Message Category=”Category3” Error=”Error3”>

      <ProductRef>ProductRef7</ProductRef>

      <ProductRef>ProductRef8</ProductRef>

      <ProductRef>ProductRef9</ProductRef>

    </Message>

  </ValidationErrors>

</CartValidation>

Schema:

The schema for CartValidation extends the Cart schema (6.1.18). It uses a type (CartValidationsType)defined in a separate schema file (Reservations\_CartValidations\_Def\_V4.xsd) which follows the main schema:

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

<xs:schema

    xmlns:tns=”http://api.redbox.com/v4/Reservations”

    elementFormDefault=”qualified”

    targetNamespace=”http://api.redbox.com/v4/Reservations”

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

>

    <xs:include schemaLocation=”Reservations\_CartValidations\_Def\_V4.xsd” />

    <xs:include schemaLocation=”Reservations\_Cart\_V4.xsd” />

    <xs:element name=”CartValidation” nillable=”true” type=”tns:CartValidation” />

    <xs:complexType name=”CartValidation” >

        <xs:complexContent >

            <xs:extension base=”tns:Cart” >

                <xs:sequence>

                    <xs:element minOccurs=”1” maxOccurs=”1” name=”ValidationErrors” type=”tns:CartValidationsType” />

                </xs:sequence>

            </xs:extension>

        </xs:complexContent>

    </xs:complexType>

</xs:schema>

Reservations\_CartValidationsDef\_V3.xsd:

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

<xs:schema xmlns:tns=”http://api.redbox.com/v4/Reservations”

   elementFormDefault=”qualified”

   targetNamespace=”http://api.redbox.com/v4/Reservations”

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

   >

<xs:complexType name=”CartValidationsType”>

<xs:sequence>

<xs:element minOccurs=”0” maxOccurs=”unbounded” name=”Message” type=”tns:ErrMessage” />

</xs:sequence>

</xs:complexType>

<xs:complexType name=”ErrMessage”>

<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

**This schema has been deprecated and is no longer used.**

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

<PricedCart xmlns=”http://api.redbox.com/v4/Reservations”>

  <UserId>UserId1</UserId>

  <StoreRef>StoreRef1</StoreRef>

  <Device>Device1</Device>

  <ClientIPAddr>ClientIPAddr1</ClientIPAddr>

  <PaymentMethod applyCreditsIfAvailable=”false”>

    <RegisteredCreditCard>

      <CardId>CardId1</CardId>

      <CardCVV>CardCVV1</CardCVV>

    </RegisteredCreditCard>

  </PaymentMethod>

  <CartItems>

    <CartItem ProductRef=”ProductRef1” Name=”Name1” ProductType=”ProductType1” Price=”1” ImageUrl=”ImageUrl1” Discount=”1” DiscountedPrice=”1” ExtraPrice=”1” Format=”Format1” ItemStatus=”Valid” Rating=”G” />

    <CartItem ProductRef=”ProductRef2” Name=”Name2” ProductType=”ProductType2” Price=”-79228162514264337593543950335” ImageUrl=”ImageUrl2” Discount=”-79228162514264337593543950335” DiscountedPrice=”-79228162514264337593543950335” ExtraPrice=”-79228162514264337593543950335” Format=”Format2” ItemStatus=”PricingMismatch” Rating=”PG” />

    <CartItem ProductRef=”ProductRef3” Name=”Name3” ProductType=”ProductType3” Price=”79228162514264337593543950335” ImageUrl=”ImageUrl3” Discount=”79228162514264337593543950335” DiscountedPrice=”79228162514264337593543950335” ExtraPrice=”79228162514264337593543950335” Format=”Format3” ItemStatus=”Exclude” Rating=”PG13” />

  </CartItems>

  <SubTotal>1</SubTotal>

  <DiscountedSubTotal>1</DiscountedSubTotal>

  <GrandTotal>1</GrandTotal>

  <Tax>1</Tax>

</PricedCart>

Schema:

This schema imports and uses the common ProductRatings schema (6.1.51).

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

<xs:schema

xmlns:tns=”http://api.redbox.com/v4/Reservations”

elementFormDefault=”qualified”

targetNamespace=”http://api.redbox.com/v4/Reservations”

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

xmlns:productRatings=”http://api.redbox.com/v3/ProductRatings”

>

<!—Importing the ProductRatings namespace to use the MPAA and ESRB ratings defined in that namespace. 🡪

<xs:import namespace=”http://api.redbox.com/v3/ProductRatings” />

<xs:include schemaLocation=”Reservations\_Cart\_v4.xsd” />

<xs:element name=”PricedCart” nillable=”true” type=”tns:PricedCart” />

<xs:complexType name=”PricedCart”>

<xs:complexContent >

<xs:extension base=”tns:CartWithNoProductRefs”>

<xs:sequence>

<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:sequence>

</xs:extension>

</xs:complexContent>

</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=”productRatings:CombinedRatings” 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: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

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

<PricedCartResponse xmlns=”http://api.redbox.com/v4/Reservations”>

  <Cart>

    <UserId>UserId1</UserId>

    <StoreRef>StoreRef1</StoreRef>

    <Device>Device1</Device>

  <ClientIPAddr>ClientIPAddr1</ClientIPAddr>

    <PaymentMethod applyCreditsIfAvailable=”false”>

      <RegisteredCreditCard>

        <CardId>CardId1</CardId>

        <CardCVV>CardCVV1</CardCVV>

      </RegisteredCreditCard>

    </PaymentMethod>

    <CartItems>

      <CartItem ProductRef=”ProductRef1” Name=”Name1” ProductType=”ProductType1” Price=”1” ImageUrl=”ImageUrl1” Discount=”1” DiscountedPrice=”1” ExtraPrice=”1” Format=”Format1” ItemStatus=”Valid” Rating=”G” />

      <CartItem ProductRef=”ProductRef2” Name=”Name2” ProductType=”ProductType2” Price=”-79228162514264337593543950335” ImageUrl=”ImageUrl2” Discount=”-79228162514264337593543950335” DiscountedPrice=”-79228162514264337593543950335” ExtraPrice=”-79228162514264337593543950335” Format=”Format2” ItemStatus=”PricingMismatch” Rating=”PG” />

      <CartItem ProductRef=”ProductRef3” Name=”Name3” ProductType=”ProductType3” Price=”79228162514264337593543950335” ImageUrl=”ImageUrl3” Discount=”79228162514264337593543950335” DiscountedPrice=”79228162514264337593543950335” ExtraPrice=”79228162514264337593543950335” Format=”Format3” ItemStatus=”Exclude” Rating=”PG13” />

    </CartItems>

    <SubTotal>1</SubTotal>

    <DiscountedSubTotal>1</DiscountedSubTotal>

    <GrandTotal>1</GrandTotal>

    <Tax>1</Tax>

  </Cart>

  <ValidationErrors>

    <Message Category=”Category1” Error=”Error1”>

      <ProductRef>ProductRef1</ProductRef>

      <ProductRef>ProductRef2</ProductRef>

      <ProductRef>ProductRef3</ProductRef>

    </Message>

    <Message Category=”Category2” Error=”Error2”>

      <ProductRef>ProductRef4</ProductRef>

      <ProductRef>ProductRef5</ProductRef>

      <ProductRef>ProductRef6</ProductRef>

    </Message>

    <Message Category=”Category3” Error=”Error3”>

      <ProductRef>ProductRef7</ProductRef>

      <ProductRef>ProductRef8</ProductRef>

      <ProductRef>ProductRef9</ProductRef>

    </Message>

  </ValidationErrors>

</PricedCartResponse>

Schema:

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

<xs:schema xmlns:tns=”http://api.redbox.com/v4/Reservations”

           elementFormDefault=”qualified”

           targetNamespace=”http://api.redbox.com/v4/Reservations”

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

           >

    <xs:include schemaLocation=”Reservations\_CartValidations\_Def\_V4.xsd” />

    <xs:include schemaLocation=”Reservations\_PricedCart\_V4.xsd” />

    <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=”ValidationErrors” type=”tns:CartValidationsType” />

        </xs:sequence>

    </xs:complexType>

</xs:schema>

### ReservationResponse Schema

Example:

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

<ReservationResponse xmlns=”http://api.redbox.com/v4/Reservations”>

  <Cart>

    <UserId>UserId1</UserId>

    <StoreRef>StoreRef1</StoreRef>

    <Device>Device1</Device>

  <ClientIPAddr>ClientIPAddr1</ClientIPAddr>

    <PaymentMethod applyCreditsIfAvailable=”false”>

      <RegisteredCreditCard>

        <CardId>CardId1</CardId>

        <CardCVV>CardCVV1</CardCVV>

      </RegisteredCreditCard>

    </PaymentMethod>

    <CartItems>

      <CartItem ProductRef=”ProductRef1” Name=”Name1” ProductType=”ProductType1” Price=”1” ImageUrl=”ImageUrl1” Discount=”1” DiscountedPrice=”1” ExtraPrice=”1” Format=”Format1” ItemStatus=”Valid” Rating=”G” />

      <CartItem ProductRef=”ProductRef2” Name=”Name2” ProductType=”ProductType2” Price=”-79228162514264337593543950335” ImageUrl=”ImageUrl2” Discount=”-79228162514264337593543950335” DiscountedPrice=”-79228162514264337593543950335” ExtraPrice=”-79228162514264337593543950335” Format=”Format2” ItemStatus=”PricingMismatch” Rating=”PG” />

      <CartItem ProductRef=”ProductRef3” Name=”Name3” ProductType=”ProductType3” Price=”79228162514264337593543950335” ImageUrl=”ImageUrl3” Discount=”79228162514264337593543950335” DiscountedPrice=”79228162514264337593543950335” ExtraPrice=”79228162514264337593543950335” Format=”Format3” ItemStatus=”Exclude” Rating=”PG13” />

    </CartItems>

    <SubTotal>1</SubTotal>

    <DiscountedSubTotal>1</DiscountedSubTotal>

    <GrandTotal>1</GrandTotal>

    <Tax>1</Tax>

  </Cart>

  <ValidationErrors>

    <Message Category=”Category1” Error=”Error1”>

      <ProductRef>ProductRef1</ProductRef>

      <ProductRef>ProductRef2</ProductRef>

      <ProductRef>ProductRef3</ProductRef>

    </Message>

    <Message Category=”Category2” Error=”Error2”>

      <ProductRef>ProductRef4</ProductRef>

      <ProductRef>ProductRef5</ProductRef>

      <ProductRef>ProductRef6</ProductRef>

    </Message>

    <Message Category=”Category3” Error=”Error3”>

      <ProductRef>ProductRef7</ProductRef>

      <ProductRef>ProductRef8</ProductRef>

      <ProductRef>ProductRef9</ProductRef>

    </Message>

  </ValidationErrors>

  <ReservationId>ReservationId1</ReservationId>

  <PickupBy>PickupBy1</PickupBy>

</ReservationResponse>

Schema:

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

<xs:schema xmlns:tns=”http://api.redbox.com/v4/Reservations”

   elementFormDefault=”qualified”

   targetNamespace=”http://api.redbox.com/v4/Reservations”

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

   >

<xs:include schemaLocation=”Reservations\_PricedCartResponse\_V4.xsd” />

<xs:element name=”ReservationResponse” nillable=”true” type=”tns:ReservationResponse” />

<xs:complexType name=”ReservationResponse”>

<xs:complexContent>

<xs:extension base=”tns:PricedCartResponse” >

<xs:sequence>

<xs:element name=”ReservationId” type=”xs:string” />

<xs:element name=”PickupBy” type=”xs:string” />

</xs:sequence>

</xs:extension>

</xs:complexContent>

</xs:complexType>

</xs:schema>

### BasicProfile Schema

Example BasicProfile:

<CustomerAccountProfile xmlns=”http://api.redbox.com/v5/CustomerAccount”>

  <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>

  <CustomerNumber>CustomerNumber1</CustomerNumber>

  <DisplayName>DisplayName1</DisplayName>

  <EmailAddress>EmailAddress1</EmailAddress>

  <FirstName>FirstName1</FirstName>

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

  <LastUpdatedTime>1900-01-01T01:01:01-06:00</LastUpdatedTime>

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

  <LastName>LastName1</LastName>

  <MobilePhoneNumber>MobilePhoneNumber1</MobilePhoneNumber>

  <StateCd>StateCd1</StateCd>

  <StreetAddressLine1>StreetAddressLine11</StreetAddressLine1>

  <StreetAddressLine2>StreetAddressLine21</StreetAddressLine2>

  <ZipPostalCode>ZipPostalCode1</ZipPostalCode>

  <ReviewTOSAccepted>true</ReviewTOSAccepted>

  <Extensions />

</CustomerAccountProfile>

Schema:

<xs:schema id=”CustomerAccountProfile”

    targetNamespace=”http://api.redbox.com/v5/CustomerAccount”

    elementFormDefault=”qualified”

    xmlns=”http://api.redbox.com/v5/CustomerAccount”

    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” schemaLocation=”..\..\OpenAPI-Paging.xsd”/>

  <xs:complexType name=”CA\_CustomerAccountBasicProfile” >

    <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” minOccurs=”0 “ type=”xs:string”/>

      <xs:element name=”CustomerNumber” type=”xs:string” />

      <!--<xs:element name=”DefaultAccountNumber” type=”xs:string” minOccurs=”0” />🡪

      <xs:element name=”DisplayName” minOccurs=”0” type=”xs:string”/>

      <xs:element name=”EmailAddress” type=”xs:string”/>

      <xs:element name=”FirstName” minOccurs=”0” type=”xs:string” />

      <xs:element name=”IsActive” type=”xs:oolean” nillable=”true”/>

      <xs:element name=”LastUpdatedTime” type=”xs:dateTime”/>

      <xs:element name=”IsMobileUser” type=”xs:oolean” nillable=”true” />

      <xs:element name=”LastName” minOccurs=”0” type=”xs:string”/>

      <xs:element name=”MobilePhoneNumber” minOccurs=”0” type=”xs:string” />

      <xs:element name=”StateCd” minOccurs=”0” type=”xs:string” />

      <xs:element name=”StreetAddressLine1” minOccurs=”0” type=”xs:string”/>

      <xs:element name=”StreetAddressLine2” minOccurs=”0” type=”xs:string” />

      <xs:element name=”ZipPostalCode” type=”xs:string” />

      <xs:element name=”ReviewTOSAccepted” type=”xs:oolean” nillable=”true”/>

      <xs:element name=”Extensions” type=”oai:ExtensionData” minOccurs=”0” />

    </xs:sequence>

  </xs:complexType>

  <xs:element name=”CustomerAccountProfile” type=”CA\_CustomerAccountBasicProfile” />

</xs:schema>

### Preferences

Example:

<Preferences lastUpdated=”1900-01-01T01:01:01-06:00” xmlns=”http://api.redbox.com/v5/CustomerAccount”>

  <NotificationPreferences>

    <OptInSetting setting=”setting1” value=”true” />

    <OptInSetting setting=”setting2” value=”false” />

    <OptInSetting setting=”setting3” value=”true” />

  </NotificationPreferences>

  <FavoriteStores>

    <StoreId>StoreId1</StoreId>

    <StoreId>StoreId2</StoreId>

    <StoreId>StoreId3</StoreId>

  </FavoriteStores>

  <RecentStores>

    <StoreId>StoreId1</StoreId>

    <StoreId>StoreId2</StoreId>

    <StoreId>StoreId3</StoreId>

  </RecentStores>

  <GamePreferences>

    <GenrePreferences>

      <Genre>Genre1</Genre>

      <Genre>Genre2</Genre>

      <Genre>Genre3</Genre>

    </GenrePreferences>

    <FormatPreferences>

      <Format>Format1</Format>

      <Format>Format2</Format>

      <Format>Format3</Format>

    </FormatPreferences>

  </GamePreferences>

  <MoviePreferences>

    <GenrePreferences>

      <Genre>Genre4</Genre>

      <Genre>Genre5</Genre>

      <Genre>Genre6</Genre>

    </GenrePreferences>

    <FormatPreferences>

      <Format>Format4</Format>

      <Format>Format5</Format>

      <Format>Format6</Format>

    </FormatPreferences>

  </MoviePreferences>

  <InterestedComingSoonTitles>

    <ProductId>ProductId1</ProductId>

    <ProductId>ProductId2</ProductId>

    <ProductId>ProductId3</ProductId>

  </InterestedComingSoonTitles>

  <Extensions />

</Preferences>

Schema:

<xs:schema id=”CA\_Preferences\_V5”

    targetNamespace=”http://api.redbox.com/v5/CustomerAccount”

    elementFormDefault=”qualified”

    xmlns=”http://api.redbox.com/v5/CustomerAccount”

    xmlns:tns=”http://api.redbox.com/v5/CustomerAccount”

    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:include schemaLocation=”CA\_NotificationPreferencesType\_V5.xsd” id=”CA\_NotificationPreferencesType\_V5” />

  <xs:complexType name=”CA\_Preferences” >

    <xs:sequence>

      <xs:element name=”NotificationPreferences” type=”tns:CA\_NotificationPreferencesType” />

<!--      <xs:element name=”OptInForBetaTesting” type=”xs:oolean”/>

      <xs:element name=”OptInForEmailReceipts” type=”xs:oolean”/>

      <xs:element name=”OptInForSmsMessages” type=”xs:oolean”/>

🡪

      <xs:element name=”FavoriteStores” type=”tns:ArrayOfStoreIds” />

      <xs:element name=”RecentStores” 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=”ArrayOfFormats”  >

    <xs:sequence>

      <xs:element minOccurs=”0” maxOccurs=”unbounded” name=”Format” type=”xs:string” />

    </xs:sequence>

  </xs:complexType>

  <xs:complexType name=”ProductPreferencesType” >

    <xs:sequence>

      <xs:element name=”GenrePreferences” type=”tns:ArrayOfGenres” />

      <xs:element name=”FormatPreferences” type=”tns:ArrayOfFormats” />

    </xs:sequence>

<!—

    <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:

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

<RentalHistory lastUpdated=”1900-01-01T01:01:01-06:00” xmlns=”http://api.redbox.com/v5/CustomerAccount”>

  <RentalRecord productId=”productId1”>

    <TransactionNumber>TransactionNumber1</TransactionNumber>

    <ProductTitle>ProductTitle1</ProductTitle>

    <RentedDate>1900-01-01</RentedDate>

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

    <Detail>Detail1</Detail>

    <FormatType>FormatType1</FormatType>

    <RentalStatus>RentalStatus1</RentalStatus>

    <Extensions />

  </RentalRecord>

  <RentalRecord productId=”productId2”>

    <TransactionNumber>TransactionNumber2</TransactionNumber>

    <ProductTitle>ProductTitle2</ProductTitle>

    <RentedDate>0001-01-01</RentedDate>

    <ReturnedDate>1900-01-01</ReturnedDate>

    <Detail>Detail2</Detail>

    <FormatType>FormatType2</FormatType>

    <RentalStatus>RentalStatus1</RentalStatus>

    <Extensions />

  </RentalRecord>

  <RentalRecord productId=”productId3”>

    <TransactionNumber>TransactionNumber3</TransactionNumber>

    <ProductTitle>ProductTitle3</ProductTitle>

    <RentedDate>9999-12-31</RentedDate>

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

    <Detail>Detail3</Detail>

    <FormatType>FormatType3</FormatType>

    <RentalStatus>RentalStatus1</RentalStatus>

    <Extensions />

  </RentalRecord>

</RentalHistory>

Schema:

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

<xs:schema id=”CA\_RentalHistory\_V5”

    targetNamespace=”http://api.redbox.com/v5/CustomerAccount”

    elementFormDefault=”qualified”

    xmlns=”http://api.redbox.com/v5/CustomerAccount”

    xmlns:tns=”http://api.redbox.com/v5/CustomerAccount”

    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=”TransactionNumber” type=”xs:string” />

      <xs:element name=”ProductTitle” type=”xs:string”/>

      <xs:element name=”RentedDate”  type=”xs:date”/>

      <xs:element name=”ReturnedDate” nillable=”true” type=”xs:date” />

      <xs:element name=”Detail” type=”xs:string” minOccurs=”0” maxOccurs=”1”/>

      <xs:element name=”FormatType” type=”xs:string” />

      <xs:element name=”RentalStatus” 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/v3/CustomerAccount”>

  <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\_V3”

    targetNamespace=”http://api.redbox.com/v3/CustomerAccount”

    elementFormDefault=”qualified”

    xmlns=”http://api.redbox.com/v3/CustomerAccount”

    xmlns:tns=”http://api.redbox.com/v3/CustomerAccount”

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

>

  <xs:include schemaLocation=”CA\_QueueInfo\_V3.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/v3/CustomerAccount”>

  <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\_V3”

    targetNamespace=”http://api.redbox.com/v3/CustomerAccount”

    elementFormDefault=”qualified”

    xmlns=”http://api.redbox.com/v3/CustomerAccount”

    xmlns:tns=”http://api.redbox.com/v3/CustomerAccount”

    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

\*\*\* DEPRECATED. USE 6.1.8 INSTEAD \*\*\*\*

Example:

<RecommendedProductsResponse generatedAt=”1900-01-01T01:01:01-06:00” productType=”productType1” basedUponProductId=”basedUponProductId1” basedUponRentalHistory=”true” xmlns=”http://api.redbox.com/v5/CustomerAccount”>

  <ProductId>ProductId1</ProductId>

  <ProductId>ProductId2</ProductId>

  <ProductId>ProductId3</ProductId>

</RecommendedProductsResponse>

Schema:

<xs:schema id=”CA\_Recommendations\_V5”

    targetNamespace=”http://api.redbox.com/v5/CustomerAccount”

    elementFormDefault=”qualified”

    xmlns=”http://api.redbox.com/v5/CustomerAccount”

    xmlns:mstns=”http://api.redbox.com/v5/CustomerAccount”

    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:attribute name=”basedUponProductId” type=”xs:string” />

    <xs:attribute name=”basedUponRentalHistory” type=”xs:oolean” />

  </xs:complexType>

  <xs:element name=”RecommendedProductsResponse” type=”CA\_RecommendedProductsResponse” />

</xs:schema>

### AccountListResponse

Example:

<AccountList lastUpdated=”1900-01-01T01:01:01-06:00” xmlns=”http://api.redbox.com/v5/CustomerAccount”>

  <Account isPreferredAccount=”false”>

    <AccountNumber>AccountNumber1</AccountNumber>

    <AccountAlias>AccountAlias1</AccountAlias>

    <FirstName>FirstName1</FirstName>

    <LastName>LastName1</LastName>

    <LastFour>LastFour1</LastFour>

    <ExpirationMonth>ExpirationMonth1</ExpirationMonth>

    <ExpirationYear>ExpirationYear1</ExpirationYear>

    <AVSChecked>true</AVSChecked>

    <CVVChecked>true</CVVChecked>

    <AccountBillingAddress>

      <CityName>CityName1</CityName>

      <StateCd>StateCd1</StateCd>

      <StreetAddressLine1>StreetAddressLine11</StreetAddressLine1>

      <StreetAddressLine2>StreetAddressLine21</StreetAddressLine2>

      <ZipPostalCode>ZipPostalCode1</ZipPostalCode>

    </AccountBillingAddress>

    <Extensions />

  </Account>

  <Account isPreferredAccount=”true”>

    <AccountNumber>AccountNumber2</AccountNumber>

    <AccountAlias>AccountAlias2</AccountAlias>

    <FirstName>FirstName2</FirstName>

    <LastName>LastName2</LastName>

    <LastFour>LastFour2</LastFour>

    <ExpirationMonth>ExpirationMonth2</ExpirationMonth>

    <ExpirationYear>ExpirationYear2</ExpirationYear>

    <AVSChecked>false</AVSChecked>

    <CVVChecked>false</CVVChecked>

    <AccountBillingAddress>

      <CityName>CityName2</CityName>

      <StateCd>StateCd2</StateCd>

      <StreetAddressLine1>StreetAddressLine12</StreetAddressLine1>

      <StreetAddressLine2>StreetAddressLine22</StreetAddressLine2>

      <ZipPostalCode>ZipPostalCode2</ZipPostalCode>

    </AccountBillingAddress>

    <Extensions />

  </Account>

  <Account isPreferredAccount=”false”>

    <AccountNumber>AccountNumber3</AccountNumber>

    <AccountAlias>AccountAlias3</AccountAlias>

    <FirstName>FirstName3</FirstName>

    <LastName>LastName3</LastName>

    <LastFour>LastFour3</LastFour>

    <ExpirationMonth>ExpirationMonth3</ExpirationMonth>

    <ExpirationYear>ExpirationYear3</ExpirationYear>

    <AVSChecked>true</AVSChecked>

    <CVVChecked>true</CVVChecked>

    <AccountBillingAddress>

      <CityName>CityName3</CityName>

      <StateCd>StateCd3</StateCd>

      <StreetAddressLine1>StreetAddressLine13</StreetAddressLine1>

      <StreetAddressLine2>StreetAddressLine23</StreetAddressLine2>

      <ZipPostalCode>ZipPostalCode3</ZipPostalCode>

    </AccountBillingAddress>

    <Extensions />

  </Account>

</AccountList>

Schema:

<xs:schema id=”CA\_AccountList\_V5”

    targetNamespace=”http://api.redbox.com/v5/CustomerAccount”

    elementFormDefault=”qualified”

    xmlns=”http://api.redbox.com/v3/CustomerAccount”

    xmlns:tns=”http://api.redbox.com/v5/CustomerAccount”

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

>

  <xs:include schemaLocation=”CA\_AccountInfo\_V5.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 isPreferredAccount=”false” avsChecked=”true” cvvChecked=”true” cvv=”cvv1” xmlns=”http://api.redbox.com/v5/CustomerAccount”>

  <AccountNumber>AccountNumber1</AccountNumber>

  <AccountAlias>AccountAlias1</AccountAlias>

  <FirstName>FirstName1</FirstName>

  <LastName>LastName1</LastName>

  <LastFour>LastFour1</LastFour>

  <ExpirationMonth>ExpirationMonth1</ExpirationMonth>

  <ExpirationYear>ExpirationYear1</ExpirationYear>

  <AccountBillingAddress>

    <CityName>CityName1</CityName>

    <StateCd>StateCd1</StateCd>

    <StreetAddressLine1>StreetAddressLine11</StreetAddressLine1>

    <StreetAddressLine2>StreetAddressLine21</StreetAddressLine2>

    <ZipPostalCode>ZipPostalCode1</ZipPostalCode>

  </AccountBillingAddress>

  <Extensions />

</AccountInfo>

Schema:

The schema for AccountInfo is below. It uses a shared schema that describes the Billing Address for a card, CA\_BillingAddress\_V5.xsd, which is included in the schema files in Section 6.2.

<xs:schema id=”CA\_AccountInfo\_v5”

    targetNamespace=”http://api.redbox.com/v5/CustomerAccount”

    elementFormDefault=”qualified”

    xmlns=”http://api.redbox.com/v5/CustomerAccount”

    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” schemaLocation=”..\..\OpenAPI-Paging.xsd” />

  <xs:include schemaLocation=”CA\_BillingAddress\_v5.xsd” />

  <xs:complexType name=”CA\_AccountInfo” >

    <xs:sequence>

      <xs:element name=”AccountNumber” type=”xs:string” />

      <xs:element name=”AccountAlias” type=”xs:string”/>

      <xs:element name=”FirstName” type=”xs:string” />

      <xs:element name=”LastName” type=”xs:string” />

      <xs:element name=”LastFour” type=”xs:string”/>

      <xs:element name=”ExpirationMonth” type=”xs:string” />

      <xs:element name=”ExpirationYear” type=”xs:string” />

      <xs:element name=”AccountBillingAddress” type=”CA\_AccountBillingAddressType” minOccurs=”0” maxOccurs=”1” />

      <xs:element name=”Extensions” type=”oai:ExtensionData” minOccurs=”0”/>

    </xs:sequence>

    <xs:attribute name=”isPreferredAccount” type=”xs:oolean” use=”optional” default=”false” />

    <xs:attribute name=”avsChecked” type=”xs:oolean” use=”optional” />

    <xs:attribute name=”cvvChecked” type=”xs:oolean” use=”optional” />

    <xs:attribute name=”cvv” type=”xs:string” use=”optional” />

  </xs:complexType>

  <xs:element name=”AccountInfo” type=”CA\_AccountInfo” />

</xs:schema>

### CreditBalance

**This schema has been deprecated and no longer used.**

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=”1900-01-01T01:01:01-06:00” xmlns=”http://api.redbox.com/v5/CustomerAccount”>

  <Credit>

    <CreditId>CreditId1</CreditId>

    <CreditName>CreditName1</CreditName>

    <CreditType>CreditType1</CreditType>

    <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>

    <CreditType>CreditType2</CreditType>

    <EffectiveDate>1900-01-01</EffectiveDate>

    <ExpirationDate>1900-01-01</ExpirationDate>

    <Extensions />

  </Credit>

  <Credit>

    <CreditId>CreditId3</CreditId>

    <CreditName>CreditName3</CreditName>

    <CreditType>CreditType3</CreditType>

    <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\_V5”

    targetNamespace=”http://api.redbox.com/v5/CustomerAccount”

    elementFormDefault=”qualified”

    xmlns=”http://api.redbox.com/v5/CustomerAccount”

    xmlns:tns=”http://api.redbox.com/v5/CustomerAccount”

    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” schemaLocation=”..\..\OpenAPI-Paging.xsd” />

  <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=”CreditType” 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=”1900-01-01T01:01:01-06:00” xmlns=”http://api.redbox.com/v5/CustomerAccount”>

  <CreditHistory>

    <CreditId>CreditId1</CreditId>

    <CreditName>CreditName1</CreditName>

    <CreditType>CreditType1</CreditType>

    <EffectiveDate>1900-01-01</EffectiveDate>

    <CreditDisposition creditState=”Redeemed” dispositionDate=”1900-01-01”>

      <Redemption>

        <RedeemProductId>RedeemProductId1</RedeemProductId>

        <TransactionNumber>TransactionNumber1</TransactionNumber>

      </Redemption>

    </CreditDisposition>

    <Extensions />

  </CreditHistory>

  <CreditHistory>

    <CreditId>CreditId2</CreditId>

    <CreditName>CreditName2</CreditName>

    <CreditType>CreditType2</CreditType>

    <EffectiveDate>0001-01-01</EffectiveDate>

    <CreditDisposition creditState=”Expired” dispositionDate=”0001-01-01”>

    </CreditDisposition>

    <Extensions />

  </CreditHistory>

  <CreditHistory>

    <CreditId>CreditId3</CreditId>

    <CreditName>CreditName3</CreditName>

    <CreditType>CreditType3</CreditType>

    <EffectiveDate>9999-12-31</EffectiveDate>

    <CreditDisposition creditState=”Deleted” dispositionDate=”9999-12-31”>

    </CreditDisposition>

    <Extensions />

  </CreditHistory>

</CreditUseHistory>

Schema:

<xs:schema id=”CA\_CreditUseHistory\_V5”

    targetNamespace=”http://api.redbox.com/v5/CustomerAccount”

    elementFormDefault=”qualified”

    xmlns=”http://api.redbox.com/v5/CustomerAccount”

    xmlns:tns=”http://api.redbox.com/v5/CustomerAccount”

    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”  schemaLocation=”..\..\OpenAPI-Paging.xsd” />

  <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=”CreditType” type=”xs:string” />

      <xs:element name=”EffectiveDate” type=”xs:date” />

      <!--<xs:element name=”Quantity” type=”xs:int” />🡪

      <xs:element name=”CreditDisposition”>

        <xs:complexType>

          <xs:choice minOccurs=”0” maxOccurs=”1”>

            <xs:element name=”Redemption” type=”tns:RedemptionDetail” />

          </xs:choice>

          <xs:attribute name=”creditState” type=”tns:CreditStates” />

          <xs:attribute name=”dispositionDate” type=”xs:date” />

        </xs:complexType>

      </xs:element>

      <xs:element name=”Extensions” type=”oai:ExtensionData” />

    </xs:sequence>

  </xs:complexType>

  <xs:complexType name=”RedemptionDetail” >

    <xs:sequence>

      <xs:element name=”RedeemProductId” type=”xs:string”  />

      <xs:element name=”TransactionNumber” type=”xs:string”  />

    </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” saveToProfile=”true” cvv=”cvv1”

xmlns=”http://api.redbox.com/v5/CustomerAccount”>

  <Number>Number1</Number>

  <FirstName>FirstName1</FirstName>

  <LastName>LastName1</LastName>

  <Zip>Zip1</Zip>

  <Month>Month1</Month>

  <Year>Year1</Year>

  <Alias>Alias1</Alias>

  <AccountBillingAddress>

    <CityName>CityName1</CityName>

    <StateCd>StateCd1</StateCd>

    <StreetAddressLine1>StreetAddressLine11</StreetAddressLine1>

    <StreetAddressLine2>StreetAddressLine21</StreetAddressLine2>

    <ZipPostalCode>ZipPostalCode1</ZipPostalCode>

  </AccountBillingAddress>

</CreditCardInfo>

Schema:

The schema for CreditCardInfo is below. It uses a shared schema for the Billing Address element, which is contained in the file CA\_BillingAddress\_V5.xsd in the schema files in Section 6.2.

<xs:schema id=”CA\_CreditCardInfo\_V5”

    targetNamespace=”http://api.redbox.com/v5/CustomerAccount”

    elementFormDefault=”qualified”

    xmlns=”http://api.redbox.com/v5/CustomerAccount”

    xmlns:mstns=”http://api.redbox.com/v5/CustomerAccount”

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

>

  <xs:include schemaLocation=”CA\_BillingAddress\_V5.xsd” />

  <xs:complexType name=”CreditCardInfo”>

    <xs:sequence>

      <xs:element name=”Number” type=”xs:string” />

      <xs:element name=”FirstName” type=”xs:string” />

      <xs:element name=”LastName” 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=”Alias” type=”xs:string” />

      <xs:element name=”AccountBillingAddress” type=”CA\_AccountBillingAddressType” minOccurs=”0” maxOccurs=”1” />

    </xs:sequence>

    <xs:attribute name=”preferred” type=”xs:oolean” use=”optional” />

    <xs:attribute name=”saveToProfile” type=”xs:oolean” use=”optional” />

    <xs:attribute name=”cvv” type=”xs:string” 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

***This schema not used in R2 or R3; included for future reference.***

<?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=”optional” />

  </xs:complexType>

</xs:schema>

### StoreChangedEventMessage Schema

***This schema not used in R2 or R3; included for future reference.***

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

***This schema not used in R2 or R3; included for future reference.***

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

***This schema not used in R2 or R3; included for future reference.***

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

<xs:schema id=”ExternalItemChangesMessage”

    targetNamespace=”http://api.redbox.com/v3/OpenAPI/ItemChangeMessages”

    elementFormDefault=”qualified”

    xmlns=”http://api.redbox.com/v3/OpenAPI/ItemChangeMessages”

    xmlns:tns=”http://api.redbox.com/v3/OpenAPI/ItemChangeMessages”

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

>

  <xs:complexType name=”ItemChangesMessageBase” >

    <xs:attribute name=”dateTimeIssued” use=”required” type=”xs:dateTime” />

    <xs:attribute name=”dateTimeEpochIssued” use=”required” type=”xs:long” />

    <xs:attribute name=”source” type=”xs:string” use=”required” />

  </xs:complexType>

</xs:schema>

### StoreChangesMessage Schema

***This schema not used in R2 or R3; included for future reference.***

Example:

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

<StoreChangesMessage dateTimeIssued=”1900-01-01T01:01:01-06:00” dateTimeEpochIssued=”1” source=”source1” xmlns=”http://api.redbox.com/v3/OpenAPI/ItemChangeMessages”>

  <StoreChange storeId=”storeId1” storeType=”Indoor” lastUpdated=”1900-01-01T01:01:01-06:00” lastUpdatedEpoch=”1”>

    <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=”1900-01-01T01:01:01-06:00” lastUpdatedEpoch=”1” />

  <StoreChange storeId=”storeId2” storeType=”Outdoor” lastUpdated=”0001-01-01T00:00:00-06:00” lastUpdatedEpoch=”-9223372036854775807”>

    <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:

Note: in this schema, the xs:included schema, “ExternalItemChangesMessagev3.xsd”, is defined in 6.1.43.

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

<xs:schema id=”ExternalStoreChangesMessage”

    targetNamespace=”http://api.redbox.com/v3/OpenAPI/ItemChangeMessages”

    elementFormDefault=”qualified”

    xmlns=”http://api.redbox.com/v3/OpenAPI/ItemChangeMessages”

    xmlns:tns=”http://api.redbox.com/v3/OpenAPI/ItemChangeMessages”

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

>

  <xs:include schemaLocation=”ExternalItemChangesMessagev3.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:attribute name=”lastUpdatedEpoch” type=”xs:long” 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:attribute name=”lastUpdatedEpoch” type=”xs:long” 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

***This schema not used in R2 or R3; included for future reference.***

Example:

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

<InventoryChangesMessage dateTimeIssued=”1900-01-01T01:01:01-06:00” dateTimeEpochIssued=”1” source=”source1” xmlns=”http://api.redbox.com/v3/OpenAPI/ItemChangeMessages”>

  <InventoryRemoval storeId=”storeId1” productId=”productId1” lastUpdated=”1900-01-01T01:01:01-06:00” lastUpdatedEpoch=”1” />

  <InventoryChange inventoryAmountCode=”inventoryAmountCode1” storeId=”storeId1” productId=”productId1” lastUpdated=”1900-01-01T01:01:01-06:00” lastUpdatedEpoch=”1” />

  <InventoryRemoval storeId=”storeId2” productId=”productId2” lastUpdated=”0001-01-01T00:00:00-06:00” lastUpdatedEpoch=”-9223372036854775807” />

</InventoryChangesMessage>

Schema:

Note: in this schema, the xs:included schema, “ExternalItemChangesMessagev3.xsd”, is defined in 6.1.43.

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

<xs:schema id=”ExternalInventoryChangesMessage”

    targetNamespace=”http://api.redbox.com/v3/OpenAPI/ItemChangeMessages”

    elementFormDefault=”qualified”

    xmlns=”http://api.redbox.com/v3/OpenAPI/ItemChangeMessages”

    xmlns:tns=”http://api.redbox.com/v3/OpenAPI/ItemChangeMessages”

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

>

  <xs:include schemaLocation=”ExternalItemChangesMessagev3.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:attribute name=”lastUpdatedEpoch” type=”xs:long” 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:attribute name=”lastUpdatedEpoch” type=”xs:long” 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/v3/Products/Trailers”>

  <Trailer 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>

  </Trailer>

  <Trailer 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>

  </Trailer>

</Trailers>

Schema:

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

<xs:schema id=”Trailers”

         xmlns:tns=”http://api.redbox.com/v3/Products/Trailers”

         elementFormDefault=”qualified”

         targetNamespace=”http://api.redbox.com/v3/Products/Trailers”

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

>

  <xs:complexType name=”ArrayOfTrailers”>

    <xs:sequence>

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

    </xs:sequence>

  </xs:complexType>

  <xs:complexType name=”Trailer”>

    <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=”Trailers” type=”tns:ArrayOfTrailers” />

 </xs:schema>

### Customer Identity Schema

Sample Request:

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

<CustomerIdentity xmlns=”http://api.redbox.com/v5/CustomerAccount”>

  <ContactEmailAddress>ContactEmailAddress1</ContactEmailAddress>

  <FirstName>FirstName1</FirstName>

  <LastName>LastName1</LastName>

  <ZipPostalCode>ZipPostalCode1</ZipPostalCode>

  <Password>Password1</Password>

  <Extensions />

</CustomerIdentity>

Schema:

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

<xs:schema id=”CA\_CustomerIdentity\_V5”

      targetNamespace=”http://api.redbox.com/v5/CustomerAccount”

      elementFormDefault=”qualified”

      xmlns=”http://api.redbox.com/v5/CustomerAccount”

      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\_CustomerIdentity” >

      <xs:sequence>

        <xs:element name=”ContactEmailAddress” type=”xs:string” minOccurs=”1” maxOccurs=”1” />

        <xs:element name=”FirstName” type=”xs:string” minOccurs=”0” maxOccurs=”1” />

        <xs:element name=”LastName” type=”xs:string” minOccurs=”0” maxOccurs=”1” />

        <xs:element name=”ZipPostalCode” type=”xs:string” minOccurs=”1” maxOccurs=”1” />

        <xs:element name=”Password” type=”xs:string” minOccurs=”1” maxOccurs=”1” />

        <xs:element name=”Extensions” type=”oai:ExtensionData”  minOccurs=”0” maxOccurs=”1”/>

      </xs:sequence>

    </xs:complexType>

    <xs:element name=”CustomerIdentity” type=”CA\_CustomerIdentity” />

  </xs:schema>

### New Customer Creation Response Schema

Sample Response:

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

<NewCustomerCreationResponse xmlns=”http://api.redbox.com/v5/CustomerAccount”>

  <AccountCreationSource>AccountCreationSource1</AccountCreationSource>

  <CustomerNumber>CustomerNumber1</CustomerNumber>

  <ContactEmailAddress>ContactEmailAddress1</ContactEmailAddress>

  <LastUpdatedTime>1900-01-01T01:01:01-06:00</LastUpdatedTime>

  <Extensions />

</NewCustomerCreationResponse>

Schema:

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

<xs:schema id=”CA\_NewCustomerCreationResponse\_V5”

      targetNamespace=”http://api.redbox.com/v5/CustomerAccount”

      elementFormDefault=”qualified”

      xmlns=”http://api.redbox.com/v5/CustomerAccount”

      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\_NewCustomerCreationResponse” >

      <xs:sequence>

        <xs:element name=”AccountCreationSource” type=”xs:string” />

        <xs:element name=”CustomerNumber” type=”xs:string” />

        <xs:element name=”ContactEmailAddress” type=”xs:string”  />

        <xs:element name=”LastUpdatedTime” type=”xs:dateTime”/>

        <xs:element name=”Extensions” type=”oai:ExtensionData” />

      </xs:sequence>

    </xs:complexType>

    <xs:element name=”NewCustomerCreationResponse” type=”CA\_NewCustomerCreationResponse” />

  </xs:schema>

### V3 External Products Schema

This schema is used for both Movies and Games. Operations which return only Movies will not have any <Game> elements and vice versa.

This schema is used for both paged and non-paged versions. The <Paging> element is optional and is omitted when no paging information is present or required by the operation.

The schemas can be found in the attached schemas files. The schemas are:

* ProductListv3.xsd
* Productsv3.xsd

The following is an example response Xml body:

<Products lastUpdated=”1900-01-01T01:01:01-06:00” searchText=”searchText1” searchField=”searchField1” searchOperator=”searchOperator1” sortField=”sortField1” sortOrder=”Asc” xmlns=”http://api.redbox.com/v3/Products”>

  <Movie isClosedCaptioned=”true” format=”format1” productId=”productId1” websiteUrl=”http://uri1”>

    <Title>Title1</Title>

    <SortTitle>SortTitle1</SortTitle>

    <UppercaseTitle>UppercaseTitle1</UppercaseTitle>

    <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” />

    <Flags>

      <Flag type=”ComingSoon” beginDate=”1900-01-01” endDate=”1900-01-01” />

      <Flag type=”NewRelease” beginDate=”0001-01-01” endDate=”0001-01-01” />

      <Flag type=”LeavingSoon” beginDate=”9999-12-31” endDate=”9999-12-31” />

      <Flag type=”AvailableAtRedbox” beginDate=”1899-11-30” endDate=”1899-11-30” />

      <Flag type=”BackAgain” beginDate=”1900-02-02” endDate=”1900-02-02” />

    </Flags>

    <Extensions />

    <SynopsisShort>SynopsisShort1</SynopsisShort>

    <SynopsisLong>SynopsisLong1</SynopsisLong>

    <RedboxDoNotRentDate 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>

    <RatingContext ratingReason=”ratingReason1” ratingDescription=”ratingDescription1” ratedBy=”ratedBy1” />

    <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>

    <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” />

    <DomesticHomeVideoDistributor>DomesticHomeVideoDistributor1</DomesticHomeVideoDistributor>

    <DomesticTheatricalDistributor>DomesticTheatricalDistributor1</DomesticTheatricalDistributor>

    <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>

    <BoxOfficeRevenue>1</BoxOfficeRevenue>

    <OriginalLanguages>

      <Language>Language1</Language>

      <Language>Language2</Language>

      <Language>Language3</Language>

    </OriginalLanguages>

    <DubbedLanguages>

      <Language>Language1</Language>

      <Language>Language2</Language>

      <Language>Language3</Language>

    </DubbedLanguages>

    <RedboxGenresV2>

      <Genre>Genre4</Genre>

      <Genre>Genre5</Genre>

      <Genre>Genre6</Genre>

    </RedboxGenresV2>

    <ScreenFormat>ScreenFormat1</ScreenFormat>

  </Movie>

  <Game platform=”platform1” sub-platform=”sub-platform1” productId=”productId1” websiteUrl=”http://uri1”>

    <Title>Title1</Title>

    <SortTitle>SortTitle1</SortTitle>

    <UppercaseTitle>UppercaseTitle1</UppercaseTitle>

    <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” />

    <Flags>

      <Flag type=”ComingSoon” beginDate=”1900-01-01” endDate=”1900-01-01” />

      <Flag type=”NewRelease” beginDate=”0001-01-01” endDate=”0001-01-01” />

      <Flag type=”LeavingSoon” beginDate=”9999-12-31” endDate=”9999-12-31” />

      <Flag type=”AvailableAtRedbox” beginDate=”1899-11-30” endDate=”1899-11-30” />

      <Flag type=”BackAgain” beginDate=”1900-02-02” endDate=”1900-02-02” />

    </Flags>

    <Extensions />

    <SynopsisShort>SynopsisShort1</SynopsisShort>

    <SynopsisLong>SynopsisLong1</SynopsisLong>

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

    <Genres>

      <Genre>Genre7</Genre>

      <Genre>Genre8</Genre>

      <Genre>Genre9</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>

    <RatingContext ratingReason=”ratingReason2” ratingDescription=”ratingDescription2” ratedBy=”ratedBy2” />

    <Publisher>Publisher1</Publisher>

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

    <NumberOfConsolePlayers>NumberOfConsolePlayers1</NumberOfConsolePlayers>

    <NumberOfOnlinePlayers>NumberOfOnlinePlayers1</NumberOfOnlinePlayers>

  </Game>

  <Paging pageNum=”1” pageSize=”1” totalNumItems=”1” pageCount=”1” />

</Products>

### V3 External Top20 Schema

The following is an example response Xml body:

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

<Top20 period=”0” productType=”productType1” lastUpdated=”1900-01-01T01:01:01-06:00” xmlns=”http://api.redbox.com/v3/Products”>

  <Item isClosedCaptioned=”true” format=”format1” productId=”productId1” websiteUrl=”http://uri1”>

    <Title>Title1</Title>

    <SortTitle>SortTitle1</SortTitle>

    <UppercaseTitle>UppercaseTitle1</UppercaseTitle>

    <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” />

    <Flags>

      <Flag type=”type1” value=”value1” beginDate=”1900-01-01” endDate=”1900-01-01” />

      <Flag type=”type2” value=”value2” beginDate=”0001-01-01” endDate=”0001-01-01” />

      <Flag type=”type3” value=”value3” beginDate=”9999-12-31” endDate=”9999-12-31” />

    </Flags>

    <Extensions />

    <SynopsisShort>SynopsisShort1</SynopsisShort>

    <SynopsisLong>SynopsisLong1</SynopsisLong>

    <RedboxDoNotRentDate 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 xml:space=”default” href=”http://uri1” rel=”rel1” type=”type1” hreflang=”Token1” title=”title1” length=”1” xml:base=”http://uri1” xml:lang=”en” xmlns=”http://www.w3.org/2005/Atom”>text</link>

      <link xml:space=”preserve” href=”http://uri2” rel=”rel2” type=”type2” hreflang=”Token2” title=”title2” length=”79228162514264337593543950335” xml:base=”http://uri2” xml:lang=”fr” xmlns=”http://www.w3.org/2005/Atom”>text</link>

      <link xml:space=”default” href=”http://uri3” rel=”rel3” type=”type3” hreflang=”Token3” title=”title3” length=”2” xml:base=”http://uri3” xml:lang=”de” xmlns=”http://www.w3.org/2005/Atom”>text</link>

    </BoxArtImages>

    <RatingContext ratingReason=”ratingReason1” ratingDescription=”ratingDescription1” ratedBy=”ratedBy1” />

    <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>

    <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” />

    <DomesticHomeVideoDistributor>DomesticHomeVideoDistributor1</DomesticHomeVideoDistributor>

    <DomesticTheatricalDistributor>DomesticTheatricalDistributor1</DomesticTheatricalDistributor>

    <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>

    <BoxOfficeRevenue>1</BoxOfficeRevenue>

    <OriginalLanguages>

      <Language>Language1</Language>

      <Language>Language2</Language>

      <Language>Language3</Language>

    </OriginalLanguages>

    <DubbedLanguages>

      <Language>Language1</Language>

      <Language>Language2</Language>

      <Language>Language3</Language>

    </DubbedLanguages>

    <RedboxGenresV2>

      <Genre>Genre4</Genre>

      <Genre>Genre5</Genre>

      <Genre>Genre6</Genre>

    </RedboxGenresV2>

    <ScreenFormat>ScreenFormat1</ScreenFormat>

    <Top20Entry xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance” xsi:type=”p4:NewTop20Entry” position=”0” xmlns:p4=”http://api.redbox.com/v3/Top20” />

  </Item>

  <Item isClosedCaptioned=”false” format=”format2” productId=”productId2” websiteUrl=”http://uri2”>

    <Title>Title2</Title>

    <SortTitle>SortTitle2</SortTitle>

    <UppercaseTitle>UppercaseTitle2</UppercaseTitle>

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

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

    <Flags>

      <Flag type=”type4” value=”value4” beginDate=”1899-11-30” endDate=”1899-11-30” />

      <Flag type=”type5” value=”value5” beginDate=”1900-02-02” endDate=”1900-02-02” />

      <Flag type=”type6” value=”value6” beginDate=”0001-02-02” endDate=”0001-02-02” />

    </Flags>

    <Extensions />

    <SynopsisShort>SynopsisShort2</SynopsisShort>

    <SynopsisLong>SynopsisLong2</SynopsisLong>

    <RedboxDoNotRentDate>1900-01-01</RedboxDoNotRentDate>

    <Genres>

      <Genre>Genre7</Genre>

      <Genre>Genre8</Genre>

      <Genre>Genre9</Genre>

    </Genres>

    <BoxArtImages>

      <link xml:space=”preserve” href=”http://uri4” rel=”rel4” type=”type4” hreflang=”Token4” title=”title4” length=”79228162514264337593543950334” xml:base=”http://uri4” xml:lang=”da” xmlns=”http://www.w3.org/2005/Atom”>text</link>

      <link xml:space=”default” href=”http://uri5” rel=”rel5” type=”type5” hreflang=”Token5” title=”title5” length=”3” xml:base=”http://uri5” xml:lang=”el” xmlns=”http://www.w3.org/2005/Atom”>text</link>

      <link xml:space=”preserve” href=”http://uri6” rel=”rel6” type=”type6” hreflang=”Token6” title=”title6” length=”79228162514264337593543950333” xml:base=”http://uri6” xml:lang=”it” xmlns=”http://www.w3.org/2005/Atom”>text</link>

    </BoxArtImages>

    <RatingContext ratingReason=”ratingReason2” ratingDescription=”ratingDescription2” ratedBy=”ratedBy2” />

    <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>

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

    <ReleaseYear>0</ReleaseYear>

    <DomesticHomeVideoDistributor>DomesticHomeVideoDistributor2</DomesticHomeVideoDistributor>

    <DomesticTheatricalDistributor>DomesticTheatricalDistributor2</DomesticTheatricalDistributor>

    <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>

    <BoxOfficeRevenue>-2147483647</BoxOfficeRevenue>

    <OriginalLanguages>

      <Language>Language4</Language>

      <Language>Language5</Language>

      <Language>Language6</Language>

    </OriginalLanguages>

    <DubbedLanguages>

      <Language>Language4</Language>

      <Language>Language5</Language>

      <Language>Language6</Language>

    </DubbedLanguages>

    <RedboxGenresV2>

      <Genre>Genre10</Genre>

      <Genre>Genre11</Genre>

      <Genre>Genre12</Genre>

    </RedboxGenresV2>

    <ScreenFormat>ScreenFormat2</ScreenFormat>

    <Top20Entry xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance” xsi:type=”p4:NewTop20Entry” position=”65535” xmlns:p4=”http://api.redbox.com/v3/Top20” />

  </Item>

  <Item isClosedCaptioned=”true” format=”format3” productId=”productId3” websiteUrl=”http://uri3”>

    <Title>Title3</Title>

    <SortTitle>SortTitle3</SortTitle>

    <UppercaseTitle>UppercaseTitle3</UppercaseTitle>

    <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” />

    <Flags>

      <Flag type=”type7” value=”value7” beginDate=”9999-11-29” endDate=”9999-11-29” />

      <Flag type=”type8” value=”value8” beginDate=”1899-10-29” endDate=”1899-10-29” />

      <Flag type=”type9” value=”value9” beginDate=”1900-03-06” endDate=”1900-03-06” />

    </Flags>

    <Extensions />

    <SynopsisShort>SynopsisShort3</SynopsisShort>

    <SynopsisLong>SynopsisLong3</SynopsisLong>

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

    <Genres>

      <Genre>Genre13</Genre>

      <Genre>Genre14</Genre>

      <Genre>Genre15</Genre>

    </Genres>

    <BoxArtImages>

      <link xml:space=”default” href=”http://uri7” rel=”rel7” type=”type7” hreflang=”Token7” title=”title7” length=”4” xml:base=”http://uri7” xml:lang=”en-US” xmlns=”http://www.w3.org/2005/Atom”>text</link>

      <link xml:space=”preserve” href=”http://uri8” rel=”rel8” type=”type8” hreflang=”Token8” title=”title8” length=”79228162514264337593543950332” xml:base=”http://uri8” xml:lang=”en” xmlns=”http://www.w3.org/2005/Atom”>text</link>

      <link xml:space=”default” href=”http://uri9” rel=”rel9” type=”type9” hreflang=”Token9” title=”title9” length=”5” xml:base=”http://uri9” xml:lang=”fr” xmlns=”http://www.w3.org/2005/Atom”>text</link>

    </BoxArtImages>

    <RatingContext ratingReason=”ratingReason3” ratingDescription=”ratingDescription3” ratedBy=”ratedBy3” />

    <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>

    <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” />

    <DomesticHomeVideoDistributor>DomesticHomeVideoDistributor3</DomesticHomeVideoDistributor>

    <DomesticTheatricalDistributor>DomesticTheatricalDistributor3</DomesticTheatricalDistributor>

    <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>

    <BoxOfficeRevenue>2147483647</BoxOfficeRevenue>

    <OriginalLanguages>

      <Language>Language7</Language>

      <Language>Language8</Language>

      <Language>Language9</Language>

    </OriginalLanguages>

    <DubbedLanguages>

      <Language>Language7</Language>

      <Language>Language8</Language>

      <Language>Language9</Language>

    </DubbedLanguages>

    <RedboxGenresV2>

      <Genre>Genre16</Genre>

      <Genre>Genre17</Genre>

      <Genre>Genre18</Genre>

    </RedboxGenresV2>

    <ScreenFormat>ScreenFormat3</ScreenFormat>

    <Top20Entry xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance” xsi:type=”p4:NewTop20Entry” position=”1” xmlns:p4=”http://api.redbox.com/v3/Top20” />

  </Item>

</Top20>

The following are the response Xml Schemas. This schema re-uses the **Productsv3** schema defined in 6.1.49 External Product List.

* Top20List.xsd
* Top20Entry.xsd

### ProductRatings Schema

Whenever a schema references a Movie or Game ratings-board rating, the following enumerations will be used.

The schema is included in the file:

* Products\_Ratings\_v3.xsd

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

<xs:schema xmlns:tns=”http://api.redbox.com/v3/ProductRatings”

           elementFormDefault=”qualified”

           targetNamespace=”http://api.redbox.com/v3/ProductRatings”

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

>

  <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:enumeration value=”ALL AGES” />

      <xs:enumeration value=”TVY” />

      <xs:enumeration value=”TVY7” />

      <xs:enumeration value=”TVG” />

      <xs:enumeration value=”TVPG” />

      <xs:enumeration value=”TV14” />

      <xs:enumeration value=”TVMA” />

    </xs:restriction>

  </xs:simpleType>

  <xs:simpleType name=”ESRBRatings”>

    <xs:restriction base=”xs:string”>

      <xs:enumeration value=”EC” />

      <xs:enumeration value=”E” />

      <xs:enumeration value=”E 10+” />

      <xs:enumeration value=”T” />

      <xs:enumeration value=”M (17+)” />

      <xs:enumeration value=”RP” />

    </xs:restriction>

  </xs:simpleType>

  <xs:simpleType name=”CombinedRatings” >

    <xs:union memberTypes=”tns:ESRBRatings tns:MPAARatings” />

  </xs:simpleType>

</xs:schema>

### Product Metrics List Schema

This schema provides a list of product metrics per product. This schema will be used to provide metrics for both Movies and Games.

A product metric may represent information about the popularity, sales or any other quantifiable data that allows products to be compared. Metrics are typically not intrinsic to the product itself (contrasted against intrinsic metadata such as Title or Release Year). Specific metrics are not defined in this schema, but example metrics in R6 might be Ordinal Ranked Sales Volume and Weighted Rank Sales Volume.

Example output:

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

<ProductsMetrics lastUpdated=”1900-01-01T01:01:01-06:00” xmlns=”http://api.redbox.com/v3/Products”>

  <ProductMetrics productId=”productId1”>

    <Metric name=”name1” value=”value1” type=”type1” scope=”scope1” />

    <Metric name=”name2” value=”value2” type=”type2” scope=”scope2” />

    <Metric name=”name3” value=”value3” type=”type3” scope=”scope3” />

  </ProductMetrics>

  <ProductMetrics productId=”productId2”>

    <Metric name=”name4” value=”value4” type=”type4” scope=”scope4” />

    <Metric name=”name5” value=”value5” type=”type5” scope=”scope5” />

    <Metric name=”name6” value=”value6” type=”type6” scope=”scope6” />

  </ProductMetrics>

  <ProductMetrics productId=”productId3”>

    <Metric name=”name7” value=”value7” type=”type7” scope=”scope7” />

    <Metric name=”name8” value=”value8” type=”type8” scope=”scope8” />

    <Metric name=”name9” value=”value9” type=”type9” scope=”scope9” />

  </ProductMetrics>

</ProductsMetrics>

The attributes for each metric have the following definition:

* Name: Friendly name of the metric (Examples: Sales, Search Popularity)
* Value: the computed value of the metric
* Type: How the metric was computed (ex: OrdinalRank, WeightedRank, Sum)
* Scope: Defines the scope of data used to compute the metric (ex: Last30Days, Last Year, Trailing 12 months)

The schema is included in the file:

* Products\_Metrics\_v3.xsd

In the schemas zip file in Section 6.2.

### Invoice

Example:

<Invoice xmlns=”http://api.redbox.com/v5/CustomerAccount”>

  <TransactionInfo>

    <CustomerNumber>CustomerNumber1</CustomerNumber>

    <TransactionNumber>TransactionNumber1</TransactionNumber>

    <TransactionStatus>TransactionStatus1</TransactionStatus>

    <CardHolderName>CardHolderName1</CardHolderName>

    <StoreId>StoreId1</StoreId>

    <StoreLocation>StoreLocation1</StoreLocation>

    <StoreZip>StoreZip1</StoreZip>

    <CardLastFour>CardLastFour1</CardLastFour>

  </TransactionInfo>

  <Items oolean=”1.00” tax=”1.00” amountOwed=”2.00”>

    <InvoiceLineItem sequenceNo=”1”>

      <Date>1900-01-01</Date>

      <ItemType>ItemType1</ItemType>

      <Description>Description1</Description>

      <Qty>1</Qty>

      <Amount>1.00</Amount>

    </InvoiceLineItem>

    <InvoiceLineItem sequenceNo=”-2147483647”>

      <Date>0001-01-01</Date>

      <ItemType>ItemType2</ItemType>

      <Description>Description2</Description>

      <Qty>1</Qty>

      <Amount>0.00</Amount>

    </InvoiceLineItem>

  </Items>

  <Payments totalCharges=”2.00” totalPaid=”2.00” balanceDue=”0.00” >

    <InvoicePayment sequenceNo=”1”>

      <CreditCardDate>1900-01-01</CreditCardDate>

      <InvoiceNumber>InvoiceNumber1</InvoiceNumber>

      <Description>Description1</Description>

      <AmountDue>2.00</AmountDue>

      <AmountCharged>1.00</AmountCharged>

      <BalanceDue>1.00</BalanceDue>

    </InvoicePayment>

    <InvoicePayment sequenceNo=”-2147483647”>

      <CreditCardDate>0001-01-01</CreditCardDate>

      <InvoiceNumber>InvoiceNumber2</InvoiceNumber>

      <Description>Description2</Description>

      <AmountDue>1.00</AmountDue>

      <AmountCharged>1.00</AmountCharged>

      <BalanceDue>0.00</BalanceDue>

    </InvoicePayment>

  </Payments>

</Invoice>

Schema:

<xs:schema id=”CA\_Invoice\_V5”

    targetNamespace=”http://api.redbox.com/v5/CustomerAccount”

    elementFormDefault=”qualified”

    xmlns=”http://api.redbox.com/v5/CustomerAccount”

    xmlns:tns=”http://api.redbox.com/v5/CustomerAccount”

    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”  schemaLocation=”..\..\OpenAPI-Paging.xsd” />

  <xs:complexType name=”CA\_Invoice” >

    <xs:sequence >

      <xs:element name=”TransactionInfo” type=”tns:InvoiceTransactionInfo” />

      <xs:element name=”Items” type=”tns:ArrayOfInvoiceItems” />

      <xs:element name=”Payments” type=”tns:ArrayOfInvoicePayments” />

    </xs:sequence>

   </xs:complexType>

  <xs:complexType name=”InvoiceTransactionInfo” >

    <xs:sequence>

      <xs:element name=”CustomerNumber” type=”xs:string” />

      <xs:element name=”TransactionNumber” type=”xs:string” />

      <xs:element name=”TransactionStatus” type=”xs:string” />

      <xs:element name=”CardHolderName” type=”xs:string” />

      <xs:element name=”StoreId” type=”xs:string” />

      <xs:element name=”StoreLocation” type=”xs:string” />

      <xs:element name=”StoreZip” type=”xs:string” />

      <xs:element name=”CardLastFour” type=”xs:string” minOccurs=”0” />  <!—CardLastFour is omitted when the Last4 value is unknown 🡪

    </xs:sequence>

  </xs:complexType>

  <xs:complexType name=”ArrayOfInvoiceItems” >

    <xs:sequence>

      <xs:element name=”InvoiceLineItem” minOccurs=”1” maxOccurs=”unbounded” type=”tns:InvoiceLineItemType” />

    </xs:sequence>

    <xs:attribute name=”oolean” type=”xs:decimal” use=”required” />

    <xs:attribute name=”tax” type=”xs:decimal” use=”optional” />

    <xs:attribute name=”amountOwed” type=”xs:decimal” use=”required” />

  </xs:complexType>

  <xs:complexType name=”ArrayOfInvoicePayments” >

    <xs:sequence>

      <xs:element name=”InvoicePayment” minOccurs=”0” maxOccurs=”unbounded” type=”tns:InvoicePaymentType” />

    </xs:sequence>

    <xs:attribute name=”totalCharges” type=”xs:decimal” use=”optional” />

    <xs:attribute name=”totalPaid” type=”xs:decimal” use=”optional” />

    <xs:attribute name=”balanceDue” type=”xs:decimal” use=”optional” />

  </xs:complexType>

  <xs:complexType name=”InvoiceLineItemType” >

    <xs:sequence>

      <xs:element name=”Date” type=”xs:date” />

      <xs:element name=”ItemType” type=”xs:string” />

      <xs:element name=”Description” type=”xs:string” />

      <xs:element name=”Qty” type=”xs:int” />

      <xs:element name=”Amount” type=”xs:decimal” />

      <!--<xs:element name=”Currency” type=”xs:string” />🡪

    </xs:sequence>

    <xs:attribute name=”sequenceNo” type=”xs:int” />

  </xs:complexType>

  <xs:complexType name=”InvoicePaymentType” >

    <xs:sequence>

      <xs:element name=”CreditCardDate” type=”xs:date” />

      <xs:element name=”InvoiceNumber” type=”xs:string” />

      <xs:element name=”Description” type=”xs:string” />

      <xs:element name=”AmountDue” type=”xs:decimal” />

      <xs:element name=”AmountCharged” type=”xs:decimal” />

      <xs:element name=”BalanceDue” type=”xs:decimal” />

    </xs:sequence>

    <xs:attribute name=”sequenceNo” type=”xs:int” />

  </xs:complexType>

  <xs:element name=”Invoice” type=”tns:CA\_Invoice” />

</xs:schema>

### Partner Preferences

Example:

<DigitalPreferences lastUpdated=”1900-01-01T01:01:01-06:00” xmlns=”http://api.redbox.com/v5/CustomerAccount”>

  <Preferences lastUpdated=”1900-01-01T01:01:01-06:00”>

    <NotificationPreferences>

      <OptInSetting setting=”setting1” value=”true” />

      <OptInSetting setting=”setting2” value=”false” />

      <OptInSetting setting=”setting3” value=”true” />

    </NotificationPreferences>

    <FavoriteStores>

      <StoreId>StoreId1</StoreId>

      <StoreId>StoreId2</StoreId>

      <StoreId>StoreId3</StoreId>

    </FavoriteStores>

    <RecentStores>

      <StoreId>StoreId1</StoreId>

      <StoreId>StoreId2</StoreId>

      <StoreId>StoreId3</StoreId>

    </RecentStores>

    <GamePreferences>

      <GenrePreferences>

        <Genre>Genre1</Genre>

        <Genre>Genre2</Genre>

        <Genre>Genre3</Genre>

      </GenrePreferences>

      <FormatPreferences>

        <Format>Format1</Format>

        <Format>Format2</Format>

        <Format>Format3</Format>

      </FormatPreferences>

    </GamePreferences>

    <MoviePreferences>

      <GenrePreferences>

        <Genre>Genre4</Genre>

        <Genre>Genre5</Genre>

        <Genre>Genre6</Genre>

      </GenrePreferences>

      <FormatPreferences>

        <Format>Format4</Format>

        <Format>Format5</Format>

        <Format>Format6</Format>

      </FormatPreferences>

    </MoviePreferences>

    <InterestedComingSoonTitles>

      <ProductId>ProductId1</ProductId>

      <ProductId>ProductId2</ProductId>

      <ProductId>ProductId3</ProductId>

    </InterestedComingSoonTitles>

    <Extensions />

  </Preferences>

  <NotificationPreferences>

    <OptInSetting setting=”setting4” value=”false” />

    <OptInSetting setting=”setting5” value=”true” />

    <OptInSetting setting=”setting6” value=”false” />

  </NotificationPreferences>

  <ViewingPreferences showClosedCaptions=”true”>

    <PreferredSubtitleLanguages>

      <SubtitleLanguage>SubtitleLanguage1</SubtitleLanguage>

      <SubtitleLanguage>SubtitleLanguage2</SubtitleLanguage>

      <SubtitleLanguage>SubtitleLanguage3</SubtitleLanguage>

    </PreferredSubtitleLanguages>

  </ViewingPreferences>

  <BillingPreferences>

    <SubscriptionBillingAccountReference>SubscriptionBillingAccountReference1</SubscriptionBillingAccountReference>

  </BillingPreferences>

  <Extensions />

</DigitalPreferences>

Schema:

<xs:schema id=”CA\_Preferences\_Digital\_V5”

    targetNamespace=”http://api.redbox.com/v5/CustomerAccount”

    elementFormDefault=”qualified”

    xmlns=”http://api.redbox.com/v5/CustomerAccount”

    xmlns:tns=”http://api.redbox.com/v5/CustomerAccount”

    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:include id=”CA\_Preferences\_V5” schemaLocation=”CA\_Preferences\_V5.xsd” />

  <xs:include id=”CA\_NotificationPreferencesType\_V5” schemaLocation=”CA\_NotificationPreferencesType\_V5.xsd” />

  <xs:complexType name=”CA\_Preferences\_Digital” >

    <xs:sequence>

      <xs:element ref=”Preferences” />

      <xs:element name=”NotificationPreferences” type=”CA\_NotificationPreferencesType”/>

      <!—

      <xs:element name=”SearchBrowsePreferences” type=”CA\_SearchBrowsePreferencesType”/>

      <xs:element name=”ParentalControlSettings” type=”CA\_ParentalControlSettingsType”/>

      🡪

      <xs:element name=”ViewingPreferences”      type=”CA\_ViewingPreferencesType” />

      <xs:element name=”BillingPreferences”      type=”CA\_BillingPreferencesType” minOccurs=”0” />

      <xs:element name=”Extensions”              type=”oai:ExtensionData” minOccurs=”0” />

    </xs:sequence>

    <xs:attribute name=”lastUpdated” type=”xs:dateTime” />

  </xs:complexType>

  <!--  Commenting out until we get complete requirements on DoNotShow & Parental Controls

  <xs:complexType name=”CA\_SearchBrowsePreferencesType” >

    <xs:choice minOccurs=”0” maxOccurs=”unbounded” >

      <xs:element name=”DoNotShow” type=”CA\_DoNotShowType” />

    </xs:choice>

  </xs:complexType>

🡪

  <!—‘productType’ attribute will indicate which type of product (Movie or TVShow) that the list of genres applies to 🡪

  <!—

  <xs:complexType name=”CA\_DoNotShowType” >

    <xs:complexContent >

      <xs:extension base=”ArrayOfGenres” >

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

      </xs:extension>

    </xs:complexContent>

  </xs:complexType>

  <xs:complexType name=”CA\_ParentalControlSettingsType” >

    <xs:sequence  minOccurs=”1” maxOccurs=”unbounded”>

      <xs:element name=”PCSetting”>

        <xs:complexType>

          <xs:attribute name=”deviceID” type=”xs:string” />

          <xs:attribute name=”setting” type=”xs:string” />

        </xs:complexType>

      </xs:element>

    </xs:sequence>

  </xs:complexType>

🡪

  <xs:complexType name=”CA\_ViewingPreferencesType” >

    <xs:sequence>

      <xs:element name=”PreferredSubtitleLanguages”>

        <xs:complexType>

          <xs:sequence>

            <xs:element name=”SubtitleLanguage”  minOccurs=”0” maxOccurs=”unbounded” type=”xs:string” />

          </xs:sequence>

        </xs:complexType>

      </xs:element>

    </xs:sequence>

    <xs:attribute name=”showClosedCaptions” type=”xs:oolean” />

    <!—

    <xs:attribute name=”preferredVideoFormat” type=”xs:string” />

    <xs:attribute name=”preferredBandwith” type=”xs:string” />

    🡪

  </xs:complexType>

  <xs:complexType name=”CA\_BillingPreferencesType” >

    <xs:sequence>

      <xs:element name=”SubscriptionBillingAccountReference” type=”xs:string” />

    </xs:sequence>

  </xs:complexType>

  <xs:element name=”DigitalPreferences” type=”tns:CA\_Preferences\_Digital” />

</xs:schema>

### Partner Profile

Example:

<DigitalCustomerPartnerProfile partnerId=”partnerId1” partnerCustomerNumber=”partnerCustomerNumber1” partnerRelationshipActive=”true” xmlns=”http://api.redbox.com/v5/CustomerAccount”>

  <CustomerAccountProfile>

    <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>

    <CustomerNumber>CustomerNumber1</CustomerNumber>

    <DisplayName>DisplayName1</DisplayName>

    <EmailAddress>EmailAddress1</EmailAddress>

    <FirstName>FirstName1</FirstName>

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

    <LastUpdatedTime>1900-01-01T01:01:01-06:00</LastUpdatedTime>

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

    <LastName>LastName1</LastName>

    <MobilePhoneNumber>MobilePhoneNumber1</MobilePhoneNumber>

    <StateCd>StateCd1</StateCd>

    <StreetAddressLine1>StreetAddressLine11</StreetAddressLine1>

    <StreetAddressLine2>StreetAddressLine21</StreetAddressLine2>

    <ZipPostalCode>ZipPostalCode1</ZipPostalCode>

    <ReviewTOSAccepted>true</ReviewTOSAccepted>

    <Extensions />

    <PasswordChangeDate>1900-01-01T01:01:01-06:00</PasswordChangeDate>

  </CustomerAccountProfile>

  <PartnerRelationshipStartDate>1900-01-01</PartnerRelationshipStartDate>

  <PartnerRelationshipEndDate>1900-01-01</PartnerRelationshipEndDate>

  <Extensions />

  <DigitalTOSAcceptanceDate>1900-01-01T01:01:01-06:00</DigitalTOSAcceptanceDate>

  <CreditsSubscriptionProfile>

    <CreditsSubscriptionSkuId>CreditsSubscriptionSkuId1</CreditsSubscriptionSkuId>

    <CreditsSubscriptionSkuName>CreditsSubscriptionSkuName1</CreditsSubscriptionSkuName>

    <EffectiveDate>1900-01-01</EffectiveDate>

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

    <CreditQuantity>0</CreditQuantity>

  </CreditsSubscriptionProfile>

</DigitalCustomerPartnerProfile>

Schema:

<xs:schema id=”CustomerAccountPartnerProfile”

    targetNamespace=”http://api.redbox.com/v5/CustomerAccount”

    elementFormDefault=”qualified”

    xmlns=”http://api.redbox.com/v5/CustomerAccount”

    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”  schemaLocation=”..\..\OpenAPI-Paging.xsd” />

  <xs:include schemaLocation=”CA\_CustomerAccountProfile\_v5.xsd” />

  <xs:complexType name=”CA\_CustomerAccountBasicProfile\_with\_PasswordChangeDate”>

    <xs:complexContent>

      <xs:extension base=”CA\_CustomerAccountBasicProfile”>

        <xs:sequence>

          <xs:element name=”PasswordChangeDate” type=”xs:dateTime” minOccurs=”1” maxOccurs=”1” />

        </xs:sequence>

      </xs:extension>

    </xs:complexContent>

  </xs:complexType>

  <xs:complexType name=”CA\_PartnerProfileBase”>

    <xs:sequence>

      <xs:element name=”CustomerAccountProfile” type=”CA\_CustomerAccountBasicProfile\_with\_PasswordChangeDate” />

      <xs:element name=”PartnerRelationshipStartDate” type=”xs:date”  minOccurs=”0” maxOccurs=”1” />

      <xs:element name=”PartnerRelationshipEndDate” type=”xs:date”  minOccurs=”0” maxOccurs=”1” />

      <xs:element name=”Extensions” type=”oai:ExtensionData” minOccurs=”0” />

    </xs:sequence>

    <xs:attribute name=”partnerId” type=”xs:string” />

    <xs:attribute name=”partnerCustomerNumber” type=”xs:string” />

    <xs:attribute name=”partnerRelationshipActive” type=”xs:oolean” />

  </xs:complexType>

  <xs:complexType name=”CA\_DigitalCustomerPartnerProfile” >

    <xs:complexContent>

      <xs:extension base=”CA\_PartnerProfileBase”>

        <xs:sequence>

          <xs:element name=”DigitalTOSAcceptanceDate” type=”xs:dateTime” nillable=”true” />

          <xs:element name=”CreditsSubscriptionProfile” type=”CA\_CreditsSubscriptionProfileType” maxOccurs=”1” minOccurs=”0”/>

        </xs:sequence>

        <!--<xs:attribute name=”digitalSubscriptionState” type=”xs:string” />🡪

      </xs:extension>

    </xs:complexContent>

  </xs:complexType>

  <xs:complexType name=”CA\_CreditsSubscriptionProfileType”   >

    <xs:sequence>

      <xs:element name=”CreditsSubscriptionSkuId” type=”xs:string”/>

      <xs:element name=”CreditsSubscriptionSkuName” type=”xs:string” />

      <xs:element name=”EffectiveDate” type=”xs:date” />

      <xs:element name=”ExpirationDate” type=”xs:date” nillable=”true” />

      <xs:element name=”CreditQuantity” type=”xs:unsignedInt” />

    </xs:sequence>

  </xs:complexType>

  <xs:element name=”DigitalCustomerPartnerProfile” type=”CA\_DigitalCustomerPartnerProfile” />

</xs:schema>

### AssignPartnerRelationshipRequest

Example:

<AssignPartnerRelationshipRequest xmlns=”http://api.redbox.com/v5/CustomerAccount”>

  <Extensions />

</AssignPartnerRelationshipRequest>

Schema:

<xs:schema id=”CA\_AssignPartnerRelationshipRequest\_V5”

      targetNamespace=”http://api.redbox.com/v5/CustomerAccount”

      elementFormDefault=”qualified”

      xmlns=”http://api.redbox.com/v5/CustomerAccount”

      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\_AssignPartnerRelationshipRequest” >

    <xs:sequence >

      <xs:element name=”Extensions”  minOccurs=”0” type=”oai:ExtensionData” />

    </xs:sequence>

  </xs:complexType>

  <xs:element name=”AssignPartnerRelationshipRequest” type=”CA\_AssignPartnerRelationshipRequest” />

</xs:schema>

### AssignPartnerRelationshipResponse

Example:

<AssignPartnerRelationshipResponse xmlns=”http://api.redbox.com/v5/CustomerAccount”>

  <PartnerCustomerNumber>PartnerCustomerNumber1</PartnerCustomerNumber>

  <LastUpdatedTime>1900-01-01T01:01:01-06:00</LastUpdatedTime>

  <Extensions />

</AssignPartnerRelationshipResponse>

Schema:

<xs:schema id=”CA\_AssignPartnerRelationshipResponse”

      targetNamespace=”http://api.redbox.com/v5/CustomerAccount”

      elementFormDefault=”qualified”

      xmlns=”http://api.redbox.com/v5/CustomerAccount”

      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\_AssignPartnerRelationshipResponse” >

    <xs:sequence>

      <xs:element name=”PartnerCustomerNumber” type=”xs:string” />

       <xs:element name=”LastUpdatedTime” type=”xs:dateTime”/>

      <xs:element name=”Extensions” type=”oai:ExtensionData” />

    </xs:sequence>

  </xs:complexType>

  <xs:element name=”AssignPartnerRelationshipResponse” type=”CA\_AssignPartnerRelationshipResponse” />

</xs:schema>

### LoginChange

Example:

<LoginChange xmlns=”http://api.redbox.com/v5/CustomerAccount”>

  <LoginEmailAddress>LoginEmailAddress1</LoginEmailAddress>

  <PasswordConfirmationToken>PasswordConfirmationToken1</PasswordConfirmationToken>

  <Extensions />

</LoginChange>

Schema:

<xs:schema id=”CA\_LoginChange\_V5”

      targetNamespace=”http://api.redbox.com/v5/CustomerAccount”

      elementFormDefault=”qualified”

      xmlns=”http://api.redbox.com/v5/CustomerAccount”

      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”  schemaLocation=”..\..\OpenAPI-Paging.xsd” />

  <xs:include schemaLocation=”CA\_PasswordConfirmationResponse\_V5.xsd” />

  <xs:complexType name=”CA\_LoginChange” >

    <xs:sequence>

      <xs:element name=”LoginEmailAddress” type=”xs:string” minOccurs=”1” maxOccurs=”1” />

      <!--<xs:element name=”Password” type=”xs:string” minOccurs=”1” maxOccurs=”1” />🡪

      <xs:element name=”PasswordConfirmationToken” type=”CA\_PasswordConfirmationResponseTokenType” minOccurs=”1” maxOccurs=”1” />

      <xs:element name=”Extensions” type=”oai:ExtensionData”  minOccurs=”0” maxOccurs=”1”/>

    </xs:sequence>

  </xs:complexType>

  <xs:element name=”LoginChange” type=”CA\_LoginChange” />

</xs:schema>

### PasswordChange

Example:

<PasswordChange xmlns=”http://api.redbox.com/v5/CustomerAccount”>

  <NewPassword>NewPassword1</NewPassword>

  <PasswordConfirmationToken>PasswordConfirmationToken1</PasswordConfirmationToken>

  <Extensions />

</PasswordChange>

Schema:

<xs:schema id=”CA\_PasswordChange\_V5”

      targetNamespace=”http://api.redbox.com/v5/CustomerAccount”

      elementFormDefault=”qualified”

      xmlns=”http://api.redbox.com/v5/CustomerAccount”

      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”  schemaLocation=”..\..\OpenAPI-Paging.xsd” />

  <xs:include schemaLocation=”CA\_PasswordConfirmationResponse\_V5.xsd” />

  <xs:complexType name=”CA\_PasswordChange” >

    <xs:sequence>

      <xs:element name=”NewPassword” type=”xs:string” minOccurs=”1” maxOccurs=”1” />

      <!--<xs:element name=”OldPassword” type=”xs:string” minOccurs=”1” maxOccurs=”1” />🡪

      <xs:element name=”PasswordConfirmationToken” type=”CA\_PasswordConfirmationResponseTokenType” minOccurs=”1” maxOccurs=”1” />

      <xs:element name=”Extensions” type=”oai:ExtensionData”  minOccurs=”0” maxOccurs=”1”/>

    </xs:sequence>

  </xs:complexType>

  <xs:element name=”PasswordChange” type=”CA\_PasswordChange” />

</xs:schema>

### Password Confirmation Request

Example:

<PasswordConfirmationRequest xmlns=”http://api.redbox.com/v5/CustomerAccount”>

  <Password>Password1</Password>

  <Extensions />

</PasswordConfirmationRequest>

Schema:

<xs:schema id=”CA\_PasswordConfirmationRequest\_V5”

      targetNamespace=”http://api.redbox.com/v5/CustomerAccount”

      elementFormDefault=”qualified”

      xmlns=”http://api.redbox.com/v5/CustomerAccount”

      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\_PasswordConfirmationRequest” >

    <xs:sequence>

      <xs:element name=”Password” type=”xs:string” minOccurs=”1” maxOccurs=”1” />

      <xs:element name=”Extensions” type=”oai:ExtensionData”  minOccurs=”0” maxOccurs=”1”/>

    </xs:sequence>

  </xs:complexType>

  <xs:element name=”PasswordConfirmationRequest” type=”CA\_PasswordConfirmationRequest” />

</xs:schema>

### Notification Preferences

This defines a shared schema used by both the Preferences and PartnerPreferences schemas to define a type that holds Opt-in settings.

Schema:

<xs:schema id=”CA\_NotificationPreferencesType\_V5”

    targetNamespace=”http://api.redbox.com/v5/CustomerAccount”

    elementFormDefault=”qualified”

    xmlns=”http://api.redbox.com/v5/CustomerAccount”

    xmlns:tns=”http://api.redbox.com/v5/CustomerAccount”

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

    xmlns:oai=”http://api.redbox.com/OpenAPI/v1”

>

  <xs:complexType name=”CA\_NotificationPreferencesType” >

    <xs:sequence >

      <xs:element name=”OptInSetting” type=”CA\_OptInSettingType”  minOccurs=”0” maxOccurs=”unbounded” />

    </xs:sequence>

  </xs:complexType>

  <!--  OptIn Settings:

        valid settings are:

          NewlettersOptIN

          MarketingEmailsOptIN

          ProductEmailsOptIN

          VOD/ESTPromo

🡪

  <xs:complexType name=”CA\_OptInSettingType” >

    <xs:attribute name=”setting” type=”xs:string” />

    <xs:attribute name=”value” type=”xs:oolean” />

  </xs:complexType>

</xs:schema>

### PasswordResetRequest

Example:

<PasswordResetRequest xmlns=”[http://api.redbox.com/v5/CustomerAccount](http://api.redbox.com/v4/CustomerAccount)”>

  <ContactEmailAddress>[iveforgotten@myemail.com</ContactEmailAddress](mailto:iveforgotten@myemail.com%3c/ContactEmailAddress)>

  <Extensions />

</PasswordResetRequest>

Schema:

<xs:schema id=”CA\_PasswordResetRequest\_V5”

      targetNamespace=”[http://api.redbox.com/v5/CustomerAccount](http://api.redbox.com/v4/CustomerAccount)”

      elementFormDefault=”qualified”

      xmlns=”[http://api.redbox.com/v5/CustomerAccount](http://api.redbox.com/v4/CustomerAccount)”

      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\_PasswordResetRequest” >

    <xs:sequence>

      <xs:element name=”ContactEmailAddress” type=”xs:string” minOccurs=”1” maxOccurs=”1” />

      <xs:element name=”Extensions” type=”oai:ExtensionData”  minOccurs=”0” maxOccurs=”1”/>

    </xs:sequence>

  </xs:complexType>

  <xs:element name=”PasswordResetRequest” type=”CA\_PasswordResetRequest” />

</xs:schema>

### PasswordConfirmationResponseToken

Output of CP29 and used as part of the input to CP25 and CP27.

Example:

<PasswordConfirmationToken xmlns=”http://api.redbox.com/v5/CustomerAccount”>PasswordConfirmationToken1</PasswordConfirmationToken>

Schema:

<xs:schema id=”CA\_PasswordConfirmationResponse\_V5”

      targetNamespace=”http://api.redbox.com/v5/CustomerAccount”

      elementFormDefault=”qualified”

      xmlns=”http://api.redbox.com/v5/CustomerAccount”

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

      xmlns:oai=”http://api.redbox.com/OpenAPI/v1”

>

  <xs:simpleType name=”CA\_PasswordConfirmationResponseTokenType” >

      <xs:restriction base=”xs:string” />

 </xs:simpleType>

  <xs:element name=”PasswordConfirmationToken” type=”CA\_PasswordConfirmationResponseTokenType” />

</xs:schema>

### CreditCardVerificationRequest

Example:

<CreditCardVerificationRequest xmlns=”http://api.redbox.com/v5/CustomerAccount”>

  <CVV>CVV1</CVV>

</CreditCardVerificationRequest>

Schema:

<xs:schema id=”CA\_CreditCardVerificationRequest\_v5”

    targetNamespace=”http://api.redbox.com/v5/CustomerAccount”

    elementFormDefault=”qualified”

    xmlns=”http://api.redbox.com/v5/CustomerAccount”

    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” schemaLocation=”..\..\OpenAPI-Paging.xsd” />

  <xs:complexType name=”CA\_CreditCardVerificationRequest” mixed=”false” >

    <xs:sequence >

      <xs:element name=”CVV” type=”xs:string” minOccurs=”0” />

      <xs:element name=”Extensions” type=”oai:ExtensionData” minOccurs=”0”/>

    </xs:sequence>

  </xs:complexType>

  <xs:element name=”CreditCardVerificationRequest” type=”CA\_CreditCardVerificationRequest” />

</xs:schema>

## Schema Files

The xsd schema definition files for the OpenAPI are attached below. If there is any discrepancy between the contents of the schema files and the samples provided in Section 6.1, the schema files should be considered authoritative.



JSON schemas:



# 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 I 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. |
| 5/9/11 | 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. |
| 5/27/11 | 3.0 Draft 1 | Chris Rudolphi | Modified P4 and P6 to make them HTTP GET operations instead of POST operations. Moved their inputs from XML content to be query string parameters. Deprecated schemas 6.1.4 and 6.1.5 as they are no longer needed.  Modified all Apigee URL paths to use lowercase.  Modified the schemas for ItemChangedEventMessage and ItemChangesMessage to make the source attribute as optional instead of required.  Corrected P6 (Search) description to correctly define defaults for sort field and sort order.  Changed the URL of /createcustomer to /customers at both Apigee and gateway layers.  Modified URL structure of P9 (Similar Titles) to /products/movies/{pid}/similar  Modified URL structure of I2: /storesinventory to /inventory  Modified URL structure of Cart to remove {UserID} – not needed  Modified URL structure of creating a reservation to /reservations  Added Games to Products; modified schemas for Source & External Product and PaginatedProductList.  Revised wording of Inventory & Reservations to remove “Movie” specific terminology (no change to behavior)  **Modified schema names to reflect revised schema naming convention (v3 at the start of the schema name rather than at the end) for External Schemas affected by R3**  Changed the operation signature of CP12: GetRecommendations to accept one or more product format values as input.  Changed name of MovieTrailers schema to Trailers (to support games).  Corrected default sort field and sort order for Store and Inventory searches. |
|  | 3.0 Draft 2 |  | Fixed Source Top20 and Inventory schemas as some attributes were incorrectly specified as ‘string’ instead of ‘xs:string’.  Added schemas for CreateCustomer Request and Response.  Modified CreditCardInfo schema to split Name into FirstName and LastName.  Eliminated ProductList and PaginatedProductList schemas in favor of a single combined Products schema. |
| 6/14/11 | 3.0 Draft 3 |  | Added <UppercaseTitle> element to Products schema to support easier product searching on Apigee side.  Deprecated Source Products schema (6.1.2).  Modified schemas for External Item Changes messages (Store and Inventory) by adding Epoch-style update timestamps (to make date comparisons easier at Apigee).  Added detail on which of the built-in service protection policies will be used at Apigee (section 2.7.15) |
|  | 3.0 Draft 4 |  | Included correction of R2-cr1 to correct the valid values of MPAARatings.  Added schemas for CP19 operation (6.1.47 and 6.1.48)  Modified Source Inventory schema to include Epoch style update timestamps. |
| 6/16/11 | 3.0 Draft 5 |  | Eliminated use of Inventory schema 6.1.16 that combined Store and Inventory metadata into a single response. All Inventory query operations return 6.1.14 – which is a simple list of inventory by Store.  Added prefix of ‘/v3’ to all Inventory Apigee URLs and OpenServices Inventory Gateway URLs.  Added description of new, granular caching of Products on Apigee side (caches PC1 through PC12).  Added V3 operations for Products and Products cache management.  Added new v3 schema for Products (6.1.49).  Added new v3 schema for Top20 (6.1.50).  Added new schemas for CustomerCreation Request and Response (6.1.47 and 48).  Eliminated Products operation descriptions for operations that were not actually delivered in R2.  Updated Reservation Gateway description to include v3 operations for validating Cart contents. Apigee will pass through calls to these operations for validation of the Cart.  Added ‘/v3’ as a prefix for ALL r3 related URLs (Inventory, Reservations & Customer). Also added to the r3 Products URLs.  Modified description of Products Cache Refresh Message Handler (2.7.13.2) to add description that the Refresh service will directly invoke the Apigee Products cache refresh operations.  Updated Products service description diagram to depict the Cache Refresh Service directly invoking the Apigee layer to cause Products cache refreshes.  Modified the AccountInfo schema (6.1.30) to add FirstName and LastName fields. |
| 6/20/2011 | 3.0 Draft 6 |  | Modified Reservations operations to consolidate R1, R2, and R3 into a single ‘validation’ operation. Changed both Apigee Reservation endpoint description and Reservation Gateway descriptions. |
| 6/21/11 | 3.0 Draft 7 |  | Modified (internal) OpenServices gateway URL structures to make the naming consistent across gateway service operations. |
| 6/22/11 | 3.0 Draft 8 |  | Updated Inventory cache update handling to match Apigee capabilities:   * Full refresh (I4) schema is now the External InventoryLookup schema (6.1.14) * Source inventory list schema is deprecated (6.1.12) * Inventory change message to Apigee is deprecated (6.1.45) * Operation I3 at Apigee is deprecated * Operation /Inventory/InventoryChanged at Inventory Gateway is deprecated |
| 6/23/11 | 3.0 Draft 9 |  | Modified Customer Preferences schema to match changes made to CP system. |
| 6/27/11 | 3.0 Draft 10 |  | Reverted changes to Source Products Schema (6.1.2) to the version that was published in Release 2. (None of the Products changes described for R3 were actually made for Release 3). |
| 6/30/11 | 3.0 Draft 11 |  | Modified BasicProfile Schema (6.1.23) to rename ContactEmailAddress as EmailAddress and eliminated NotificationEmailAddress.  Modified AccountInfo schema (6.130) by breaking ExpirationDate into two distinct fields: ExpirationMonth and ExpirationYear, both xs:int datatype.  Modified CreditCardInfo schema (6.1.34) by eliminating the use of “Type” and “CVV” attributes and by changing the Month and Year elements from “string” to “int” datatypes. |
| 7/1/11 | 3.0 Draft 12 |  | Corrected schema 6.1.30 AccountInfo by changing ExpirationMonth and ExpirationYear into xs:string datatype.  Corrected schema 6.1.34, CreditCardInfo, by changing Month and Year into xs:string datatype..  Modified schema 6.1.23, BasicProfile, to make the DefaultAccountNumber element optional by setting its minOccurs=0. |
| 7/5/11 | 3.0 Draft 13 |  | Modified Inventory design to reflect Single Store inventory approach:   * Modified description of Inventory service and diagram in Section 2.4 * Modified Inventory Gateway design (section 2.7.4) by adding for the capability of Apigee to call the gateway to request a single store’s inventory. Also changed description of how full refresh should work. Added dependency on AppFabric to cache inventory data. * The output of the Inventory Gateway is now the external schema (6.1.14). The source inventory schema (0) is deprecated. * Modified description of Inventory Endpoint (section 2.7.15.1.3), by eliminating operation I2, modifying I1 to retrieve a single store’s inventory, and modifying I4 to allow us to POST a single store’s inventory to refresh the Apigee cache. * Marked sections 2.7.11 and 2.7.14 as OUT OF SCOPE for R2 and R3 as we no longer will be using the item-level change notification mechanisms of the original design. * Marked schemas 6.1.40 thru 6.1.45 as OUT OF SCOPE for R2 and R3 R3 as we no longer will be using the item-level change notification mechanisms of the original design. * Added lastUpdate timestamps back into schema 6.1.14 (the primary Inventory schema). |
| 7/6/11 | 3.0 Draft 14 |  | Modified Inventory design so that cache refreshes are pulled by the Apigee gateway from the I Inventory Gateway. Operation I4 was changed from a POST to a GET.  Modified RentalHistory schema (6.1.25) to make the Details element optional. *This is to support a phased implementation of Rental History and the first phase will not include the Details element*. |
| 7/8/11 | 3.0 Draft 15 |  | Reverted Products, Stores and Top20 schemas to R2 versions so that the design aligns with the project decision to not make any changes in these areas for R3. Affected schemas include:   * 6.1.6 Paginated Product List * 6.1.9 Source Stores Schema * 6.1.10 External Stores Schema * 6.1.11 Common External Store Schema * 6.1.13 External Store Lookup Response Schema * 6.1.15 Source Top20 Schema   The 6.1.15 schema was modified to correct the data-type for the productId attribute. |
| 7/14/11 | 3.0 Draft 16 |  | Corrected schema 6.1.33 Credit Use History to allow for null RedeemProductId.  Modified the structure of validation error reporting for PricedCartResponse and ReservationResponse schemas (6.1.21 and 6.1.22).  Modified Cart, CartValidation, PricedCart, PricedCartResponse and Reservation Response schemas, and added ProductRatings schema, to refactor how Credit use is signaled in the Cart. Eliminated the concept of AppliedDiscounts.  Corrected namespace of CreditCardInfo (6.1.34)  Modified RentalHistory schema to allow ReturnedDate to be nillable. |
| 7/18/11 | 3.0 Draft 17 |  | Fixed error in enumerated values in the MPAARatings schema |
| 8/3/11 | 3.0 Draft 18 |  | Fixed error in CustomerAccountProfile schema (6.1.23) {fixed missing double-quotes around the 0 on the minOccurs attribute of the DefaultAccountNumber element. |
| 9/30/11 | 6.0 Draft 1 | C Rudolphi | First draft of R6 Products.  Added Products schema element for Flags.  Added Products schema element for ProductMetrics. |
| 10/6/2011 | 6.0 Draft 2 |  | Revised Products schema to include SynopsisLong & Short to Games. Refactored the product schema inheritance hierarchy so that all products have Flags. These two changes resulted in changes to the samples generated for Products & Top20 (6.1.49 and 6.1.50). |
| 10/11/2011 | 6.0 Draft 3 |  | Revised the description of the Operation P3-1 (/products/movies/default) to add support for the ‘includeComingSoon’ flag.  Revised the description of the Operation P3-15 (/products/games/default) to add support for the ‘includeComingSoon’ flag.  Revised the description of the cache contents for caches #2 and 4 on page 7 to clarify that these caches contain ALL products of their type, not just in-circulation products.  Added Operations P3-17 and P3-18 to separate Product Metrics (movies and games) into their own operations.  Added Operations PC13 and PC14 to refresh the product metrics caches.  Added Products caches #13 and 14 at Apigee cache product metric data.  Added operations to the OpenServices Products gateway to provide movie and games metrics to Apigee.  Modified the Products schema (6.1.49) to remove the product metrics from the main products schema.  Revised the Products service diagram in Section 2.1 to reflect that the CRS will call Apigee directly. |
| 10/12/11 | 6.0 Draft 4 |  | Revised the Products service diagram in Section 2.1 to correct that CRS will also call OpenServices (/products/fullrefresh) before calling Apigee.  Modified the CacheDataSubSet column of the table on page 7 that describes each Products cache to clarify which set of data each cache contains. |
| 10/25/11 | 7.0 Draft 1 |  | Revised Products schema to add new elements and attributes to Movie and Games definitions to match the new data available from EPC. (Card 493) |
| 10/27/11 | 7.0 Draft 2 |  | Revised the AccountInfo and CreditCardInfo schemas to include a common BillingAddress. Billing Address is now an optional editable element of each credit card.  Revised the schema version numbers of the above to /v4.  Revised the schema version number of AccountList to /v4.  (Card 435) |
|  |  |  | Added operation CP24 and schema for retrieval of customer Invoice. Schema: 6.1.53.  (Card 426) |
|  |  |  | Added operations CP22 and CP23 for the retrieval and update of customer partner preferences. Schema: 6.1.54.  (Cards 440, 441, 442, 443, 522, and 528) |
|  |  |  | Added operation CP20 to retrieve a customer Partner Profile. In this design, the operation will return a “DigitalPartnerProfile” element, but the operation design is flexible to allow returning an element specific to each partner.  (Card 517) |
|  |  |  | Added operation CP21 to assign a partner customer number.  (Card 516) |
|  |  |  | Added operation CP25 to support updating a user’s login email address and password. Renamed Schema 6.1.47 to CustomerIdentity and it is reused in both CP19 and CP25 (New customer creation and login change).  (Card 525) |
|  |  |  | Added operation CP26 to support reset of a user’s password. This operation will kick-off the existing password change process.  (Card 504) |
|  |  |  | Updated ALL Customers related schema namespaces and names to /v4 |
|  |  |  | Updated most Customer URL paths to accept two forms of identity, either the existing {CustomerNumber} URL segment or the combination of {VendorCode}.{CustomerPartnerNumber}. Either can be used interchangeably to identify a customer.  (Cards: 438, 518, 519, 520, 521) |
|  |  |  | Updated the descriptions of the Reservation service and API to describe that a user can be identified by either his CustomerNumber or by the contatenation of VendorCode and “.” And CustomerPartnerNumber.  Card: 439 |
| 10/31/2011 | 7.0 Draft 3 |  | Revised Operation CP25 to limit the operation to changing the Login information. Created an input schema specifically for this operation: Schema 6.1.58.  (Card 525, W1:PC) |
|  |  |  | New operation CP27: Change Password. New schema: 6.1.59 takes in old and new password.  (Card 525, W1:PC) |
| 11/2/11 | 7.0 Draft 4 |  | Added Operation CP28 (Delete Account). No schema changes required.  (Card 550, W1:PC) |
|  |  |  | Added Operation CP29: Confirm Password. New schema: 6.1.60 PasswordConfirmationRequest.  (Card 549, W1:PD) |
|  |  |  | Modified the URL paths to operations CP25, 27 and 29 to clarify the operations that modify login and passwords.  CP25 PUT to /customers/:id/identity/login                               to update a login email  CP27 PUT to /customers/:id/identity/password                       to update a password  CP29 POST to /customers/:id/identity/password/confirm     to confirm an existing password |
|  |  |  | Added a <Password> element to the LoginChangeRequest schema 6.1.58. Each request to change a login email address is supposed to be confirmed by validating the password of the customer. |
|  |  |  | Added a date for TOS Acceptance Date to the Digital Profile schema 6.1.55  (Card 536 W1:PB) |
|  |  |  | Added <RecentStores> to the Preferences schema: 6.1.24.  (Card 509 W1:PD) |
|  |  |  | Modified Products cache designs to include new caches and operations for Movies Default ComingSoon and Games Default ComingSoon caches. New caches were defined as PC15 and PC16. New Apigee operations P3-15 and P3-16. New Products OpenServices gateway operations products/movies/defaultComingSoon and products/games/defaultComingSoon.  Cards: 552 & 553 |
|  |  |  | Modified the Game and Movie schemas such that only <Movie> has a ‘format’ attribute, while <Game> has both a ‘platform’ and ‘sub-platform’ attributes.  Schema: 6.1.49 |
|  |  |  | Added a ‘locationId’ attribute to the <Location> tag in the <Store> schema. This allows client code to easily group stores that are housed at the same location (when the Address fields may contain subtle differences).  Schemas: 6.1.10, 6.1.11, 6.1.13  Card: 554 |
|  |  |  | Modified the PartnerPreferences schema for DigitalPartnerPreferences to match the known requirements for W1:PD.  Eliminated DoNotShow, video bandwith and format.  Cards 440 and 442.  Eliminated Parent Controls – not in scope. Changed the name of the ‘enables’ attribute on the PCSetting element to be ‘setting’ (for ParentalControl Settings). Eliminated PIN.  Card 441. |
|  |  |  | Corrected all Stores schemas (external) to be v2. This matches what has been implemented. |
| 11/8/11 | Draft 5 |  | Revised Operation Descriptions text for Reservations & Customers services (Apigee) to indicate that the required security evidence is an Oauth access\_token (not an APIKey). |
|  |  |  | Modified CreditUseHistory schema (6.1.33). Added Quantity and TransactionNumber fields. Restructured the schema to improve its structure.  (Card 556, W1:PD) |
|  |  |  | Modified RentalHistory schema (6.1.25) by adding a field for TransactionNumber.  (Card 569, 570, 571 W2PA) |
|  |  |  | Modified CreditCardInfo schema (6.1.34 ) to add a flag that indicates whether the card should be saved to the user’s profile (associate the card account to the Profile).  (Card 572 W2PA) |
|  |  |  | Modified Partner Profile schema (6.1.55 ) to eliminate the Credit Subscription Description element and to correct capitalization errors on the names of the elements.  (Card 562, 574 W1:PB) |
|  |  |  | Added operation CP30: Record Acceptance of Partner Terms & Conditions. Added schema: **Error! Reference source not found.**.  (Card 536 W1PB) |
|  |  |  | Added RentalStatus to RentalHistory schema: 6.1.25  (Card 580, W2PA) |
|  |  |  | Modified Customers URLs for CP20, 21, 22, 23 & 30. These operations all access partner-related information; therefore the URLs are now structured such that /partner/ is a segment in the URL path (such as /customer/:id/partner/profile). |
|  |  |  | P3-6: added ‘game’ as valid value of search format parameter |
|  |  |  | Fixed up descriptions of Products cache refresh operations to reflect how it had actually been implemented. |
|  |  |  | Schema: 6.1.30 AccountInfo; added field CVVChecked (Boolean) to track whether the CVV value for that card had been confirmed by a previous transaction.  (Card: 584, W2PA) |
|  |  |  | Modified the PricedCart schema 6.1.20 (and by extension, the schemas that inherit from it) by changing the <Tax> element to be optional. In some states(eg Maryland) the tax is built into the InitialNight Price and so there is no explicit tax amount.  Also modified the Invoice schema 6.1.53 with the same change (<Tax> element is optional) for the same reason.  (Card 586, W2PA) |
| 11/18/11 | R7 Draft 6 | Chris Rudolphi | Removed <DefaultAccountNumber> element from the CustomerAccountProfile (BasicProfile) schema: 6.1.23. Designation of ‘default’ account is moved to the Account itself.  (Card: 617 W2:PB) |
|  |  |  | Modified PartnerProfile schema 6.1.55 with the following changes:   * Added <CustomerAccountProfile> to the partner profile, which provides a way to fetch a combined basic & partner profile in a single call. * Eliminated the customerNumber attribute (as this data is provided within the <CustomerAccountProfile> * Renamed the ‘active’ attribute to ‘partnerRelationshipActive’ to better distinguish it from the ‘IsActive’ element. * Fixed a capitalization error on the ‘digitalSubscriptionState’ attribute.   (Card: 621) |
|  |  |  | Refactored the concept of Opt-In preferences by refactoring it into its own schema: CA\_NotificationPreferencesType (6.1.61). This affects both Preferences and PartnerPreferences. |
|  |  |  | Removed the PartnerTermsAcceptanceRequest schema as the operation using it has been removed.  Removed operation CP30.  (Card: 624 W2:PB) |
|  |  |  | Modified <Preferences> schema 6.1.24 to use the new NotificationPreferencesType shared schema. |
|  |  |  | Modified Partner Preferences schema 6.1.61 to use the new NotificationPreferencesType shared schema. |
|  |  |  | Added back into the Partner Preferences schema 6.1.61 the <BillingPreferences> element to hold the accountnumber of the card to use for subscription billing.  (Card: 623 W2:PB) |
|  |  |  | Modified Partner Preferences schema 6.1.61 to include the base customer preferences as a child element.  (Card: |
|  |  |  | Modified the AccountInfo schema 6.1.30 to include:   * isPreferredAccount attribute * <AVSChecked> element   (Card: 617, 618. W2:PB) |
|  |  |  | Modified the LoginChangeRequest schema to eliminate the Firstname, LastName, and Zip elements.  (Card: 614 W2:PB) |
|  |  |  | Added avsCheck query string parameter to the CP18 (Add Credit Card) operation.  (Card: 616 W2:PB) |
|  |  |  | Modified CreditUseHistory Schema 6.1.33 to remove the <Quantity> element.  (Card: 625 W2:PB) |
|  |  |  | Added operation CP31: Update Partner Profile  (Card 622 W2:PB) |
| 11/23/11 | R7 Draft 7 |  | CP26: Replaced query string param with a schema (PasswordResetRequest 6.1.62). |
|  |  |  | CP26: Revised URL to /v5/customers/identity/resetpassword. This adds the /identity/ subpath. Also added comments to clarify that the CustomerNumber is not part of the URL path.  (Card 663) |
|  |  |  | CP20 and CP31: Corrected schema 6.1.55 Partner Profile to set the ExpirationDate of the CreditSubscriptionProfile to be nillable.  (Card 621, per email with Andrew B). |
|  |  |  | CP15: Added avsCheck query string parameter to CP15 (Save Account).  (Card 662). |
|  |  |  | Revised all external schemas to include schemaLocation information on schema imports (to assist with schema validation tooling). |
|  |  |  | CP24: Fixed url of CP24 on page 30 (OpenService Customer Gateway) to correctly show that TransactionNumber is part of the /Invoices url. |
|  |  |  | CP1, CP2, CP20, CP31: Added element <ReviewTOSAccepted> to the Basic Profile 6.1.23 to indicate that the customer has accepted the Pluck Terms of Service for entering product reviews. This element also shows up in the Partner Profile 6.1.55 via inheritance.  (Card: 675 W2PB) |
|  |  |  | CP16, CP17: Added CreditType element to the CreditUseHistory schema 6.1.33 and the CreditsAvailable schema 6.1.32.  (Card: 676 W2PB) |
|  |  |  | CP25: Modified CP25 (Login Change). The operation should return only a status code, no need to return a data contract. |
|  |  |  | All: Access to OpenServices operations will be protected by restricting access to a configured AD group. OpenServices will be configured for HTTP BASIC authentication (via IIS configuration) and only callers presenting credentials of service accounts that are members of the configured group will be allowed to invoke the OpenServices operations.  (Card: 677 W2PB) |
|  |  |  | P10: Corrected the URL format for P3-10 (top20) such that the period parameter is called ‘period’ consistently. |
|  |  |  | All: Added section 2.10.3.1 ActivityID Tracing to describe the use of a custom HTTP Header that will be accepted with each request and flowed through to all downstream systems and included in all log and audit entries (at both the Apigee gateway and OpenServices layers) to facilitate log entry correlation.  (Card: 679 W2PB) |
| 12/7/11 | R7 Draft 8 |  | Corrections to the attached schemas to correctly mark each use of <Extensions> as optional (minOccurs=0); provide a schemaLocation attribute for each <import> ed namespace. (And to correct the fact that the incorrect versions of the schema files were attached to the Draft 7 document). |
| 12/9/11 | R7 Draft 9 |  | CP24: Removed Currency from Schema: 6.1.53. |
|  |  |  | CP24: Schema: 6.1.53 modified CardLastFour element to make it minOccurs=0 so that it will be omitted from an Invoice when the value is unknown. |
|  |  |  | Fix to the shared schema: OpenAPI-Paging.xsd to include a schemaLocation that references atom.xsd |
|  |  |  | CP24: Schema: 6.1.53 made the InvoicePayment line items as minOccurs=0. |
|  |  |  | CP22, CP23: Schema:6.1.54 modified the SubtitleLanguage repeating element in PreferredSubtitleLanguages to be minOccurs=0 |
|  |  |  | CP20, CP31: Schema: 6.1.55 removed ‘digitalSubscriptionState’ as an attribute of the Digital Partner Profile. |
|  |  |  | CP1, CP2, CP20, CP31- Schema: 6.1.23 Basic Account Profile – modified the elements CityName, DisplayName, FirstName, LastName, MobilePhoneNumber, StateCd, StreetAddressLine1 and 2 to all be optional (minOccurs=0). |
| 12/13/11 | R7 Draft 10 |  | CP25: Corrected internal description of CP25 to document that input is a CustomerIdentity object. Also corrected external description to match. |
|  |  |  | CP32: Added account verification operation. AVS and CVV check logic should be pulled out of Account creation an dupdate operations into this operation. |
|  |  |  | CP33: New operation to set an Account as the subscription billing card for a given PartnerID. |
|  |  |  | Common Schemas: Added a common Error schema. |
|  |  |  | CP15, CP18, and CP31: removed all query string parameters that indicate whether an AVS or CVV check should be performed. These operations no longer are responsible for such checks, instead the Digital Orchestration Layer will call operation CP32 to perform such checks. |
|  |  |  | CP18: Internal operation (OpenServices) path corrected to match implementation, ie, /Card/{CustomerNumber}/creditcard.  No change required on Apigee. |
|  |  |  | Revised URLs (both internally and externally) to consistently refer to a partner ID with the query string parameter name of ‘partnerId’. This affects OpenServices operations: CP20, CP21, CP22, CP23 and CP31.  This affects Apigee operations CP22 and CP23. |
|  |  |  | Internal Stores Schema: 6.1.9 added locationId attribute to the Location tag. This makes the schema semantically equivalent with the external stores schema. (not reflected in the example, but it is in the schema). |
| 12/16/11 | R7 Draft 11 |  | CP23: modified schema: 6.1.54 Partner Prefrences to make the BillingPreferences element optional. This change is because the caller is not allowed to SET the billing credit card via CP23 (instead should use CP33). |
| 12/19/11 | R7 Draft 12 |  | Modified sample and attached Externa l Store Schemas to match the stub and test environments:   * <StoreBulkList> instead of <StoresBulkList> * <StoreLookupList> instead of <StoresLookupList> * All store namespaces are <http://api.redbox.com/Stores/v2> |
| 12/19/11 | R7 Draft 13 |  | Corrected Store schema 6.1.11 to make the commStatus attribute as optional. |
| 12/22/11 | R7 Draft 14 |  | Updated Section 2.7.15.2 to include detail on which Apigee policies are applied to each operation. |
|  |  |  | S1, S2, S3, S4: Modified the External Store schema: 6.1.11 to make the StoreType (Indoor/Outdoor) as optional. |
| 1/9/12 | R7 Draft 15 |  | Modiified Apigee URL path for CP25 to:  /identity  instead of  /identity/login  In order to make it consistent with the OpenServices URL. |
|  |  |  | Corrected description of CP25 to indicate that the input to CP25 is schema 6.1.58, LoginChangeRequest. |
|  |  |  | Modified LoginChangeRequest, 6.1.58, schema by changing it’s element name to LoginChange, changed the ContactEmailAddress element to LoginEmailAddress, and removed the Password element. Also changed the schema name to remove the word ‘request’ from the schema name. It is now called CA\_LoginChange\_V4. |
|  |  |  | Modified PasswordChangeRequest, schema 6.1.59 by changing it’s name to PasswordChange. Eliminated the Password element from the schema. |
|  |  |  | Modified CP27 (Password Change) to reflect the name change of the schema it uses (6.1.59). |
| 1/11/2012 | R7 Draft 16 |  | CP26: Fixed URL structure description on page 56 to match that described on page 67. The URL does not include the email address. |
| 1/27/2012 | R7 Draft 17 |  | R4, R5: Fixed PricedCart use of ProductRatings schema.  Added a CombinedRatings type that combines both MPAA and ESRB ratings so that it can be used in the cart. |
| 1/20/12 | R8 Draft 1 |  | CP26: Fixed Internal OpenServices URL to match the external URL. |
|  |  |  | CP27: Fixed Internal OpenServices URL to match the external URL. |
| 1/23/12 | R8 Draft 2 |  | I4: Fixed missing forward slash ‘/’ in URL for operation I4 in the table on page 51. (Thanks Eric!) |
|  |  |  | S6: Added operation to find stores nearest a ZipCode. This required updates to sections 2.2, 2.7.2 (OpenServices Stores gateway) and 2.7.15.2 (Apigee stores endpoint) and to section 2.7.9.1.1 (ZipCode Location Table).  No contract changes. |
|  |  |  | I5, I6: Added two Inventory search operations that search across multiple stores given a Lat/Long (I5) and a PostalCode (I6). No contract changes. |
|  |  |  | CP29, CP25 & CP27: per Card: #927  Modified CP29 to return a Password Confirmation Response Token.  Created schema 6.1.63 for this data type.  Modified LoginChange, 6.1.58, and PasswordChange, 6.1.59, input schemas to require the Token as input. |
|  |  |  | P3-1, P3-2, P3-6, P3-15, P3-16, P3-17, P3-18: Added Paging support to these Products operations. (Card: 945) |
|  |  |  | P3-6 (Search). Added parameters for Search Operation, and Search Field. Renamed the search Text field to ‘q’.  Modified the ProductsList schema (6.1.49) to include attributes which echo back the input search parameters.  Card: 950 |
|  |  |  | P3-4 (/products?productList={}) is added back in scope  Card: |
|  |  |  | S2: the storeList input parameter is now optional. This provides the consumer with the ability to retrieve a paginated view of all stores.  Card: 945 |
|  |  |  | CP24: Invoice. Modified the schema (6.1.53) to make the SubTotal, Tax, and Amount Owed fields as attributes of the Invoice Items collection and added TotalCharges, TotalPaid, and BalanceDue as attributes of the Payments collection.  Card: 930. |
|  |  |  | P3-4: /products?productIds={list}; Modified the description of the operation in 2.7.15.1.1 (Apigee operation description) to note that if the List of Product IDs is missing or empty that the operation should return ALL products (subject to the applicable pagination parameters). |
| 2/3/12 | Ported from R7 Draft 18 |  | CP20, CP31: Modified the PartnerProfile schema (6.1.55) by renaming the DigitalSubscriptionStartDate and DigitalSubscriptionEndDate to be PartnerRelationshipStartDate and PartnerRelationshipEndDate, respectively. This fixes a mistake in how these fields are named and used. The fields are also made optional, as these fields should not be updatable by the affiliate.  Defect Card: 962 |
| 2/7/12 | R8 Draft 3 |  | R4, R5. Enhanced the Cart Schema (6.1.18 ) to include the <ClientIPAddr> element. Updated examples of related schemas (PricedCart, PricedCartResponse, and ReservationResponse) to also depict the use of the ClientIPAddr element.    Card: 972 |
| 2/14/12 | R9 Draft 1 |  | The input to R5 is now the Cart object (schema 6.1.18), making it the same as the input for R4. The purpose of this change is to eliminate the requirement that the caller echo back into R5 all of the pricing information that is the result of calling R4.  **As a result of this change, the Version # of the Reservations operations and schemas has changed to V4**. This affects ALL reservation operations (R0-R5) and schemas 6.1.18, .19, .21, and .22 (Cart, CartValidation, PricedCartResponse, ReservationResponse).  With this change, the PricedCart schema (6.1.20) is deprecated.  Card: 997 |
|  |  |  | Revised the font coloring of P3-4 (Products by productIDlist) to indicate that it is now IN scope.  Card: 968 |
|  |  |  | P3-4, when the ProductIds list is omitted, should return all products (optionally paged).  Card: 967 |
|  |  |  | P3-5 (/products/{pid} is now in scope.  Card: 965 |
| 2/16/12 | R9 Draft 2 |  | Added Paragraph 2.7.15.2 to describe JSON support features.  Added JSON schemas for input data contracts to Appendix 6.2 |
|  |  |  | Added CP12 (Recommendations) back in scope.  Revised output schema to echo back input params. 6.1.28  Card: 910 |
| 2/27/12 | R9 Draft 3 |  | Modified OpenServices Products URLs (as shown in Section 2.6.1) to make them consistent with the externally facing URLs. The only change made is to the capitalization of the URL segments (lower case to match external URLs).  Card: 1000 |
|  |  |  | Modified the URL structure for the Products gateway ‘similar’ operation so that the URL matches the External URL (/products/{id}/similar).  Card: 1000 |
|  |  |  | Deleted all text that described Top20 as a separate service. Its function has been incorporated into the Products service. This change made to simply match and reflect what the code already contains.  Card: 1000 |
|  |  |  | Stores: Modified URLs of OpenServices Stores URLs to match the capitalization used by the Apigee gateway (lowercase).  Card: 1000 |
|  |  |  | Stores: Marked S5 as out-of-scope as its not been implemented (in Section 2.6.14.1.2).  Card: 1000 |
|  |  |  | Inventory: Modified OpenServices URLs to match the capitalization of the Apigee URLs.  Card: 1000 |
|  |  |  | Inventory: Deleted the OpenServices operation for /inventory (operation that retrieves ALL inventory) as this function was never implemented and not needed.  Card: 1000 |
|  |  |  | Inventory: Modified the URL for the OpenServices Inventory function that retrieves the inventory for a single store to be: /inventory/stores/{StoreId}  By adding the /stores/ segment before the {storeId} segment. This matches the URL structure at the Apigee gateway.  Card: 1000 |
|  |  |  | Reservations: Modified the URLs for the OpenServices Reservations operations to match the lowercase capitalization used by Apigee operations.  Card: 1000 |
|  |  |  | Reservations: Modified the URL of the cart pricing operation from /Reservations/ExecuteCartPricingPipeline to /cart/price to have it match the Apigee operation.  Card: 1000 |
|  |  |  | Reservations: OpenServices: Marked the ‘Validate’ operation as out-of-scope as it has not been implemented.  Card: 1000 |
|  |  |  | Customers: Modififed the URLs for the OpenServices Customers service to match the lowercase capitalization of the Apigee gateway URLs.  Card: 1000 |
|  |  |  | CP20, CP21, CP22, CP23, CP24, CP26, CP31: in OpenServices, modified the URL structure to match the respective URLs at the Apigee gateway. For instance, changed /partnerprofile to be /partner/profile.  Card: 1000 |
|  |  |  | CP24: In OpenServices, modified the URL to match the Apigee gateway URL – it is now /invoices/{id} rather than /invoice/{id}.  Card: 1000 |
|  |  |  | CP12: Corrected the URL to use productType as a query string parameter rather than the previous productFormat parameter. (pg 64-65) This corrects an inconsistency between the table on page 58 and the text description of the operation on page 64. |
|  |  |  | P3-6: clarified description to indicate that valid values of productType are ‘Movies’ and ‘Games’. |
| 02/28/12 | R9 Draft 4 |  | CP12: made the default for ‘includeUserHistory’ as TRUE. |
|  |  |  | Products P3-1, P3-2, P3-4, P3-6, P3-15, P3-16, P3-17 and P3-18: clarified the expected behavior of the PageNum and PageSize optional input query string parameters. These descriptions now match the behavior described for Stores operations. |
|  |  |  | P3-9 (Recommendations for similar titles) marked as In Scope. |
|  |  |  | CP18: fixed URL path to CreditCard Gateway operation so that it is now consistent with the URL used for CP18.  Card: 1000 |
|  |  |  | ALL CP Operations: updated schema file names, schema names, and URL paths to /v5 from /v4.  Card: 1026 |
| 04/03/12 | R10 Draft 1 |  | CP20 and CP31: Changed the data-type of the <DigitalTOSAcceptanceDate> field to be xs:dateTime. |
| 04/03/12 | R10 Draft 2 |  | CP20 and CP31: Added a <PasswordChangeDate> field to the <CustomerAccountProfile> portion of the <DigitalCustomerPartnerProfile>. This field should be included in the CustomerAccountProfile only when the profile is rendered as part of the DigitalCustomerPartnerProfile. |
| 4/5/12 | R10 Draft 3 |  | P3-9: Revised to use schema: 6.1.8.  Revised schema 6.1.8 to replace 6.1.28.  Deprecated 6.1.28.  CP12: revised to use schema 6.1.8  These changes get both recommendations operations (MoreLikeThis and PersonalizedRecommendations) to use the same output schema which is in the Products schema space. |
| 4/6/12 | R10 Draft 4 |  | CP15, CP18: added optional CVV code. If present, the card is verifified after the main operation is complete using the logic of CP32. |
| 4/9/12 | R10 Draft 5 |  | CP18: Modified CreditCardInfo (6.1.34) by adding the CVV as an attribute of the schema. This allows the CVV to be included in the POSTed data rather than as a query string param on CP18.  CP15: The CVV is no longer a query string parameter.  Modified AccountInfo (6.1.30) to add CVV as an attribute of the schema.  Modified AccountInfo (6.1.30) by moving AVSChecked and CVVChecked from being XML elements to being XML attributes.  CP32: Verify Card   * Changed from GET to POST * Removed CVV from query string * Added schema 6.1.64 CreditCardVerificationRequest * This operation now accepts the CreditCardVerificationRequest object   Modified the JSON input schemas to match these changes (modified AccountInfo and added CreditCardVerificationRequest). |
| 4/10/12 | R10 Draft 6 |  | Modified Cache Control header specifications for Products, Stores and Inventory operations.  Specified that in most cases, responses should include Cache-Control headers and Last-Modified headers. Values for each header were specified.  Conditional GET/HEAD request behavior specified for operations where appropriate.  In general, search operations that would require that on a conditional GET that the search be re-run in order to determine whether the client’s copy is still valid will not support conditional GETs. The cache header specification for such operations has been changed to reflect this. Other changes were simple clean-up of previously confusing text. |
|  |  |  |  |