**App data schema**

**General Notes:**

1. The data is structured into various properties, each having their own values
2. Relationships between different property values are mapped in the tablets data structure
3. To find the property that is used as the main reference point to a product, you need to find properties that have **is\_sku** set to true, henceforth referenced as the SKU property
4. Properties associated to the SKU property follow a naming convention
   1. Description\_<sku\_property\_name> - description property related to the SKU property
   2. Price\_<sku\_property\_name> - price property related to the SKU property
   3. UOM\_<sku\_property\_name> - uom property related to the SKU property
   4. Barcode\_<sku\_property\_name> - barcode property related to the SKU property
5. Images linked to a specific product will be present in the value attachments of the Description\_<sku\_property> eg. properties[<description\_property\_key>].values[<value\_index>].attachment.path
6. qpl:// in the image path, should be replaced by http://autospec.co.za/productmedia
7. Prices are not stored as values in a property, but in a pricing table. In order to retrieve product pricing you need to parse the fetchPrice function – example listed with the pricing\_tables sample output

**Business Object sample output**

{

“properties”: contains property definitions – see **Properties sample output**,

“permutations”: contains an array of tablets defining relationships between properties and values,

“pricing\_tables”: contains an array of pricing tables

“name”: business object name,

“id32”: business object 32bit id,

“id64”: business object 64bit id

}

**Properties sample output**

“properties”: {

property\_key: { objects are indexed by property key

"values": contains an array of property values – see **Values sample output**

"property": {

"name": property name

"value\_key": **ignore for the scope of this test**,

"key": property key,

"data\_type": flag indicating if it’s a text/numeric/input property,

"uom": unit of measure,

"is\_affected\_by\_scripting": **ignore for the scope of this test**,

"is\_browse\_graphic": **ignore for the scope of this test**,

"is\_stl\_property": **ignore for the scope of this test**,

"is\_system\_hidden": **ignore for the scope of this test**,

"is\_wizard\_skipped": **ignore for the scope of this test**,

"is\_graphic\_skipped": **ignore for the scope of this test**,

"is\_attribute": **ignore for the scope of this test**,

"is\_filter\_skipped": **ignore for the scope of this test**,

"is\_sku": flag indicating if this property is a product,

"is\_included\_in\_qty\_based\_install\_calc": **ignore for the scope of this test**,

"user\_prompt": **ignore for the scope of this test**,

"filter\_value\_key": **ignore for the scope of this test**,

"note": **ignore for the scope of this test**,

"related\_subsets": **ignore for the scope of this test**

}

},

**Values sample output**

"values": [{

"value": "Arm chair",

"value\_key": 2,

“tablet\_enabled": **ignore for the scope of this test**,

"attachment": {

"path": "qpl:\/\/ndc\/techtext\/armchair.jpg",

"name": "",

"sub\_attachments": []

}

}, {

"value": "Table",

"value\_key": 3,

"tablet\_enabled": **ignore for the scope of this test**,

"attachment": {

"path": "qpl:\/\/ndc\/techtext\/table.jpg",

"name": "",

"sub\_attachments": []

}

}]

**Permutations sample output**

Contains an array of tablets. Each tablet consists of row objects that define the relationships between different property values. Each field relates back to a **property key**, and field value relates back to a property value based on its **value key**.

Below is a sample of the data that is returned by SimpleServer.

"permutations": [

[{

"23": 1,

"24": 2,

"25": 1,

"26": 1,

"27": 1,

"28": 2

}, {

"23": 2,

"24": 3,

"25": 1,

"26": 1,

"27": 1,

"28": 3

}….

Based on the sample data provided the tablet row below

{

"23": 1,

"24": 2,

"25": 1,

"26": 1,

"27": 1,

"28": 2

}

can be translated to

{

‘code’: ‘A0001’,

‘Description\_code’: ‘Armchair’,

‘Quantity\_code’: ‘’,

‘Price\_code’: ‘fetchPrice("pricelist", [code])’,

‘UOM\_code’: ‘Ea’,

‘Barcode\_code’: ‘00000001’

}

**Pricing table sample output**

The fetchPrice function does a lookup on this data structure to retrieve a product’s pricing.

The first parameter passed to fetchPrice is the pricelist it should query.

The second paremter passed to fetchPrice is the code to look for. If the parameter passed to code starts with a [ and ends with a ], it means that a property name is specified between the []. As such the correct value for that property should be extracted from the property object.

Based on the previous example in the permutations sample out, fetchPrice(‘pricelist’, [code]) should return 99.989999999999995 as the value of the code property in that specific tablet row is ‘A0001’.

"pricing\_tables": {

"pricelist": [{

"code": "A0001",

"price": 99.989999999999995,

"description": ""

}, {

"code": "T0001",

"price": 4341,

"description": ""

}

**Data as it’s mapped in the backend**

Each line represents a unique product

