|  |  |
| --- | --- |
|  | Mitchell1 Information Systems  14145 Danielson Street  Poway, CA 92064 |

Mitchell1 Online Catalog SDK

*Mitchell1’s Extension for Creating Online Shop Manager Catalogs*

Mitchell 1 Shop Management Solutions

Poway, California

# Revision Notes

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Revision** | **Changes** |  |
| 06/21/2018 | 1.0 | Initial version of documentation |  |
| 04/22/2020 | 1.1 | Updated for Order Tracking |  |
|  |  |  |  |
|  |  |  |  |

# Table of Contents

General 5

Mitchell1 Catalog Driver 6

Custom Exceptions and Error Handling 7

Interfaces for Catalog to Implement 12

*ICatalogInfo* 13

*ICatalogVersion* 14

*ICatalog* 15

*IDeliveryMethod* 17

Interfaces that Catalog Uses 18

Common Helper Classes 35

# General

## Introduction

## Goals

This document is meant to assist in the development of a parts catalog to run from and interact with the Mitchell1 Manager SE products. Using this guide, a developer should be able to create a parts catalog which will be dynamically loaded into the host and allow the user to setup a vendor link (authentication), shop for parts, check pricing, and order parts.

## Scope

This document covers the Mitchell1 Catalog Software Development Kit which contains details on how to develop a parts catalog that includes vendor setup, shopping for parts, price check, and parts ordering.

The host application is designed to persist information such as the vehicle qualifiers, vendor setup, and all selected and ordered parts. Vendor information is **not** passed back to host application. You **must not** return sensitive information: either personal, confidential, user credentials, or other easily identifiable information from vendor setup. Instead, use tokens/cookies, which your server knows how to validate/process/identify when making calls to Vendor Setup, Go Shopping, Price Check, and Parts Ordering.

This document does not specify how the catalog should list parts, search for parts or maintain a shopping cart. These details are determined by the catalog implementer. We only require you to utilize the provided JavaScript API include for web based interaction, and conform to the in/out JSON objects for API based calls (Price Check, Order Parts).

## Applicability

Any parts catalog which is compatible with the Catalog Driver, will also be compatible with the major Manager SE and Shopkey Management SE applications. Mitchell1 requires the catalog developers/distributers to work with Mitchell1 for inclusion into the Manager SE Product.. The Shop management application only allows properly permitted parts catalogs to be used.

## Conventions

The following font conventions are used in this document:

|  |  |
| --- | --- |
| STYLE | MEANING |
| FillCart | Indicates program code or verbatim text |
| **AddLocation** | Indicates a method or property name |
| **ICartItem** | Indicates a struct, class, or interface name |

# Mitchell1 Catalog Driver

The Mitchell1 Catalog Driver is used to test the catalog development in place of the Mitchell1 ShopStream Manager software. It tests all the essential features of a parts catalog without requiring them to be digitally signed. Only catalogs that have been validated and digitally signed by Mitchell1 can be run in the actual Manager SE product.

In general terms, the steps to test a parts catalog under Catalog Driver are:

1. Build your parts catalog which implements all of the required interfaces outlined in the SDK
2. Build and run the Catalog Driver application (in the **Driver** folder)
3. Select your catalog under the *Catalog* drop-down on the user interface
4. Push the vendor setup button to enter the authentication details
5. Click the “Go Shopping” button which appears on the Catalog Driver
6. Select parts and add them to your cart
7. Transfer the carts back to the Catalog Driver
8. Verify that all of the items from the shopping cart were transferred to Catalog Driver
9. Click on “Price Check” and initiate a check of prices for your transferred items
10. Verify that the prices, quantity, and availability were updated
11. Check the “Price Check” log to verify that the request and response are correct
12. Click on “Order Parts” and initiate an ordering of the parts you price checked
13. Verify that the status of the parts has changed to indicate the order status
14. Check the “Order Parts” log to verify that the request and response are correct

## Online Catalog Configuration/Definition

Using the sample catalog driver application, you can provide the basic settings of your catalog.

**Note:** We will require the **OnlineConfig.xml** file (generated by Catalog Driver after configuring online catalog) when we validate and prepare to have your Catalog enabled in Manager.

### Basic Configuration

### 

Note: Vendor Setup Test and Go Shopping Test simply load a quick sample page with configured URL. Use the Catalog Driver’s main vendor setup / shopping carts for all validation.

|  |  |
| --- | --- |
| PROPERTY | MEANING |
| **Name** | Name of Catalog that will be presented to users |
| **Description** | Description of Catalog |
| **Support Site** | URL For Support – shown when there are HTTP/Web Errors |
| **Support Phone** | Phone for Support – shown when there are HTTP/Web Errors |
| **Base Catalog Url** | Base URL all other endpoints will utilize. E.g. https://sample.com/ |
| **Icon** | Url for getting an Icon to in various parts of host application. Url gets appending with ?width=x&height=y to give Vendor a chance to provide an icon size ideal for requested area. However, can return any size icon, it will be scaled to fit where used. |
| **Setup** | Vendor Setup page where customer would provide needed credentials to access your catalog |
| **Shopping** | Shopping Cart page. E.g. shopping.aspx |
| **Price Check** | Price Check API location – JSON Sent/Returned. E.g. api/pricecheck.aspx |
| **Order Parts** | Order Parts API location – JSON Sent/Returned. E.g. api/orderparts.aspx |
| **Order Tracking** | **Optional:** API for returning tracking information. Will be passed a tracking number, and API returns URL and short status info |
|  |  |
|  |  |

### Advanced Configuration

### 

|  |  |
| --- | --- |
| PROPERTY | MEANING |
| **bool** AllowsBlankManufacturerCode | Indicates whether the catalog allows an empty (undefined) manufacturer code for parts used in PriceCheck or OrderParts. |
| **bool** AllowsNotFoundPartsToBeOrdered | Indicates whether the catalog allows parts which were not found in the PriceCheck to be ordered. |
| **Show Deliver/WillCall** | If checked, Manager will show a “Deliver” “WillCall” choice for placing orders. Manager passes which ever choice user made. |
|  |  |
| **bool** RequiresPriceCheck | Indicates whether the host application should only allow ordering of parts that have been price checked. |
| **bool** SupportsAlternateLocations | Indicates whether the catalog supports alternate locations within the PriceCheck method. If true, the host application will allow the user to select from a list of Locations. |
| **bool** SupportsAlternateParts | Indicates whether the catalog supports alternate parts within the PriceCheck method. If true, the host application will allow the user to select from a list of AlternateParts. |
| **bool** SupportsLocation | Indicates whether the catalog can specify a location for a part. If true, the host application will display the part location. |
| **bool** SupportsOrderMessage | Indicates whether the catalog supports the OrderMessage property within the OrderRequest object. If true, the host application will allow the user to enter an order message that will be sent with the order. |
| **bool** SupportsPriceCheck | Indicates whether the catalog supports the PriceCheck method. If true, the host application will allow the user to perform a price check. If false, the host application will not allow the user to perform a price check. |

Required Catalog URLs/APIs

### Note: All configured URLs are relative to the “Base Catalog Url”. If needed to be outside this, your web server can implement Url Redirects, Url Rewrite, or other methods where Manager would still call the [base]/[component] Url, but your server would handle it or tell Manager to look elsewhere.

* Icon – Needs to return an Image File (PNG, BMP, JPG) to caller. An ideal size parameter is passed in Url: [baseUrl]/[icon]?width=58&height=28. This information can be used by your web server to find the best fitting image for the requested area (e.g. square aspect / wide aspect / etc). No resizing is needed, Manager will scale image (but, if source image is much bigger than requested size, will lose detail when scaled down, so finding the smallest/biggest image based on size is up to you).
* Setup – This needs to be a public web site that gets loaded into an internal browser (currently, Chromium based)
  + The site must have loaded the catalog-v1.0.1.js file provided (see ExampleCatalog app – configuration.html)
  + When site is loaded, any existing vendor setup data (qualifier) is available (**See** HTML help for details on JavaScript API)
  + Do not save plain text credentials or personally identifying information. You should store a token/cookie and some other small objects as necessary that you can validate the user in future calls – you will be passed this object as part of Vendor Configuration, Go Shopping, Price Check, and Order parts to verify the user / account has access
  + You have the ability to save new login token data, or cancel (**See** HTML help for details on JavaScript API)
* Shopping – This will be the web site that gets loaded to find and transfer parts back into hosting application (Manager SE)
  + Using the same JavaScript file as Setup, and qualifier (included in query string – catalogSdk.qualifier).
    - Note: You should verify all credentials as part of server back-end, example app does it in JavaScript as example only. **Important**: If credentials are missing or invalid, you must return a 403 HTTP Code. Do not add any custom UI message on your end. When Manager encounters the 403, it will message the customer and will prompt the user to go back into vendor setup area and the browser window will close automatically.
  + You can add labor/parts/notes to shopping cart – which will get price checked/ordered at some later time
  + And/or you can create a purchase order (adding parts to it) and transfer that back. Use this if your Go Shopping session directly orders/ships parts and does not need to be ordered later
  + **See** HTML help for details on JavaScript API and also look Example Catalog.
* Price Check **– API used for checking current prices and available quantity of requested parts (normally parts returned from go shopping). See example catalog app.**
  + **Request Body – application/json {**
    - **HostData: (**IHostData**)**
    - **Vendor: (**IVendor**)**
    - **PriceCheck: (**IPriceCheck**)**
    - **Vehicle: (**IVehicle**) }**
  + **Response Body – application/json**
    - { IPriceCheck }
  + Important: Ensure you return price check items in the same order as were sent to you (see example app). As well as using correct casing for JSON objects/properties, otherwise price check will not match items correctly.
* Order Parts **– API used for making an order. See example catalog app.**
  + **Request Body – application/json {**
    - **HostData: (**IHostData**)**
    - **Vendor: (**IVendor**)**
    - **Order: (**OrderRequest**)**
    - **Vehicle: (**IVehicle**) }**
  + **Response Body – application/json**
    - { OrderResponse }
  + Important: Ensure you return order parts items in the same order as were sent to you (see example app). As well as using correct casing for JSON objects/properties, otherwise parts ordering will not match items correctly.

Optional Catalog URLs/APIs

### As above, these sub URL components are built off the base url:

* Order Tracking **– API used to query for an external link to launch in external browser. Can also include a short order status message. This call will pass via HTTPS/Post the previously returned order tracking number, as well as the vendor setup qualifier. This allows you to authenticate the request (if required), and provide a URL safe to load/show in an External browser – if URL is temporary, it should stay valid for at least 8hr’s) See example catalog app.**
  + **Request Body – application/json {**
    - **HostData: (**IHostData**)**
    - **Vendor: (**IVendor**)**
    - **OrderTrackingNumber: (**String**) }**
  + **Response Body – application/json**
    - { TrackingRequestResponse }

### Object Definitions that Price Check / Order Parts API will use

This section contains all of the interfaces which the online catalog must adhere to

## IPartItem2

IPartItem2 represents a physical part that belongs to a cart. The **PartNumber** and the **ManufacturerLineCode** combined uniquely identify an individual part from the catalog.

|  |  |
| --- | --- |
| PROPERTY | MEANING |
| **string** Description | Description for the part. |
| **string** ManufacturerLineCode | Manufacturer Line Code for the part. |
| **string** ManufacturerName | Manufacturer Name for the part. |
| **string** PartNumber | Part Number of part from catalog. |
| **decimal** Quantity | Quantity requested for the during GoShopping |
| **decimal** UnitCore | Cost of the core charge for the part. |
| **decimal** UnitCost | Cost for a single unit of this part |
| **decimal** UnitList | Suggested unit price for the part |
| **string** UpcCode | Universal Product Code for the part |
| **bool** IsTire | Determines if the part is a tire or not |
| **string** Size | The size of the part (usually a tire size like “255x55 r17”) |

## ILaborItem

ILaborItem represents a labor work item that belongs to a cart.

|  |  |
| --- | --- |
| PROPERTY | MEANING |
| **string** Description | Description of the labor to be performed |
| **decimal** Hours | Estimated time that this labor item will take to complete |
| **decimal** Price | The suggested price to complete this labor item. If this property is 0, then the host application will calculate the price based on the Hours submitted and host’s labor rate. |

## INoteItem

INoteItem represents a note or comment that is added to the shopping cart. Note items should be added just before the IPartItem or the ILaborItem that they annotate. An INoteItem is not figured into the total cost for a repair order or any other billing item.

|  |  |
| --- | --- |
| PROPERTY | MEANING |
| **string** Note | A note or comment to be placed on a repair order |

## OrderRequest

An OrderRequest represents a catalog order made through the host application. It contains a list of IOrderPart along with various other properties which uniquely identify the order. An instance of this type is passed as the **Order** property of the JSON object given to the **Order Parts** API.

|  |  |
| --- | --- |
| PROPERTY | MEANING |
|  |  |
| **string** OrderMessage | User entered message sent to the catalog along with the order. |
| **IList<IOrderPart>** Parts | An ordered list of parts the user requested for ordering. These parts will be updated with the status once ordering is done. |
| **string** PurchaseOrderNumber | Purchase Order Number as determined by host application. Passed to catalog. |

## OrderResponse

An OrderResponse represents a returned object from the Order Parts REST API.

|  |  |
| --- | --- |
| PROPERTY | MEANING |
| **string** ConfirmationNumber | A confirmation number for the entire order used to verify the order was successful. If Null/Empty, order is assumed to be failure. |
| **IList<IOrderPart>** Parts | Should contain same parts that were passed in as request – however, qty/cost/etc values updated with what was ordered |
| **String** TrackingNumber | **Optional:** Used for catalog to return a tracking id/number: (Max 20 chars). |

## IOrderPart

The IOrderPart interface represents an individual line item found in a parts order (OrderRequest). Unlike the ICartItem, an order does not deal with notes or labor. Only physical parts are manipulated when performing parts ordering.

Together, the **ManufacturerLineCode** and the **PartNumber** uniquely identify a part in the order.

|  |  |
| --- | --- |
| PROPERTY | MEANING |
| **string** Description | Description for the part to order. (This property should not be updated unless it is passed in as a blank string) |
| **bool** Found | Indicates whether the catalog recognize this part. (Set by the catalog) |
| **string** LocationId | Id of the location (Set by the host application as returned from **PriceCheck**). Optional based on whether a catalog supports **Locations.** (Max length 8 characters) |
| **string** LocationName | Name of the location. (Set by the host application as returned from a **PriceCheck**) Optional based on whether a catalog supports **Locations.** (Max length 20 characters) |
| **string** ManufacturerLineCode | Manufacturer Line Code of part to order. (Set by the host application) |
| **string** ManufacturerName | Manufacturer Name of part to order. |
| **string** PartNumber | Part Number of part to order. (Set by the host application) |
| **decimal** QuantityAvailable | Quantity of this part item available at this location. (Set by the catalog). |
| **decimal** QuantityOrdered | Quantity of part ordered. (Set by the catalog) |
| **decimal** QuantityRequested | Quantity of part requested to be ordered. (Set by the host application) |
| **string** Status | Custom Status of a processed part. Reserved for future use. |
| **decimal** UnitCore | Cost of a single core item. (Set by the catalog). |
| **decimal** UnitCost | Cost for a single unit of this item. (Set by the catalog). |
| **decimal** UnitList | Suggested price for a single unit of this item. (Set by the catalog). |

## IPriceCheck

An IPriceCheck object is used to hold all of the information necessary for performing a price check on the catalog. It holds information necessary to build the request and also store the response. This is passed as **PriceCheck** property to the Price Check API call.

|  |  |
| --- | --- |
| PROPERTY | MEANING |
| **string** DeliveryOption | Delivery Option to consider for price check. – Possible values include: Deliver, WillCall (if catalog marked as showing delivery options) |
| **IList<IPriceCheckPart>** Parts | The list of parts to use when doing a price check |

## IPriceCheckPart

The IPriceCheckPart represents an individual part that is used for performing a price check. A list of IPriceCheckPart items is contained within an IPriceCheck object.

Information about the unit cost, unit list, and quantity available are contained within the ILocation object. The SelectedLocation is the original location, and it must be contained in the list of Locations. The others Locations are alternate locations. Even if a catalog does not support alternate locations, one location entry must be created to hold the pricing and availability of the part.

|  |  |
| --- | --- |
| PROPERTY | MEANING |
| **IList<IPriceCheckAlternatePart>** AlternateParts | List of alternate parts that can be substituted for this part |
| **string** Description | A description for the part. (This property should not be updated unless it is passed in as a blank string – otherwise, can be updated during Price Check) |
| **bool** Found | Indicates whether the catalog recognizes this part. (Set by the catalog) |
| **IList<ILocation>** Locations | A list of locations where the part may be ordered from |
| **string** ManufacturerLineCode | Manufacturer Line Code of part. (Set by the host application). Can be updated during Price check. |
| **string** ManufacturerName | The manufacturer name for the part. Can be updated during Price Check. |
| **string** PartNumber | Part Number of part (Set by the host application). Can be updated during price check. |
| **decimal** QuantityRequested | Requested Quantity of this part. (set by the host application) |
| **ILocation** SelectedLocation | The selected location for the part. |
| **string** Status | Custom Status of a processed part. Reserved for future use. |

## IPriceCheckAlternatePart

IPriceCheckAlternatePart represents an alternate part that can be substituted for an original part.

|  |  |
| --- | --- |
| PROPERTY | MEANING |
| **string** Description | Description for the alternate part |
| **IList<ILocation>** Locations | List of locations where the alternate part may be ordered from |
| **string** ManufacturerLineCode | Manufacturer line code of alternate part. |
| **string** ManufacturerName | Manufacturer name of the alternate part |
| **string** PartNumber | Part Number of the alternate part |
| **decimal** QuantityRequested | Requested Quantity of this part. (set by the host application) |
| **string** Status | Custom Status of a processed part. Reserved for future use. |

## ILocation

An ILocation object references a location where a part may be ordered from. It is used in an IPriceCheckPart to indicate the location of the original part along with the pricing and availability.

|  |  |
| --- | --- |
| PROPERTY | MEANING |
| **string** Id | Id of the location (as determined by the catalog) (Max length 8 characters) |
| **string** Name | Name of the location (Max length 20 characters) |
| **decimal** QuantityAvailable | Quantity of this part item available at this location. |
| **decimal** UnitCore | Core charge for this item |
| **decimal** UnitCost | Cost for a single unit of this item |
| **decimal** UnitList | Suggested price for a single unit of this item |

## IHostData

IHostData represents information that the host application makes available to catalogs.

|  |  |
| --- | --- |
| PROPERTY | MEANING |
| **string** ApplicationTitle | Title of the host application. |
| **string** ApplicationVersion | The first 3 components of the current version of the host application. |
| **decimal** LaborRate | The default hourly rate the host application uses to calculate labor items |

## IVehicle

IVehicle represents the user selected vehicle for use in filtering catalog part searches. The vehicle is defined in the host application and then passed to the catalog on a call to **GetCatalog**. Not all properties in the IVehicle object are guaranteed to be defined. Catalogs should attempt to identify the vehicle based on available information (starting with VIN), and fallback to other values given (e.g. VIN is not right, check Year / Make / Model and/or Aces Id’s). Important: This object can also be omitted in older versions of Manager.

|  |  |
| --- | --- |
| PROPERTY | MEANING |
| **int** AcesBaseId | ACES (Aftermarket Catalog Enhanced Standard) Base identifier for this vehicle |
| **int** AcesEngineId | ACES Engine identifier for this vehicle |
| **int** AcesId | ACES identifier for this vehicle. |
| **string** Body | Body style of this vehicle (like: "sedan" or "wagon") |
| **string** Brake | Description of brakes installed on this vehicle |
| **string** DriveType | Drive train type for this vehicle |
| **string** Engine | Description of engine in this vehicle |
| **string** Gvw | GVW (Gross Vehicle Weight) of this vehicle |
| **string** Make | Make of the vehicle |
| **string** Model | Model of the vehicle |
| **string** SubModel | Submodel of the vehicle |
| **string** Transmission | The type of transmission installed on this vehicle. |
| **string** Vin | VIN (Vehicle Identification Number) of the vehicle |
| **int** Year | Model year of the vehicle |
| **int** AcesEngineBaseId |  |
| **int** AcesEngineConfigId |  |
| **int** AcesSubmodelId |  |

## IVendor

IVendor represents a parts Vendor. The Code and Name are configured in the host application. The Qualifier represents custom information as defined by the catalog – this is passed to all calls, but only vendor setup/configuration can make changes to this that will be saved.

|  |  |
| --- | --- |
| PROPERTY | MEANING |
| **string** Code | A short code which uniquely identifies the vendor |
| **string** Name | Name of the vendor |
| **string** Qualifier | String that can be used by the catalog to store custom vendor information. **Important:** This value can only be changed and persisted when returned from VendorSetup. Any other time, changes to this field will be discarded. |

***TrackingRequestResponse***

Response object for returning basic tracking information and a URL to show in external browser (e.g. Chrome/Firefox/Safari/Edge)

|  |  |
| --- | --- |
| PROPERTY | MEANING |
| **string** ExternalTrackingUrl | Valid URL – either https or http that shows the order status when viewed in external browser |
| **string** StatusDisplay | **Optional:** A short message (180 chars or less) about status |
|  |  |

Price Check / Order Parts Logging / Debugging

* Request & Response messages (raw JSON) is logged to: C:\Users\<currentuser>\AppData\Local\M1-SK\<CatalogName>.OrderParts.log and PriceCheck.log. Useful when determining what was sent & received to your catalog from Manager.
* Catalog Driver is your first debugging tool
* However, when there is a question about behavior and Manager SE, Fiddler can be used on Local system to capture API calls (enable Fiddler HTTPS capture)
* Additionally, the Dev Tools can be activated from the window’s application menu to debug your JavaScript / web sources.
* If Vendor Setup or Transfer Parts in Go Shopping does not appear to work, check the JavaScript dev tools for any console JavaScript issues, and/or run DebugView (from Sys Internals) to capture any possible JavaScript to Managed code conversion errors in browser process. Also, if running Manager, check C:\ProgramData\M1-SK\ShopStreamShell.log for any logged errors