# Mitchell1 Catalog Driver

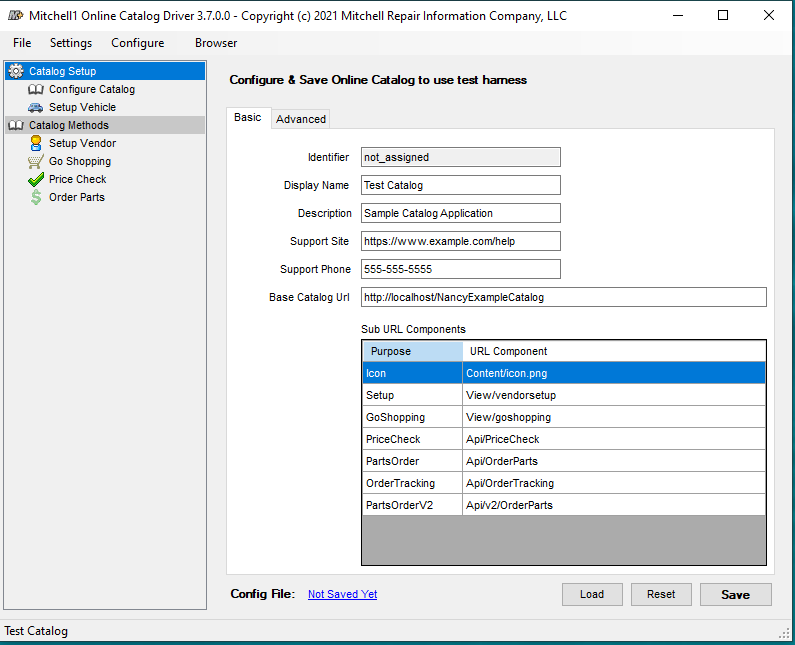
Copyright © 2020 - Mitchell1 Repair Information Company, LLC

## Introduction

This document describes the Mitchell1 Catalog Driver Online edition - which is a stand-alone development tool for testing Online Catalog Integration – for ultimate use with Manager SE. **Important:** The catalog driver is a development aid - all final testing should be performed within the Manager application before release – you will need a Signed XML file to perform that test. The driver is not attempting to replicate all behavior of Manager.

## Basic Usage Guide

Starting the Catalog Driver executable will land you on the home page.



### Configuration

Driver and catalog configuration are located in the “Catalog Setup” -> “Configure Catalog” section of the tree view.

#### Catalog Configuration

On the **Basic** tab, there are some configuration inputs.

|  |  |
| --- | --- |
| PROPERTY | MEANING |
| Display 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 |
| ApiVersionLevel | Minimum API level that this Catalog supports (should be 1 for all catalogs) |
| 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 |
| GoShopping | Shopping Cart page. e.g. shopping.aspx |
| PriceCheck | Price Check API location – JSON Sent/Returned. e.g. api/pricecheck.aspx |
| PartsOrder | (Will be sunset in the future) Order Parts API location (v1, single purchase order returned) – JSON Sent/Returned. e.g. api/orderparts |
| OrderTracking | API for returning tracking information. Will be passed a tracking number, and API returns URL and short status info |
| PartsOrderV2 | Order Parts API location (v2, multiple purchase orders returned) – JSON Sent/Returned. e.g. api/v2/orderparts |

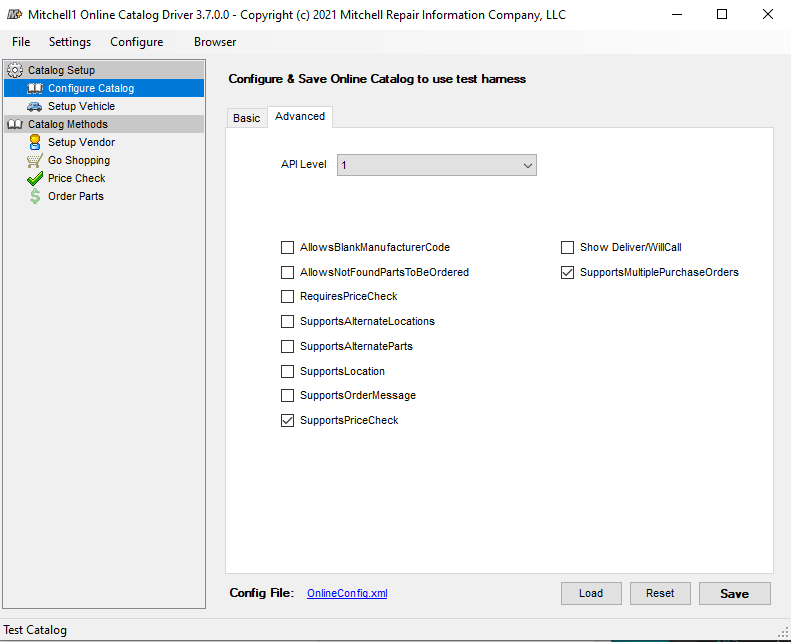
The “Sub URL Components” table should be pre-populated the first time ever running this program or if you have an existing OnlineConfig.xml file present:

|  |  |
| --- | --- |
| Purpose | URL Component |
| Setup | View/vendorsetup |
| GoShopping | View/goshopping |
| PriceCheck | Api/PriceCheck |
| PartsOrder | Api/OrderParts |
| OrderTracking | Api/OrderTracking |
| PartsOrderV2 | Api/v2/OrderParts |

* If they are not populated, the grid is editable and you will need to populate the values.
  + If any changes were made, click Save (see below) to persist them.

You can change the “Base Catalog Url” to point to where your catalog API will be listening.

The **Advanced** tab contains various settings that control how your catalog behaves. You will want to review those settings and configure applicable settings based on your catalog behavior – should currently work with Manager Product Owner to work on best integration settings for your needs.



* By Default (or if you click “Reset”) it is setup to point at the Mitchell1 Example C# Catalog.

|  |  |
| --- | --- |
| 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. |
| **bool** ShowsDeliverWillCall | Indicates whether the host application (Manager SE) will show a “Deliver” “WillCall” choice for placing orders. Manager SE 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. |
| **bool** SupportsMultiplePurchaseOrders | True if the catalog provides an OrderParts endpoint that will return one or more purchase orders |

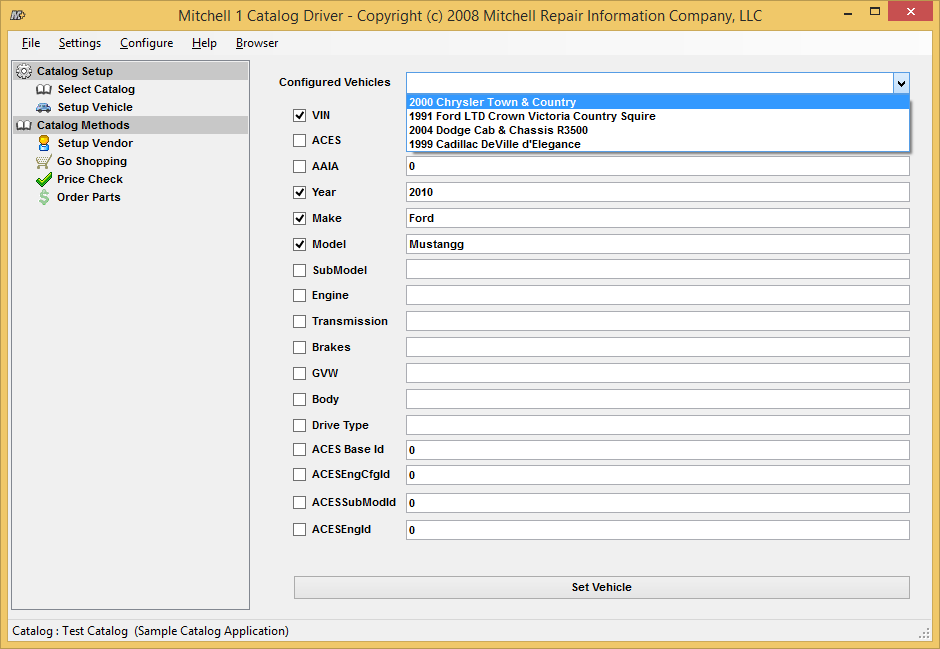
These three buttons cover both the Basic and Advanced tabs:

* **“Load”** button allows you to import an existing configuration (perhaps a signed config you recevied or a file you backed up).
* **“Reset”** button restores the settings back to demo mode.
* **“Save”** button saves your changes to disk into the *OnlineConfig.xml* file. Once saved (ever), the **Config File** link will be active, and you can click it to quickly access your saved config file (useful for uploading to M1 to be signed).

**Important**: You will need to have a valid URL/site with the above configuration. The SDK includes a very basic C# sample web site/service. Under Sample\Bin\ExampleCatalog.exe (C# Project sources lives under Sample\Source folder). This app will ask for elevated credentials to use http.sys on Windows for a self hosted localhost site (needs elevated credentials as it enables the site for local IPs as well). Once you have your own site up, running this sample is not required, simply point configuration at your URLs.

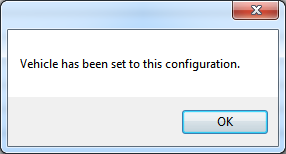
#### Vehicle Configuration

Click on the *Catalog Setup -> Setup Vehicle* link on the left to configure a vehicle:



As you can see from the Drop-Down labeled **Configured Vehicles**, there are a few pre-configured vehicles to save you some work.

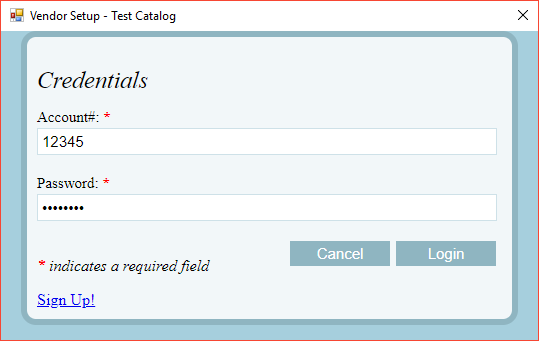
Click on the **Set Vehicle** button to save your vehicle settings and you should see this dialog:



#### Vendor (Auth) Setup

Click on the **Setup Vendor** link on the left to configure a vendor.

Clicking on the Setup Vendor button will bring up the catalog authentication dialog. If you have configured your online catalog correctly, and it is reachable, your vendor setup page will be loaded.

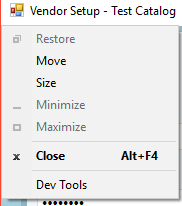


If using the provided Sample Catalog Web Site, the credentials are:

* Account#: **12345**
* Password: **password**

click Login. This will save the returned object/qualifier locally for later usage in Go Shopping / Price Check / Order Parts.

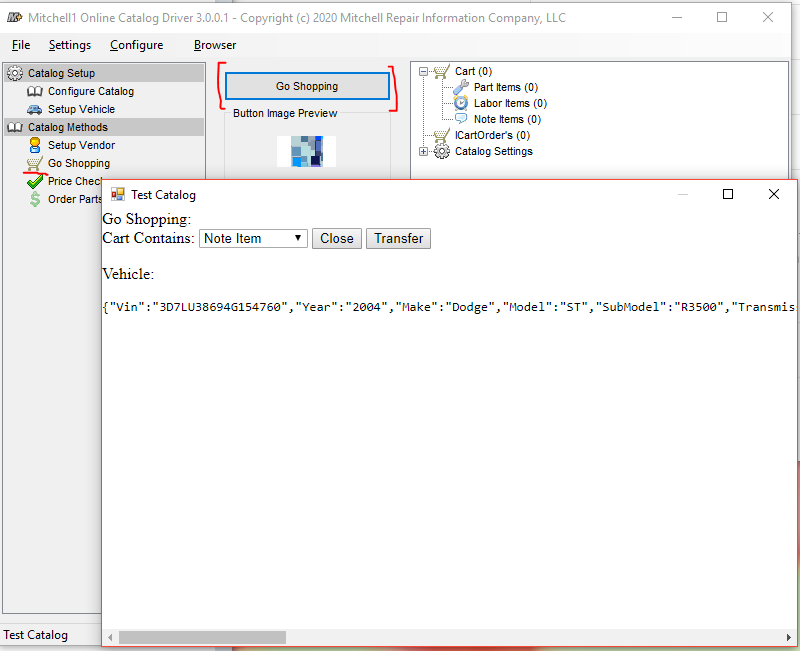
**Important Notes:**

* The HTML Cancel button will close the window from Javascript. Login will close the window, and persist the returned data blob you provide (suggest OAth token, or some other limited scope auth cookie your Server knows how to interpret during later calls)
* Sign Up button illustrates how you can break off this small window to an external browser the client is running (use for help, support, sign up, etc)
* Also, the Window icon (applies to Go Shopping window as well) has an integrated Dev Tools you can use to debug/test
  + 

### Functionality

#### Go Shopping

In the tree view, select **“Go Shopping”.** Then, click the **“Go Shopping”** button to bring up a catalog window.



**Important Notes:**

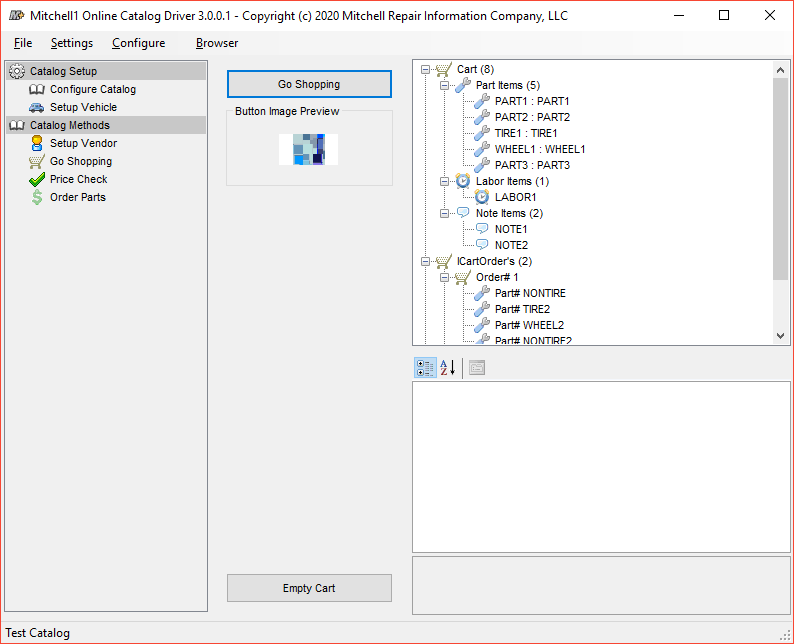
* The image/button display shows the configured Icon/Image for your catalog (from your Icon URL)
* Clicking Close (controlled via your page/javascript) will not transfer anything
* Clicking transfer will transfer what you have populated in your cart. The sample catalog fills the cart with some parts/items and even some directly ordered parts (Important: direct ordering is a good feature to support, but transferring parts without initial order is the most used shop workflow to build up estimates to show to customers – and must be supported as well – using pricecheck/ordering via REST APIs later).
* Refer to the ExpectedHttpErrorCodes.xlsx document on what HTTP errors are expected or not expected from the various integration points

The TreeView in the upper-right allows you to browse the containers that represent the host application (normally Manager SE, but when using the Catalog Driver, it acts as the host). You may click on the part, labor, and note items to view items transferred over from the catalog. You can also click on ICartOrder’s tree view item to expand any parts ordered via GoShopping’s ICartOrder interface. Clicking on the **Empty Cart** button will remove all items from the containers (resetting the cart to its empty state.)

**NOTE: Catalog developers must empty the Shopping cart when completing the Order Parts session, whether the parts were ordered or not.**

**This will prevent incorrect parts from being used in future Go Shopping/ Order Parts sessions.**

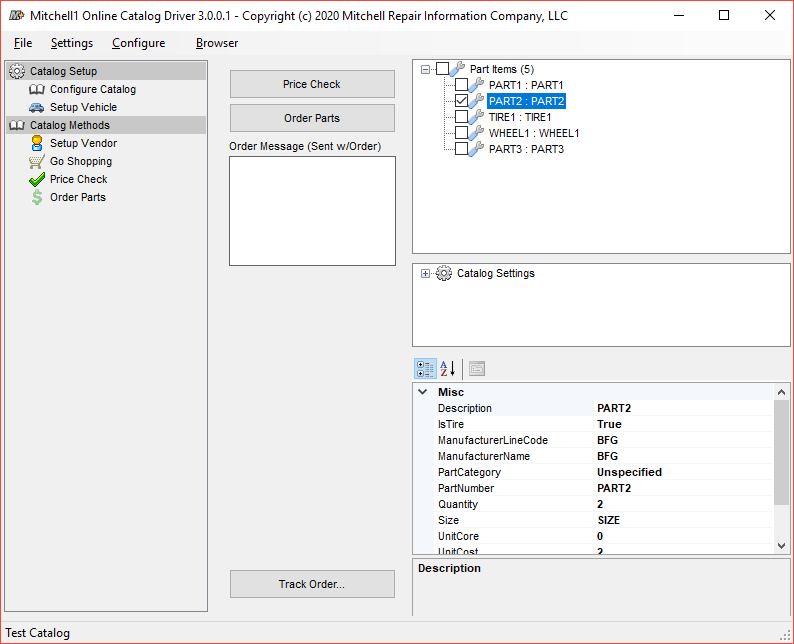
Below is a view of what the Catalog Driver looks like after transferring the part:



The part transferred has been selected and its properties are shown in the bottom right (the property grid).

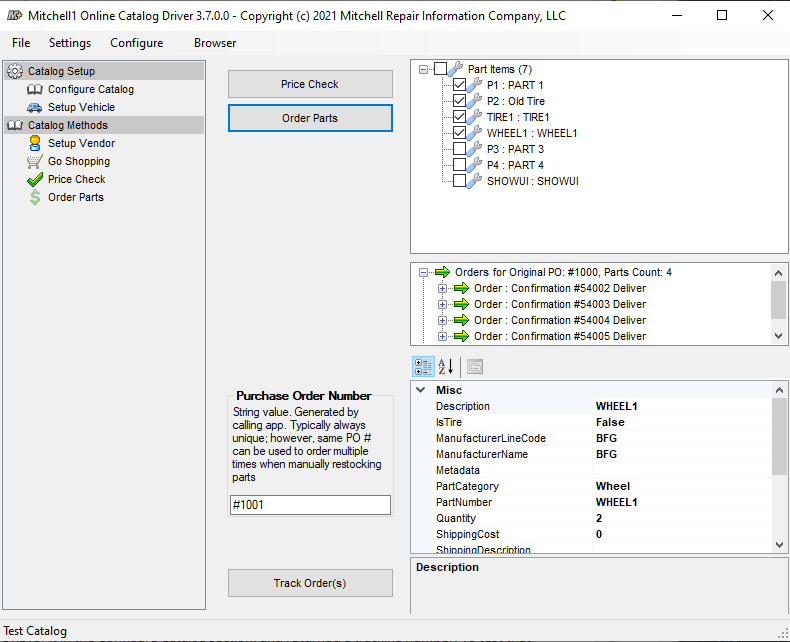
#### Price Checking / Ordering

At this point you should click on the *Catalog Methods -> Price Check* link on the left in the tree view:

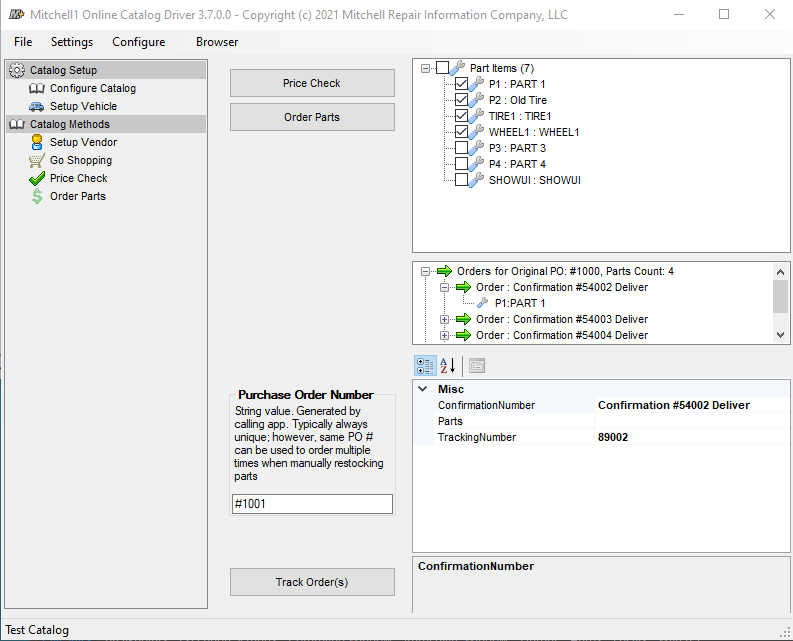


You will need to check the parts you want to Price Check / Order Parts in the TreeView control on the upper-right. This will enable the **Price Check** button. If your catalog properties require that a price check be performed before you can order parts, the **Order Parts** button will still be disabled.

Click on the **Price Check** button to invoke your configured PriceCheck API call to verify the price and availability of parts. Also, alternate locations and their associated quantities may be returned from the web service call. All of this information should be populated in the TreeView control and can be browsed by selecting parts and viewing their properties in the property grid.

Clicking on the **Order Parts** button will make another call to the configured Order Parts API. This will attempt to order the parts at the locations you selected with the quantity you requested. 

You should be able to view the metadata of each purchase order by clicking on an order.



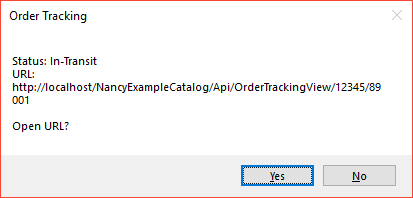
Each part’s metadata can be seen in the same manner.

#### Ordering Tracking

Versions of Manager newer than 8.2.1 support loading a URL from your REST services to bring up your order tracking page. This tracking can be returned via the Go Shopping direct orders, as well as from the Order REST API. If your catalog supports tracking, you would have configured the REST URL for it in the Configure Catalog section, and returned a tracking number. To test that functionality order a part with the tool, the “Track Order(s)” button is for this purpose. Its flow changes based on the number of purchase orders returned from the Order Parts call.

##### Single Purchase Order

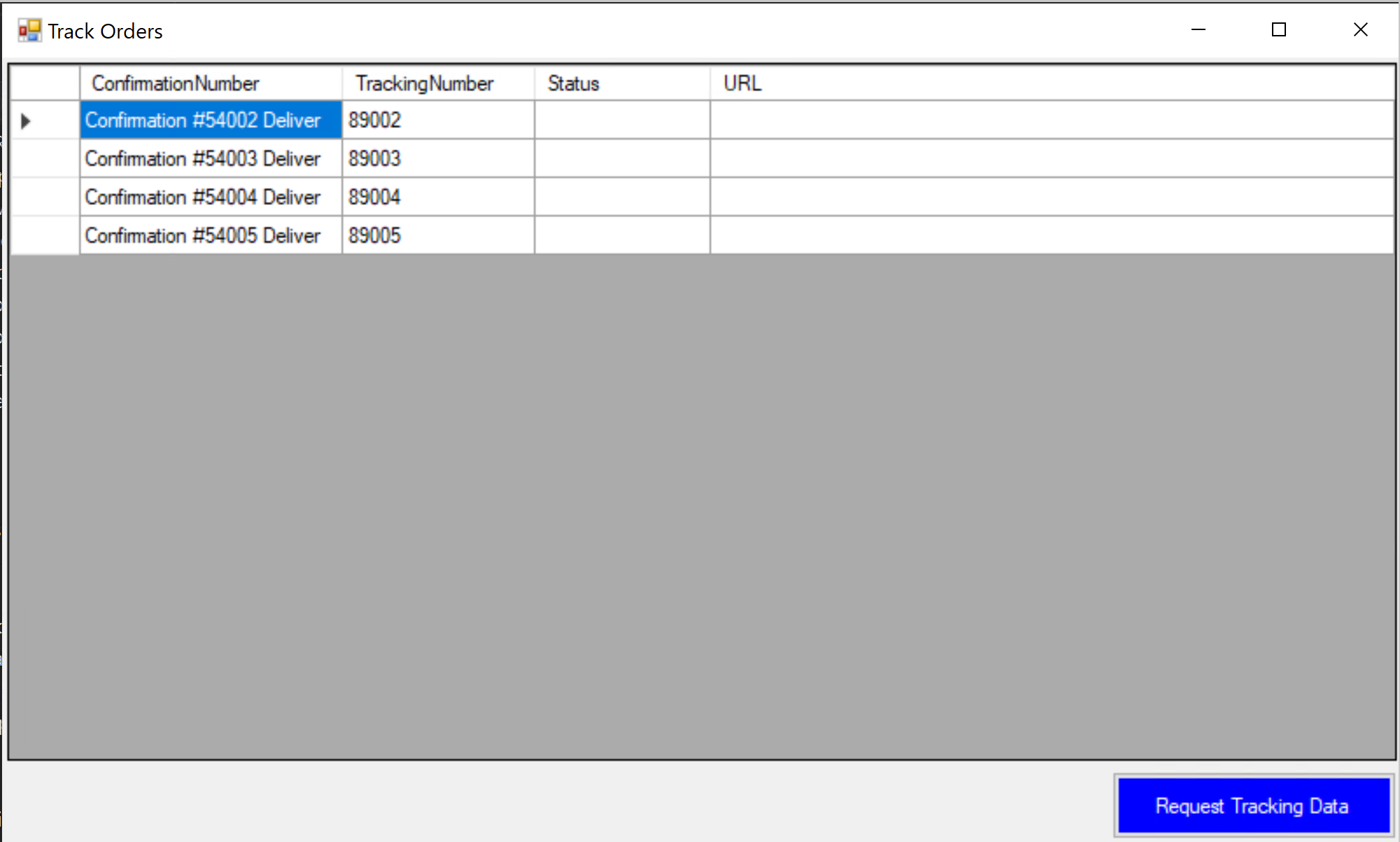
When exactly one purchase order is returned that has a valid tracking number, a dialog pops up showing both the status string you returned and the URL (clicking Yes will open the URL in the default web browser).



* Manager would present a clickable link to open your tracking page in external browser.

##### Multiple Purchase Orders

When more than one purchase order is returned, the Track Orders window opens with a grid that displays all purchase orders. The grid is read-only, the columns are resizable and the window can be resized down to a minimum of 800 x 450.



Status and Url will be filled in once the “Request Tracking Data” button is clicked and all purchase orders get their data back.

Status will be whatever your catalog returns back except in these cases:

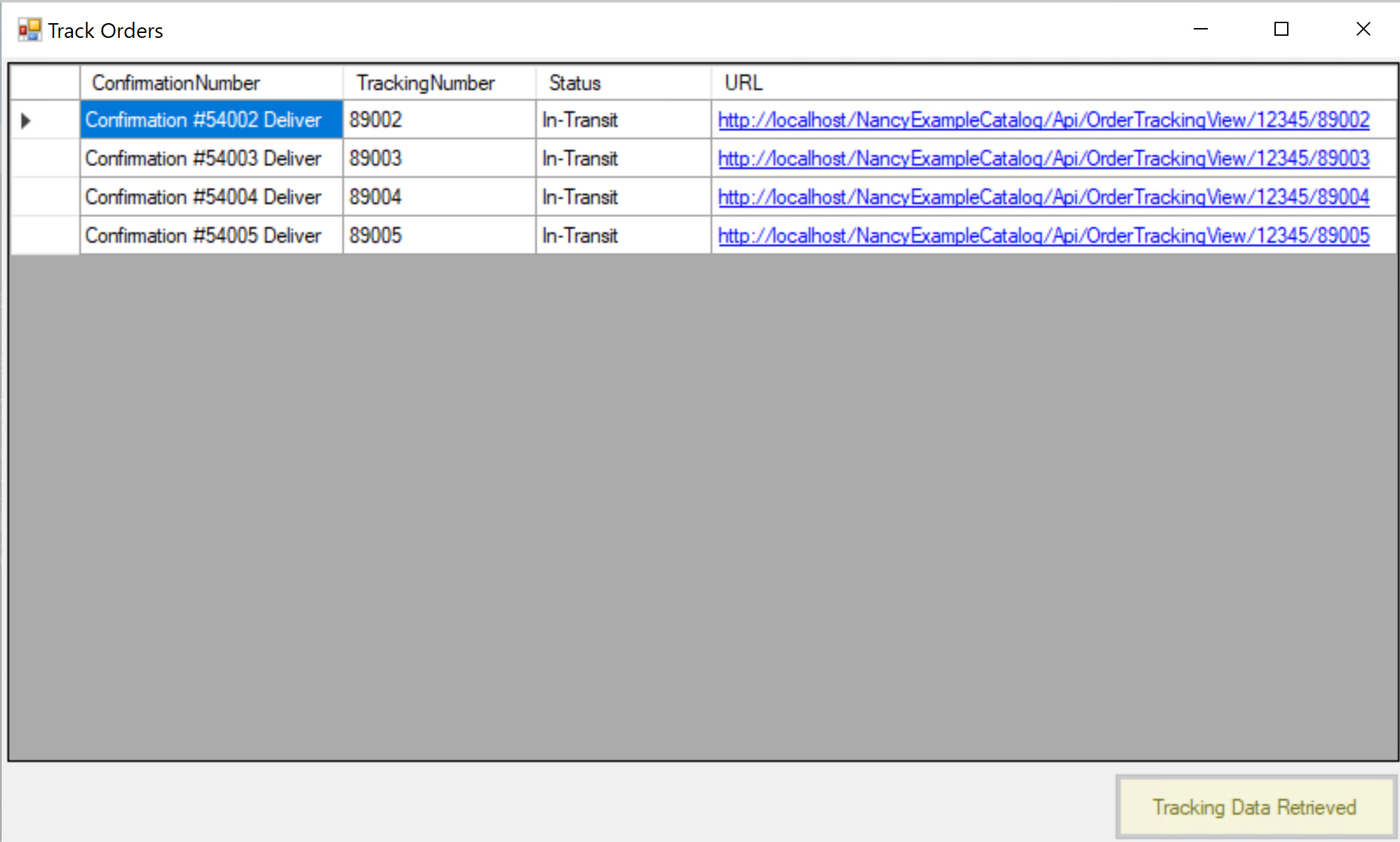
* Handled exception
* A purchase order has an empty tracking number
* No or invalid tracking information was returned for the purchase order

The URL column with be either a valid URL or error information for that purchase order. When it is a valid URL (scheme is http or https), clicking on the grid cell will open it in the default web browser.

Once the “Request Tracking Data” button is clicked, it will be disabled and update its background and foreground color and text based on the state of requesting data.

* “Requesting tracking data…”
* “Request Error” – if a handled exception occurred
* “Tracking Data Retrieved” – once all requests have been fulfilled (does not indicate success of all requests)

Example of finished request:

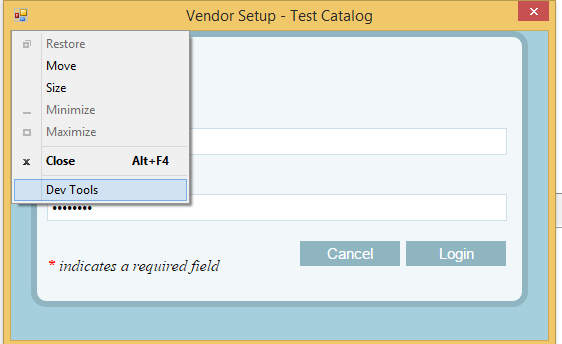


After the data tracking request is complete, the button will remain disabled. To retrigger the request, you will need to close the window and re-click “Track Order(s)”.

## Command Line Options / Browser Support

The catalog driver has some Mitchell Browser testing tools. In the file menu, under “Browser”, you can load an HTML5 test page, or a simple browser window you can type a URL to navigate to. These allow you to test the internal Web Browser that would be presented for Vendor Setup / Go Shopping web page. You can use this to view the abilities, behaviors, and to debug (with dev tools) your development site as needed.

Additionally, Vendor Setup and Go Shopping Windows have an option in windows’ application menu to launch the tools as well:



Command line flags affect the driver by:

* **--remote-debugging-port=####** (where #### is a valid port number). This specifies a port to startup a remote debugging accessible. At this time, the Mitchell1 Browser is using Chromium/CEF. You can open <http://localhost:####> in Chrome and interact with the browser.
* **Selenium Web testing:** If desired, you can use Selenium via Chrome Driver to run UI automation testing against your site with our browser. Add argument when launching Catalog Driver app - either: **--action=vendor** or **--action=goshopping** (you must have first configured the catalog via normal UI and saved/closed the app). Either argument will directly load a browser with the specified site and will allow Chrome Driver to attach or Selenium tests to run.
* **--browser=[url]** (where [url] is a URL to load). This flag will directly open a browser to said URL – also potentially useful for Selenium web testing.