# Design Document for BrickLink Scraper

Contents

[Design Document for BrickLink Scraper 1](#_Toc97820694)

[1 Overview 1](#_Toc97820695)

[2 Prerequisites 1](#_Toc97820696)

[3 Release Notes 1](#_Toc97820697)

[4 Architecture 2](#_Toc97820698)

[4.1 Inputs 2](#_Toc97820699)

[4.2 Cache 4](#_Toc97820700)

[4.3 Outputs 4](#_Toc97820701)

[4.4 Application 4](#_Toc97820702)

[4.5 Data 7](#_Toc97820703)

[4.6 Outputs 7](#_Toc97820704)

## Overview

The purpose of this design document is to record and communicate the architecture, process, and design decisions that when into the development of **BrickLink Scraper**. The program is an executable that given a configuration and an inventory produces a document containing information about the provided inventory including part quantities and price by seller.

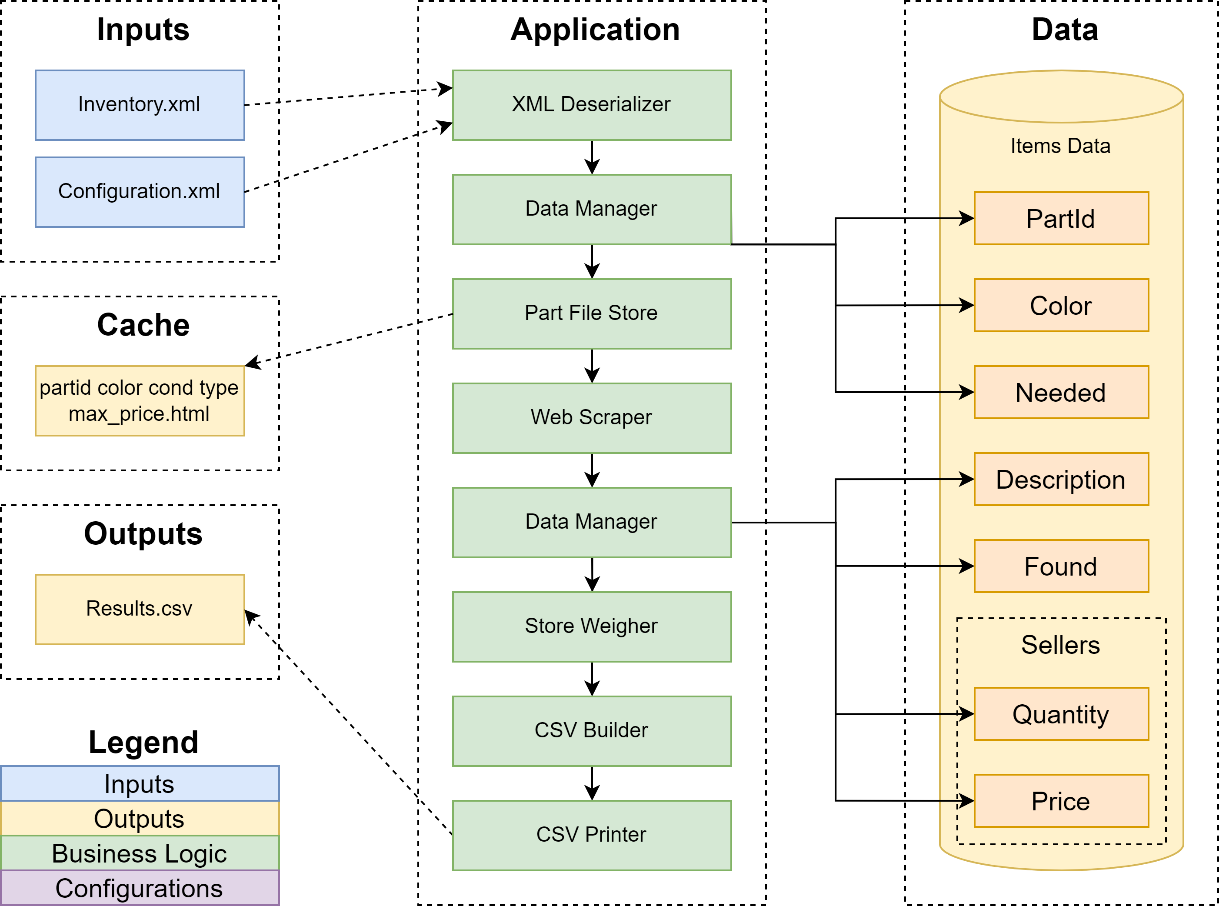
## Prerequisites

* Requires this installation of **Chrome** browser *97.0.4692.71*, *98.0.4758.102*, or *99.0.4844.35*.
* Requires user access to all files and folders within the executable directory.
* Does this require dotnet to run? **(TODO)**

## Release Notes

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Document** |
| 1.0.0 | 3/10/2022 | Release Notes/Release Notes V1.0.0.docx |

## Architecture



### Inputs

#### Configuration.xml

This xml file defines the values that will configure the application. This file can be found in the executable directory within the **Inputs** folder. Any values missing from the configuration file will use defaults instead. Here is a list of configurable values.

|  |  |  |
| --- | --- | --- |
| **Configuration** | **Default** | **Description** |
| InventoryFile | Inventory.xml | The name of the xml file that contains the inventory. Must be relative to the **./Inputs/** directory. |
| CacheTimeHoursPolicy | 24 | Defines how many hours from when the cache file was saved that it should still be considered relevant. |
| HtmlLoadTimeMs | 600 | Delay time between when the browser will navigate to the next part. Can affect accuracy if the browser navigates away before it can cache results. |
| SortPartsBy | 1 | Enumeration for “**Search By**” criteria. 1 = Lowest Price, 4 = Highest Quantity. |
| MinQuantityPercent | 10 | Percentage of the needed quantity of a part that defines the minimum quantity a seller must have to be listed. |
| NumberOfSellers | 40 | Quantity of sellers to be displayed in the final **./Outputs/Results.csv** file. |
| PreferredSellers[] | - | An array of preferred seller. These will be filtered to the top of the sellers list. |
| 🡪 Seller | Empty | The name of the preferred seller. Must match the name of the seller on the webpage and be in UTF-8. |
| IgnoredSellers[] | - | An array of ignored sellers. The will not be included in the sellers list. |
| 🡪 Seller | Empty | The name of the ignored seller. Must match the name of the seller on the webpage and be in UTF-8. |
| SellerWeights | - | A collection of weights that will be used to determine the success rate a given seller has over the inventory. |
| 🡪 PricePoint | 1.2 | The influence price has on determining seller success. |
| 🡪 Quantity | 1.5 | The influence quantity has on determining seller success. |
| 🡪 | 1.9 | The influence rarity has on determining seller success. |

#### Inventory.xml

This is a collection of parts that is required to inform the application of which parts and parameters needed to search for parts. The name of the document is configurable and should be relative to the executable directory within the **Inputs** folder.

|  |  |  |
| --- | --- | --- |
| **Value** | **Required** | **Description** |
| Inventory[] | Yes | An array of all parts that the application will process. |
| 🡪 ItemId | Yes | The part id. |
| 🡪 ItemType | No | The item type. |
| 🡪 Color | Yes | The color code. |
| 🡪 MinQty | Yes | The absolute value of quantity of parts needed. |
| 🡪 MaxPrice | Yes | The average price of the part. |
| 🡪 Remarks | No | Not used at this time. |
| 🡪 Condition | No | Determines whether the part must be new. |

**Note:** *MinQty* and *MaxPrice* are misleading values. They are outputs from another tool and have been mapped differently within the application.

### Cache

Application caching is in the form of saved html files. Html files are categorized by name using the following naming convention, ***ItemId\_color\_condition\_itemType\_maxPrice\_.html***. Files are stored in the executable directory under **HTML\_Files**. Files that are older than the configured *CacheTimeHoursPolicy* will be remove from the directory. If all parts within an inventory have valid associated html files, the application will not load *chromedriver* which will drastically speed up runtime performance.

### Outputs

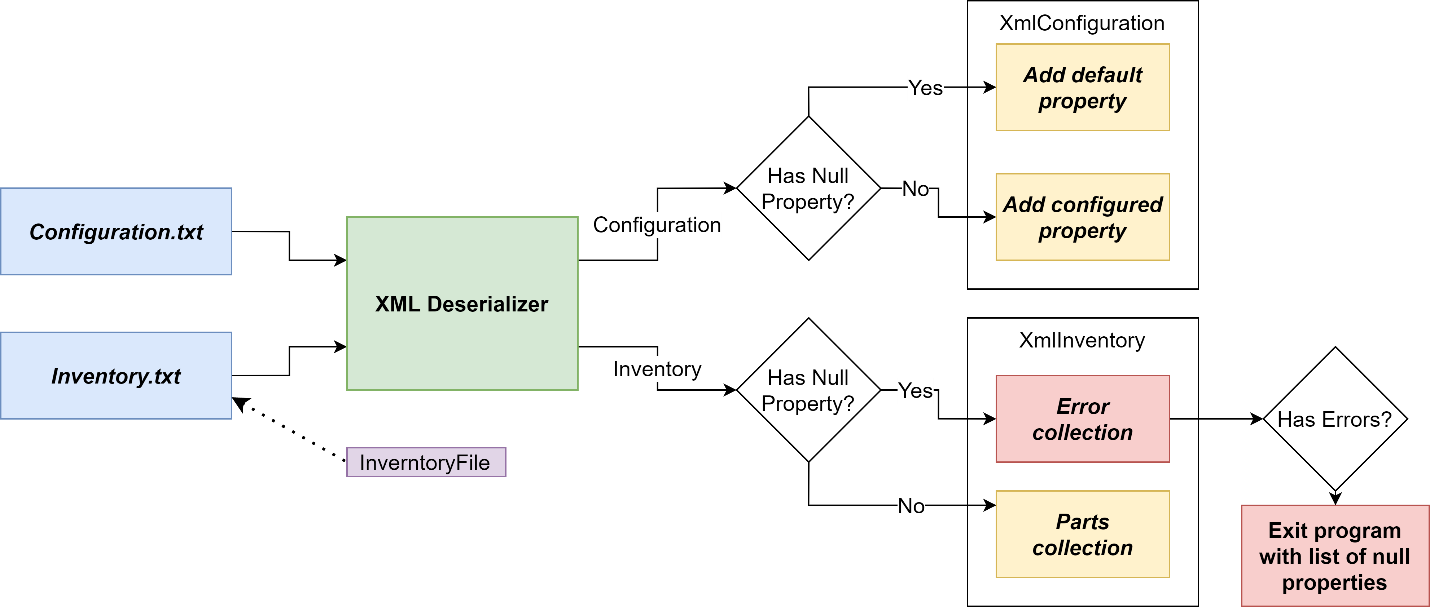
The application generates a CSV file titled Results.csv. The file contains a structured table organized into parts per row. Parts contain a part ID, color, description, product needed, product found, and a list of sellers with quantities and prices for each. Sellers are listed in order of success where success is defined by price, quantity, and part rarity.

### Application

This section will define the application and how it works. Each major method will have it’s own sections, and important design decisions will contain associated charts.

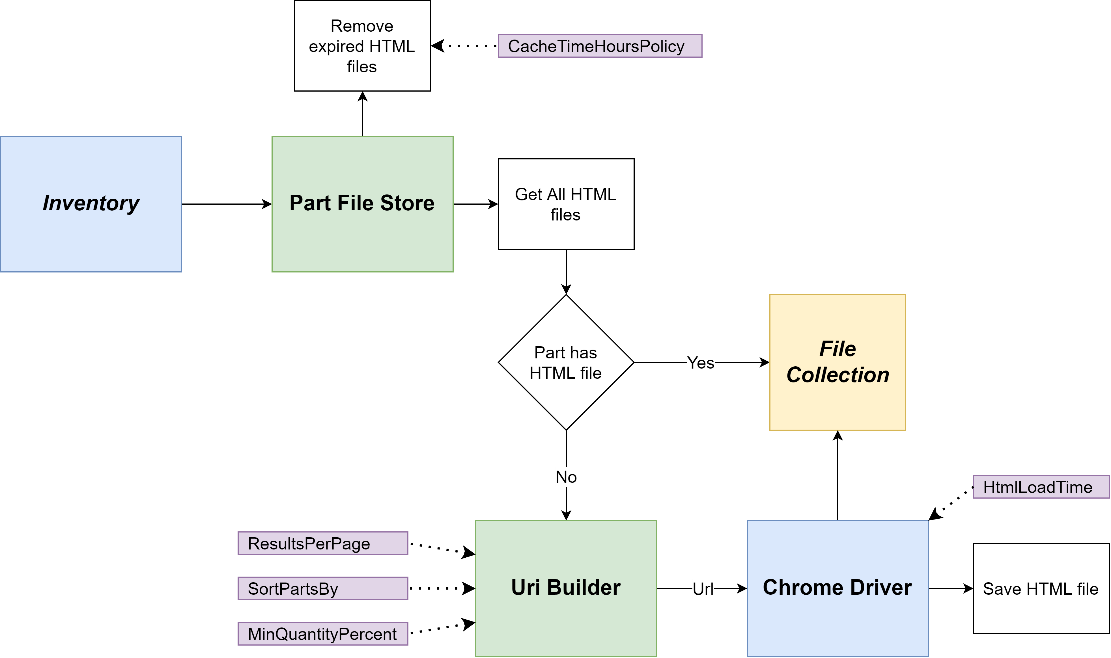
#### XML Deserializer

The application begins by taking the configuration and inventory xml files and running them through XML Deserialize methods. The configuration will fall back to default values when there are errors parsing the configuration file. The inventory is validated and will return errors if the xml is incorrectly formatted or is missing required fields.



#### Part File Store

This manages the application cache by cleaning up expired html files, loading html files that match the parts in the inventory and runs selenium chromedriver on the remaining parts and saves the generated html files into the cache. What the chromedriver returns will depend on the provided URI which is generated using a URI builder. Configurable values include the cache expiry time in hours, the results per page, the category by which to sort, the minimum quantity percent based on parts needed, and the HTML load time.



#### Web Scraper

The web scraper is a simple pass-through method that takes an HTML file and using *HTMLAgilityPack* to scrape the html for desired data, including description, each seller, the seller’s quantity, and price. This is the most brittle component of the application because any changes BrickLink makes to their DOM will potentially break the pattern that the scraper is currently programmed to use.



#### Data Manager

The application uses the data manager twice, the first time is only to relay the part number, color id, and quantity needed to the data model that gets used for creating the final output. The second time the data manager gets called is to put the scrape data returned by the web scraper and populate the same data model.

#### Store Weigher

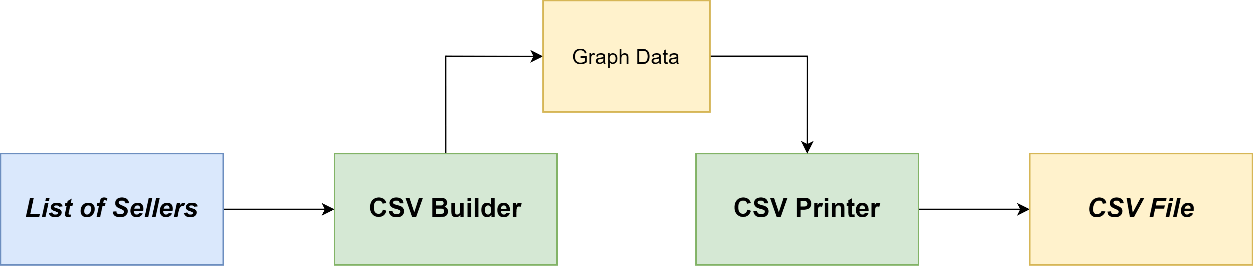
Using the data model after being populated by the web scraper, the store weigher calculates the scrape data against metrics defined by the system and multiplies each metric by an associated weight. The resulting scores are sorted by highest value. Preferred sellers are shifted to the top of the list but are still sorted by their score. Ignored sellers are removed prior to running metrics. Configurable values Include price point, quantity, and rarity for the seller weights, preferred sellers, ignored sellers, and the total number of sellers.

A picture containing diagram

Description automatically generated

#### CSV Builder and Printer

The CSV Builder takes the data model and the list of weighted sellers and organizes the data into a table format. The table format object is taken by the CSV Printer which populates a file with the table data.



### Data

Data is handled in the application using a singleton model. It is populated with data received from the input inventory file, which includes the part id, color id, and quantity needed. The remaining values, product found, part descriptions, the qualified sellers per part and their parts quantity and price are all gathered from the HTML scraper.

### Outputs

The only output from the system is a file titled Results.csv containing the table data with a comma delimiter. File should be closed between runs or renamed before opened. The service will return an error if it cannot save file data because it's being used by another process.