# Web Data Collection Considerations:

# Volume of Requests

Entities that access websites through an automated device including but not limited to data collection, scraping, harvesting or crawling (“*Data Collectors*”) should be mindful that the amount and rate of their requests to websites do not cause an adverse impact on those websites. Web servers may slow down or crash if they are tasked with serving too many requests in any given period. There are a number of volume-related parameters that can be taken into consideration by Data Collectors in limiting the potential harm to the target websites while running an automated program (referred to herein as a “*script*”). The FISD Alternative Data Council generally recommends that firms factor the below considerations into the design of their Data Collectors to reduce the probability of an adverse impact on the target website, unless circumstances merit otherwise.

Design Considerations

1. **Use of a static or dynamic/random download delay**: It is possible for a script to set a static (*e.g.* once every five seconds) or dynamic (*e.g.* random period between 2 and 7 seconds) delay between page requests.
2. **Use of “auto throttle” and similar technologies**: Many open-source libraries for web data collection offer functionality that will automatically adjust the frequency of page requests based on the current webserver load, for instance by inferring such load based on the latency between requests and responses.
3. **Calculating average daily loads**: A number of analytics firms offer data about website traffic that would allow a Data Collector to calculate the number of average page loads per day. A Data Collector should consider applying this data when determining the frequency in which the script accesses the website(s). For example, by ensuring that the percentage of page requests it makes is under some percentage of average daily page loads.
4. **Collecting data during low-traffic timeframes**: Consider limiting data collection to times when a website is less busy, such as overnight hours. This can be based on either website traffic data or inferred based on the geographic location of the site and the site’s content. It is helpful to note that applying randomness to the start time of the script is helpful so that different Data Collectors are not issuing requests concurrently—for example, a data collection script may start within a random period between 2AM and 3AM.
5. **Limiting the number of concurrent requests**:It is possible to keep a script from issuing additional requests until the webserver responds to outstanding requests. Consider keeping the number of outstanding concurrent requests below a predefined limit.
6. **Crawling at “human speed”**: If a human collecting the data by copying and pasting would load a new page once every five seconds, consider limiting scripts to a similar rate.
7. **Incorporating “speed bumps”**: Similar to applying “human speed,” consider including speed bumps that pause the script at certain intervals (e.g., script pauses for 5 seconds after 10 page loads).
8. **Following robots.txt “Crawl-delay” directive**:Website owners can specify a number of seconds that a script should wait in between successive page loads set forth in their robots.txt.