Automating File Download Instructions: Fall 2023

Background

Ginnie Mae makes available certain financial data files to users who have established a user profile for the GinnieMae.gov website. When browsing GinnieMae.gov users must be logged into their profile to access the pages containing these files and to download files directly. If the user is not logged in to their profile they are redirected to a login page. This process can sometimes cause difficulties for those seeking to write automated scripts to download files. This document is aimed at providing script developers with the necessary information related to how GinnieMae.gov manages the login and file download process so that they may successfully create their scripts.

Login Process

The process diagram in Figure 1 (enlarged in Appendix 1) provides an overview of the user login and file download process along with the interaction of www.ginniemae.gov and bulk.ginniemae.gov. While it is not necessary for a developer creating a file download script to write script against this entire process, it is helpful to understand the steps and interactions with the Ginnie Mae servers.
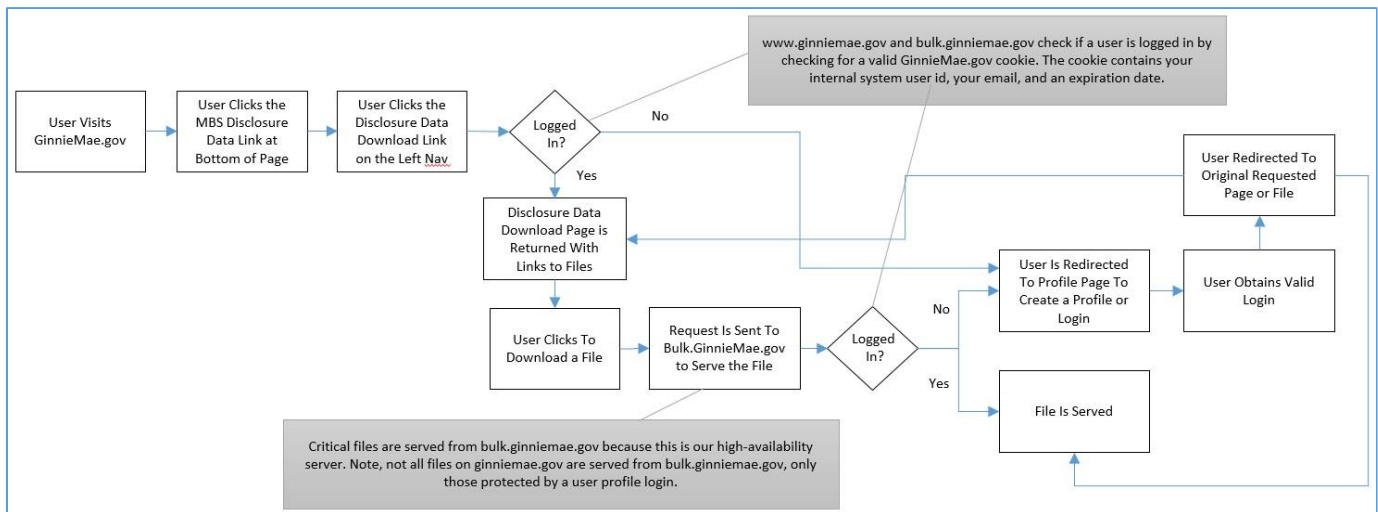

*Figure 1: GinnieMae.gov File Download Process*

Download Solution Approach

To download files from GinnieMae.gov programmatically a developer should create a script that makes HTTP request(s) directly to www.ginniemae.gov or bulk.ginniemae.gov for the desired file(s). To determine the appropriate path to the file, right click on a link from the www.ginniemae.gov website and copy the shortcut or view its properties. This is the link you should use in your script. When you make your request all you need to do is to be sure that your request passes your ginniemae.gov cookie with an expiration date well into the future. **You do not need to script the login process**. By having a valid cookie passed with your HTTP request you will pass Ginnie Mae website logic that is reviewing requests for a valid cookie (or the 'Logged In?' checks in the Figure 1 process flow).

Automating File Download Instructions: Fall 2023

Obtaining Your Ginnie Mae User Profile Cookie

1. Go to https://www.ginniemae.gov

2. Click on the Disclosure Data Download link under the Disclosure –> Disclosure Data menu located at the top of the page.
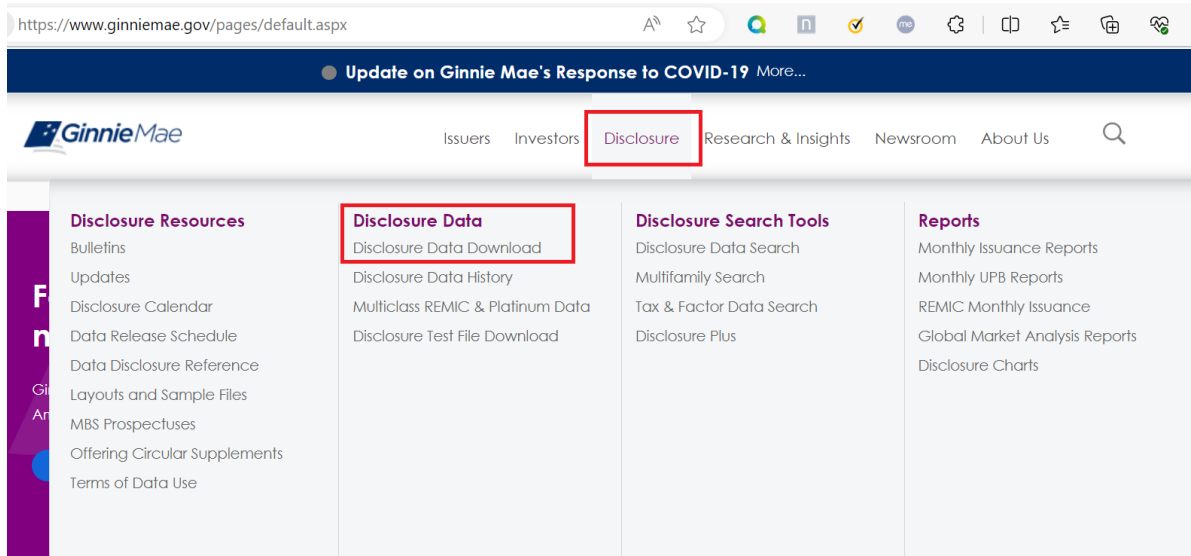
3. At this point, if you are already logged in, you will see the Disclosure Data Download Files page with a dropdown box that say who you are logged in as and can move on to step 5. If you're not logged in you will be redirected to login or create a new profile. If you have a profile already established, log in. If you don't, then create a profile. Once complete you should be at the Disclosure Data Download Files page with the dropdown displaying that you're logged in and can move on to step 5.
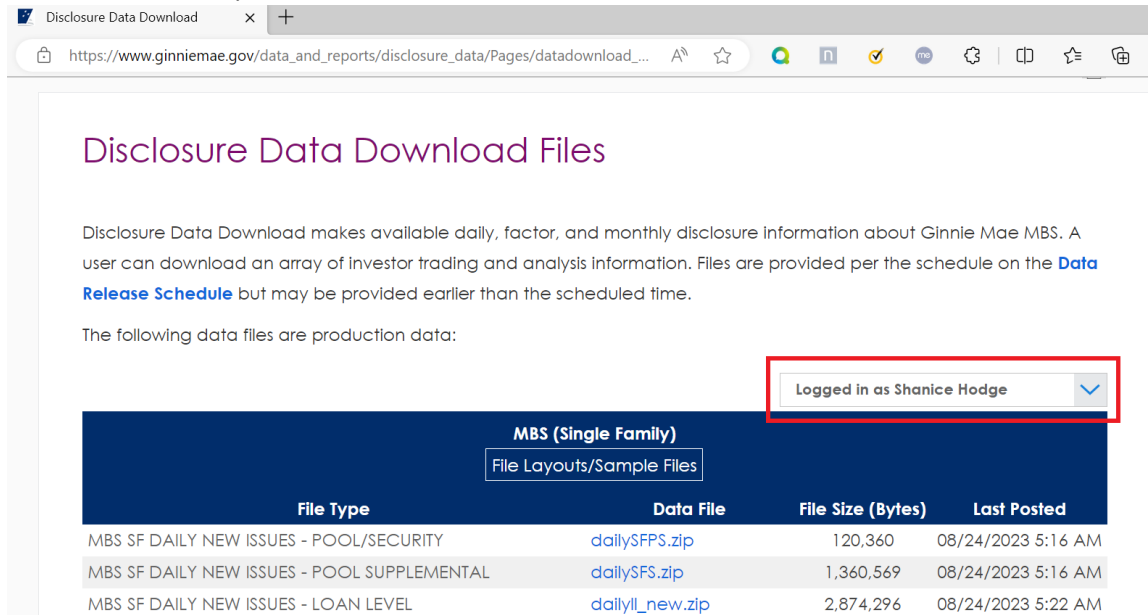
4. Now you need to retrieve your Ginnie Mae website issued cookie. You can do this by using the IE developer tools and/or Fiddler to obtain your cookie information. Run one of these tools to capture the HTTP request made when you refresh the Disclosure Data Download Files page once you are logged in. You should see something like the following:
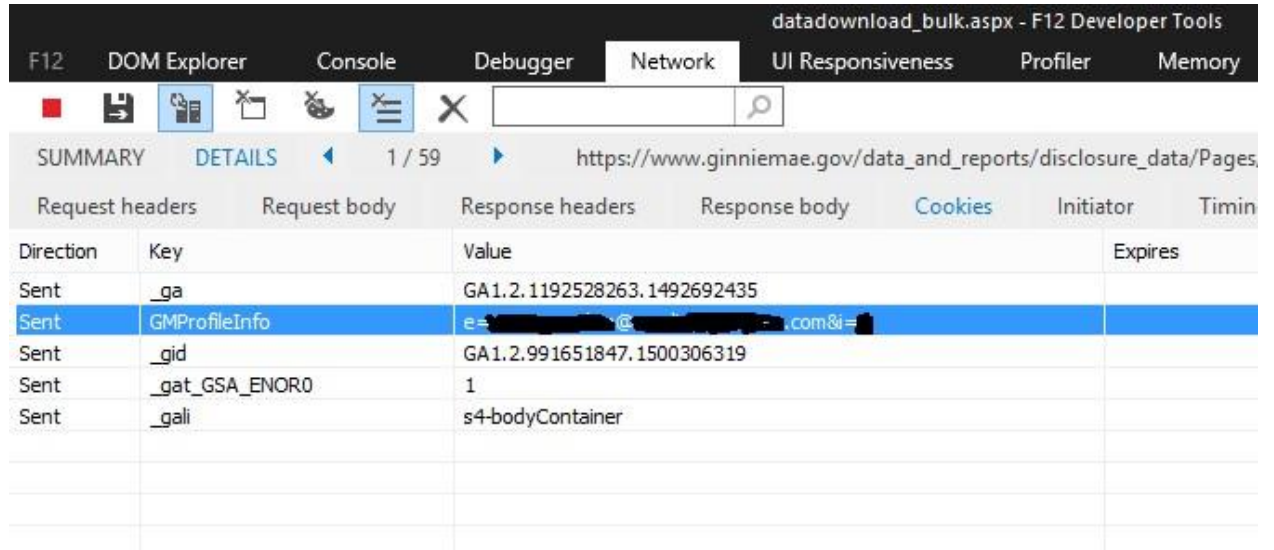
5. The cookie of interest is called GMProfileInfo and has the following format: e=myemail@domain.com&i=#### . The 'e' attribute represents your email addressed associated with your disclosure account and the 'i' attribute represents your system ID.
You will need to reproduce this cookie information programmatically when creating your HTTP Request using your scripting/code method of choice.

Creating Your GinnieMae.gov File Download Script

Now you have the information you need to create your HTTP request(s) within your script/code. In general, first your code will need to create a valid GinnieMae.gov cookie. Then it should attach/include the cookie with the HTTP requests that you send to GinnieMae.gov to download files. Finally, it should save the responses back to your computer/server as files.

The cookie is a multi-value cookie containing two keys as you can see from step 6 above. When constructing your cookie it will need to have the following elements:

a) Name: GMProfileInfo

b) Values:

Key	Value	From Step 6 Sample
e	Your User Profile Email Obtained From Your Cookie	e=myemail@domain.com

i	Your User Profile Id Obtained From Your Cookie	i=####

c) Domain: ginniemae.gov

d) Expiration: Date in the future

Cookies issued by the Ginnie Mae site typically expire in 90 days. When creating your cookie object in your code you'll want to be sure to set your cookie expiration to a date well into the future. If your cookie doesn't have an expiration date in the future you will be redirected to the user profile login page when attempting to download a file.

C# .NET Example of Setting the Required HTTP Headers

First, you must add the Ginnie Mae cookie to the headers. Below is sample code for creating the cookie and adding it to your request header in C# .NET. Begin by creating the cookie using the GMProfileInfo name. Then add the two key/value pairs for e and i. Then add the cookie expiration date, followed by the domain.

```
client.Headers.Add(HttpRequestHeader.Cookie,
@"GMProfileInfo=e=youremail@yourdomain.com&i=######;expires=" +
DateTime.Today.AddYears(1).ToUniversalTime().ToString("r") + ";domain=ginniemae.gov;path='\'");
```

Additionally, in the headers, you must pass a "user-agent" value, as below:

```
client.Headers.Add("user-agent", "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/99.0.4844.74 Safari/537.36 Edg/99.0.1150.46");
```

Guide: Procedure for Automating Downloads of Ginnie Mae Data Files

C# .NET Sample Console Application Downloading Disclosure File

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Web;
using System.Net;
using System.IO;

namespace FileDownload
{
    class Program
    {
        static void Main(string[] args)
        {
            FileStream ostrm;
            StreamWriter writer;
            Console.WriteLine("Enter a bulk file path to download:");
            string file = Console.ReadLine();
            Console.WriteLine("Enter where to save the file:");
            string saveto = Console.ReadLine();

            byte[] response ;
            if (file != "" && saveto != "")
            {
                using (CookieAwareWebClient client = new CookieAwareWebClient())
                {

                    client.Headers.Add(HttpRequestHeader.Cookie, @"GMProfileInfo=e=          &i=    ;expires=" + DateTime.Today.AddYears(1).ToUniversalTime().ToString("r") + ";domain=ginniemae.gov;path='\'");

                    client.Headers.Add("user-agent", "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.74 Safari/537.36 Edg/99.0.1150.46");

                    response = client.DownloadData(file);


                }
                Uri u = new Uri(file);

                string su = saveto + file.Split('/').Last();
                System.IO.File.WriteAllBytes(su, response);

                Console.WriteLine("File written to : " + su.ToString());
                Console.ReadLine();


            }
        }
    }

    public class CookieAwareWebClient : WebClient
    {


    //  private CookieContainer m_container = new CookieContainer();

        protected override WebRequest GetWebRequest(Uri address)
        {



            WebRequest request = base.GetWebRequest(address);
            if (request is HttpWebRequest)
            {
    //    (request as HttpWebRequest).CookieContainer = m_container;

            }
            return request;
        }
    }
}
```
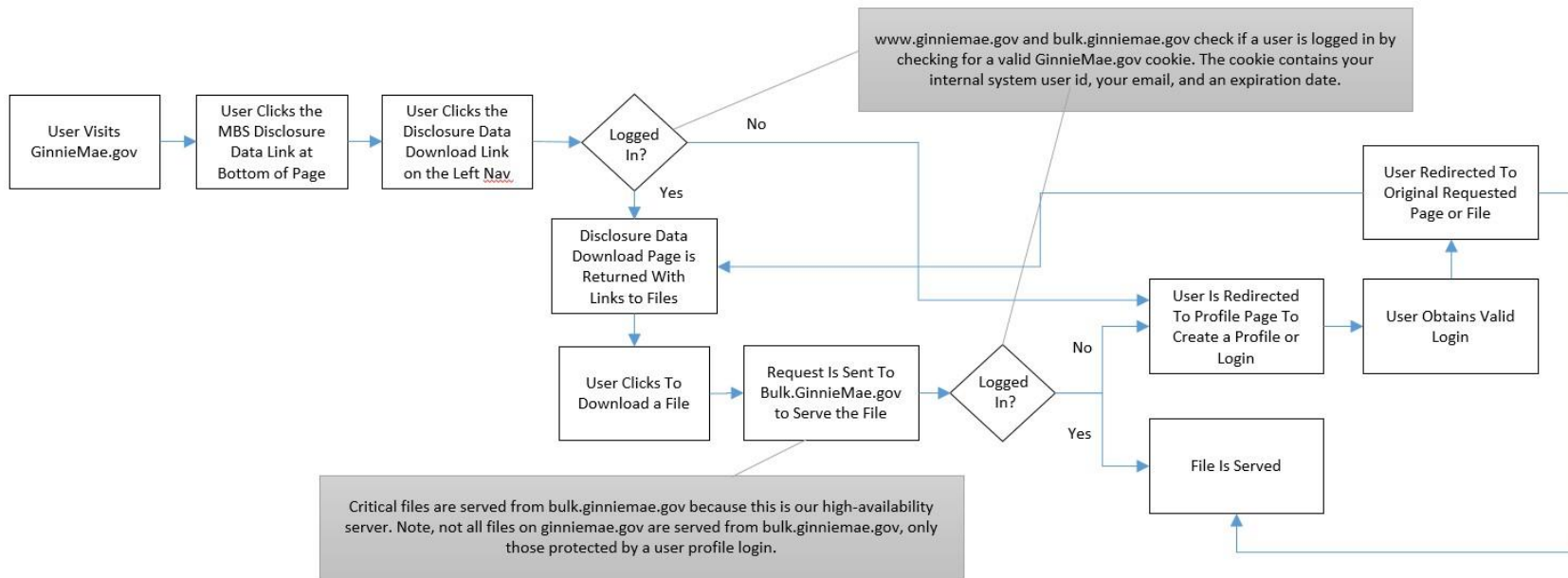
Guide: Procedure for Automating Downloads of Ginnie Mae Data Files

Appendix 1 : GinnieMae.gov File Download Process Diagram

www.ginniemae.gov and bulk.ginniemae.gov check if a user is logged in by checking for a valid GinnieMae.gov cookie. The cookie contains your internal system user id, your email, and an expiration date.

User Visits GinnieMae.gov → User Clicks the MBS Disclosure Data Link at Bottom of Page → User Clicks the Disclosure Data Download Link on the Left Nav → Logged In?

No

Yes

Disclosure Data Download Page is Returned With Links to Files

User Clicks To Download a File → Request Is Sent To Bulk.GinnieMae.gov to Serve the File → Logged In?

No → User Is Redirected To Profile Page To Create a Profile or Login → User Obtains Valid Login → User Redirected To Original Requested Page or File

Yes → File Is Served

Critical files are served from bulk.ginniemae.gov because this is our high-availability server. Note, not all files on ginniemae.gov are served from bulk.ginniemae.gov, only those protected by a user profile login.

Guide: Procedure for Automating Downloads of Ginnie Mae Data Files

Appendix 2: .NET C# Sample Code

```csharp
System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Web;
using System.Net;
using System.IO;

namespace FileDownload
{
    class Program
    {
        static void Main(string[] args)
        {
            FileStream ostrm;
            StreamWriter writer;
            Console.WriteLine("Enter a bulk file path to download:");
            string file = Console.ReadLine();
            Console.WriteLine("Enter where to save the file:");
            string saveto = Console.ReadLine();

            byte[] response ;
            if (file != "" && saveto != "")
            {
                using (CookieAwareWebClient client = new CookieAwareWebClient())
                {

                    client.Headers.Add(HttpRequestHeader.Cookie,
@"GMProfileInfo=e=youremail@yourdomain.com&i=99999;expires=" +
DateTime.Today.AddYears(1).ToUniversalTime().ToString("r") +
";domain=ginniemae.gov;path='\'");    // DEV

                    client.Headers.Add("user-agent", "Mozilla/5.0 (Windows NT 10.0; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.74 Safari/537.36 Edg/99.0.1150.46");

                    response = client.DownloadData(file);

                }
                Uri u = new Uri(file);

                string su = saveto + file.Split('/').Last();
                System.IO.File.WriteAllBytes(su, response);

                Console.WriteLine("File written to : " + su.ToString());
                Console.ReadLine();


            }
        }
    }

public class CookieAwareWebClient : WebClient
    {
```

```csharp
//   private CookieContainer m_container = new CookieContainer();

    protected override WebRequest GetWebRequest(Uri address)
    {

        WebRequest request = base.GetWebRequest(address);
        if (request is HttpWebRequest)
        {
         //   (request as HttpWebRequest).CookieContainer = m_container;

        }
        return request;
    }
}

}
```