EPF REST Web Services

Last Updated: 8-24-2016

Recent Changes:

There are now two paths to the EPF web services, the original path to the Akamai network of servers and now a direct path to the USPS servers. This new base path to the USPS servers was added due to issues encountered downloading large files recently with Akamai and gives more options when issues are encountered. With the web services, the first part of the URL is the **base path** that points to either Akamai or USPS servers; the second part of the URL is the actual call to the web service functions which are described further down.

USPS server URLs

https://epfeaup.usps.gov/up/epfupld/**epf/version**	**GET**	
https://epfeaup.usps.gov/up/epfupld/**epf/login**	**POST**	
https://epfeaup.usps.gov/up/epfupld/**epf/logout**	**POST**	
https://epfeaup.usps.gov/up/epfupld/**download/status**	**POST**	
https://epfeaup.usps.gov/up/epfupld/**download/acslist**	**POST**	
https://epfeaup.usps.gov/up/epfupld/**download/dnldlist**	**POST**	
https://epfeaup.usps.gov/up/epfupld/**download/list**	**POST**	
https://epfeaup.usps.gov/up/epfupld/**download/listplus**	**POST**	
https://epfeaup.usps.gov/up/epfupld/**download/epf**	**POST**	**(download ALL files)**

AKAMAI server URLs

https://epfws.usps.gov/ws/resources/**epf/version**	**GET**	
https://epfws.usps.gov/ws/resources/**epf/login**	**POST**	
https://epfws.usps.gov/ws/resources/**epf/logout**	**POST**	
https://epfws.usps.gov/ws/resources/**download/status**	**POST**	
https://epfws.usps.gov/ws/resources/**download/acslist**	**POST**	
https://epfws.usps.gov/ws/resources/**download/dnldlist**	**POST**	
https://epfws.usps.gov/ws/resources/**download/list**	**POST**	
https://epfws.usps.gov/ws/resources/**download/listplus**	**POST**	
https://epfws.usps.gov/ws/resources/**download/epf**	**POST**	**(download ALL files)**
https://epfws.usps.gov/ws/resources/**download/file**	**POST**	**(download ALL files)**
https://epfws.usps.gov/ws/resources/**download/akamai**	**GET**	**(SPECIFIC FILES)**

Description:

The logon key and token key should be SAVED from each reply as they are refreshed on every request/response call made to the web server. **These web services must be called one at a time in serial order, as the security token refresh process will fail if calls are threaded and you do not pass the most recently refreshed security token in the next function call. Only one active session per user login is supported; multiple users cannot use the same login and expect to succeed as only one user will have the most current security token.** The security token is returned on each call within the JSON object and also in the response header. From a programming perspective, it may be better to program using the headers from the response object, as when the actual file is downloaded, no JSON object is returned. To make the request calls, a JSON object will need to be created and passed as a POST form parameter **obj={jsonObject}** for security reasons using content-type "**application/x-www-form-urlencoded"**.

The "version" function call allows users to test service availability through their firewalls using a browser or in code using a simple GET method with no parameters or JSON object. The version is returned as a JSON object. All items are considered case sensitive.

NOTE: the form parameter name must be "**obj**", as in **obj=**{json object to string}
The web services look for a parameter called obj and takes the string value and converts it back to a JSON object to retrieve the internal values.

Function Call Details:

<u>Version</u>:

The ***epf/version*** function call is a simple GET style call that can be used as a test call in a web browser. By entering the URL, a response object containing a JSON object should be returned immediately. This can also be used to test connectivity through firewalls and if this simple call does not work it usually indicates problems with connectivity to the Internet.

INPUT: Call the URL (GET) without any additional parameters and a response JSON object will be return or a server error if the services are not running.

OUTPUT: JSON object similar to example below.

```
{
   "response":"success",
   "messages":"Web service version and build date.",
   "version":"v1.05.3",
   "build":"2016-05-23"
}
```

<u>Login</u>:

The ***epf/login*** function call is used to log into EPF system and retrieve the required keys for all future web service calls. As with most systems, one should program for sessions timing out and passwords expiring.

INPUT: Call the URL via POST method with JSON object serialized and set parameter.

```
obj={
        "login":"my_email_login@company.com",
        "pword":"my_password"
}
```

OUTPUT: JSON object similar to example below.

```
{
   "response":"success",
   "messages":"Login validation succeeded.",
   "logonkey":"123456",
   "tokenkey":"ASDJH7Y7Y76767S8DASJKNASKLDJIUY"
}
```

<u>Logout</u>:

The ***epf/logout*** function call is used to log out of EPF system and remove all security tokens from being active. This is not a required function call but is considered good practice and will deactivate your security token.

INPUT: Call the URL via POST method with JSON object serialized and set parameter.

```
obj={
        "logonkey":"123456",
        "tokenkey":""ASDJH7Y7Y76767S8DASJKNASKLDJIUY"
}
```

OUTPUT: JSON object similar to example below.

```
{
   "response":"success",
   "messages":" Logout process succeeded."
}
```

ACS Only List of Files:

The ***download/acslist*** function call is used to retrieve a list of all ACS files within scope for the supplied user logon key.

NOTE: **reccount** should be checked for records first before trying to loop through the **fileList** array.

INPUT: Call the URL via POST method with JSON object serialized and set parameter.

```
        obj={
                "logonkey":"123456",
                "tokenkey":"SAD4645KJDKJDK878934N2MNKLMK"
        }
```

OUTPUT: JSON object similar to example below.

```
        {
                "response":"success",
                "messages":"Process succeeded.",
                "logonkey":"123456",
                "tokenkey":"SAD4645KJDKJDK878934N2MNKLMK",
                "reccount":"1",
                "fileList":[
                        {
                          "productcode":"ACS",
                          "productid":"PARENT",
                          "fulfilled":"2016-04-11",
                          "status":"N",
                          "fileid":"12345",
                          "filepath":"/epf/pepf/a01shared/epfdata/acs/20160411/",
                          "filename":" P000000_000000411.zip",
                          "filesize":"12345"
                        }
                ]
        }
```

<u>All Available Files</u>:

The ***download/dnldlist*** function call is used to retrieve a list of all EPF files within scope for the supplied user logon key.

NOTE: **reccount** should be checked for records first before trying to loop through the **dnldfileList** array.

INPUT: Call the URL via POST method with JSON object serialized and set parameter.

```
obj={
        "logonkey":"123456",
        "tokenkey":"SAD4645KJDKJDK878934N2MNKLMK"
}
```

OUTPUT: JSON object similar to example below.

```
{
        "response":"success",
        "messages":"Process succeeded.",
        "logonkey":"123456",
        "tokenkey":"SAD4645KJDKJDK878934N2MNKLMK",
        "reccount":"1",
        "dnldfileList":[
                {
                  "productcode":"AIS",
                  "productid":"CR215N",
                  "fulfilled":"2014-08-15",
                  "status":"N",
                  "fileid":"12345",
                  "filepath":"/epf/pepf/a01shared/epfdata/ais/20140815/",
                  "filename":" crisnatl.zip",
                  "filesize":"12345"
                }
        ]
}
```

<u>List</u>:

The ***download/list*** function call is used to retrieve a list of actively available files for the supplied user login, product code and product id. Additional parameters can be optionally supplied to filter the list even more.

NOTE: Added the ability to filter by more than one status code … you can now pass a string of status codes. “SNX” would bring back all 3 statuses.

ALSO: **reccount** should be checked for records first before trying to loop through the **fileList** array.

INPUT: Call the URL via POST method with JSON object serialized and set parameter.

```
obj={
        "logonkey":"123456",
        "tokenkey":"SAD4645KJDKJDK878934N2MNKLMK",
        "productcode":"NCAM",
        "productid":"RDI",
        "status":"N",                    (optional)
        "fulfilled":"20160411"           (optional)
}
```

OUTPUT: JSON object similar to example below.

```
{
        "response":"success",
        "messages":"Process succeeded.",
        "logonkey":"123456",
        "tokenkey":"SAD4645KJDKJDK878934N2MNKLMK",
        "reccount":"2",
        "fileList":[
                {
                   "fileid":"12345",
                   "status":"N",
                   "filepath":"/epfdata/ncam/20120815/rdi.tar",
                   "fulfilled":"2012-08-15"
                },
                {
                   "fileid":"23456",
                   "status":"N",
                   "filepath":"/epfdata/ncam/20120915/rdi.tar",
                   "fulfilled":"2012-09-15"
                }
        ]
}
```

<u>List Plus</u>:

The ***download/listplus*** function call is used to retrieve a list of actively available files for the supplied user login, product code and product id. Additional parameters can be optionally supplied to filter the list even more.

NOTE: Added the ability to filter by more than one status code … you can now pass a string of status codes. “SNX” would bring back all 3 statuses.

ALSO: **reccount** should be checked for records first before trying to loop through the **fileList** array.

INPUT: Call the URL via POST method with JSON object serialized and set parameter.

```
        obj={
                "logonkey":"123456",
                "tokenkey":"SAD4645KJDKJDK878934N2MNKLMK",
                "productcode":"NCAM",
                "productid":"RDI",
                "status":"N",                    (optional)
                "fulfilled":"20160411"           (optional)
        }
```

OUTPUT: JSON object similar to example below.

```
        {
                "response":"success",
                "messages":"Process succeeded.",
                "logonkey":"123456",
                "tokenkey":"SAD4645KJDKJDK878934N2MNKLMK",
                "reccount":"1",
                "fileList":[
                        {
                          "fileid":"12345",
                          "status":"N",
                          "fulfilled":"2014-08-15",
                          "filepath":"":"/epfdata/ncam/20140815",
                          "filename":"rdi.tar",
                          "filesize":"1234"
                        }
                ]
        }
```

<u>Status</u>:

The **download/status** function call is used to change the file status; either before being downloaded (set to “S” for started) and / or after a successful download (set to “C” for completed).

N = new; S = download started; X = download cancelled; C = download completed.

INPUT: Call the URL via POST method with JSON object serialized and set parameter.

```
        obj={
                "logonkey":"123456",
                "tokenkey":"ASDJH7Y7Y76767S8DASJKNASKLDJIUY",
                "newstatus":"C",
                "fileid":"654321"
        }
```

OUTPUT: JSON object similar to example below.

```
        {
                "response":"success",
                "messages":"Update process succeeded.",
                "logonkey":"123456",
                "tokenkey":"ASDJH7Y7Y76767S8DASJKNASKLDJIUY"
        }
```

<u>USPS & AKAMAI File downloads</u>:

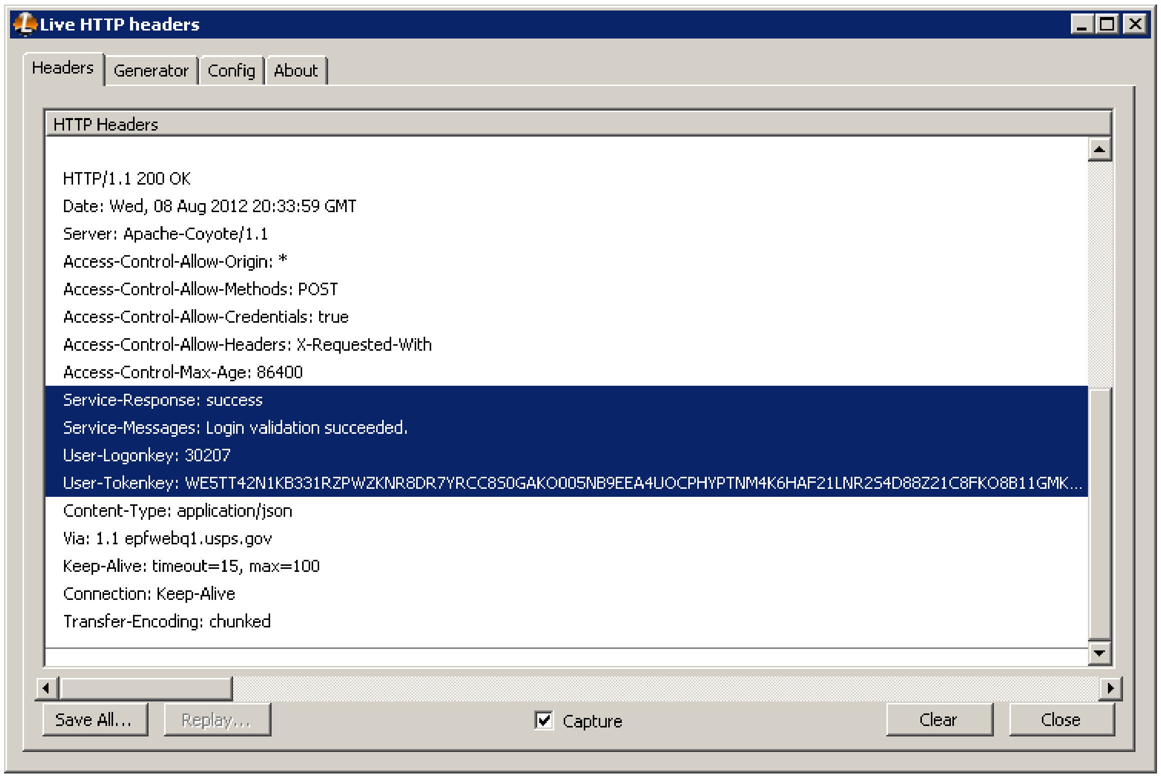
The **download/epf** function call is used to download the files from the USPS or AKAMAI servers. Request headers are no longer required, only the file id is required and if your login has rights to that file it will be returned.

INPUT: Call the URL via POST method with JSON object serialized and set parameter.

```
obj={
        "logonkey":"123456",
        "tokenkey":"ASDJH7Y7Y76767S8DASJKNASKLDJIUY",
        "fileid":"654321"
}
```

OUTPUT: File is returned as the response object. The logon key and token key have to be extracted from the response headers as the file is returned instead of a JSON object.

Response Header **User-Logonkey** = 123456
Response Header **User-Tokenkey** = ASDJH7Y7Y76767S8DASJKNASKLDJIUY



```
HTTP/1.1 200 OK
Date: Wed, 08 Aug 2012 20:33:59 GMT
Server: Apache-Coyote/1.1
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: POST
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: X-Requested-With
Access-Control-Max-Age: 86400
Service-Response: success
Service-Messages: Login validation succeeded.
User-Logonkey: 30207
User-Tokenkey: WE5TT42N1KB331RZPWZKNR8DR7YRCC8S0GAKO005NB9EEA4UOCPHYPTNM4K6HAF21LNR2S4D88Z21C8FKO8B11GMK...
Content-Type: application/json
Via: 1.1 epfwebq1.usps.gov
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
```

AKAMAI File downloads:

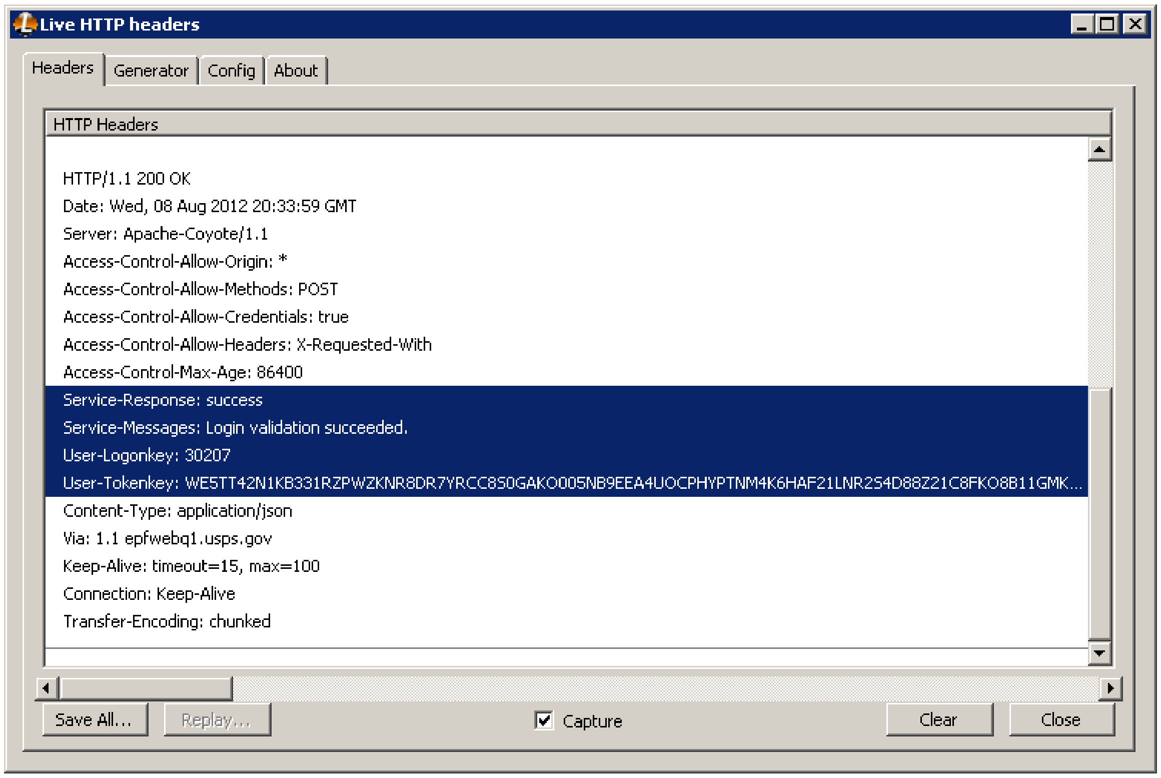
The **download/file** function call is used to download the files ONLY FROM AKAMAI servers. Request headers are no longer required, only the file id is required and if your login has rights to that file it will be returned. Previous specifications called for both Request Headers and a file path, these are now ignored if presented so as to be compatible with older code.

INPUT: Call the URL via POST method with JSON object serialized and set parameter.

```
obj={
        "logonkey":"123456",
        "tokenkey":"ASDJH7Y7Y76767S8DASJKNASKLDJIUY",
        "fileid":"654321"
}
```

OUTPUT: File is returned as the response object. The logon key and token key have to be extracted from the response headers as the file is returned instead of a JSON object.

Response Header **User-Logonkey** = 123456
Response Header **User-Tokenkey** = ASDJH7Y7Y76767S8DASJKNASKLDJIUY



```
HTTP/1.1 200 OK
Date: Wed, 08 Aug 2012 20:33:59 GMT
Server: Apache-Coyote/1.1
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: POST
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: X-Requested-With
Access-Control-Max-Age: 86400
Service-Response: success
Service-Messages: Login validation succeeded.
User-Logonkey: 30207
User-Tokenkey: WE5TT42N1KB331RZPWZKNR8DR7YRCC8S0GAKO005NB9EEA4UOCPHYPTNM4K6HAF21LNR2S4D88Z21C8FKO8B11GMK...
Content-Type: application/json
Via: 1.1 epfwebq1.usps.gov
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
```

The **download/akamai** function call is NEW for downloading specific files ONLY FROM AKAMAI servers. Four Request headers are REQUIRED, as they are intercepted by Akamai to help in determining the correct file to return. The JSON object is NOT REQUIRED and should not be sent with a GET call, as GET calls are URL based with no body content.

REQUEST HEADERS:

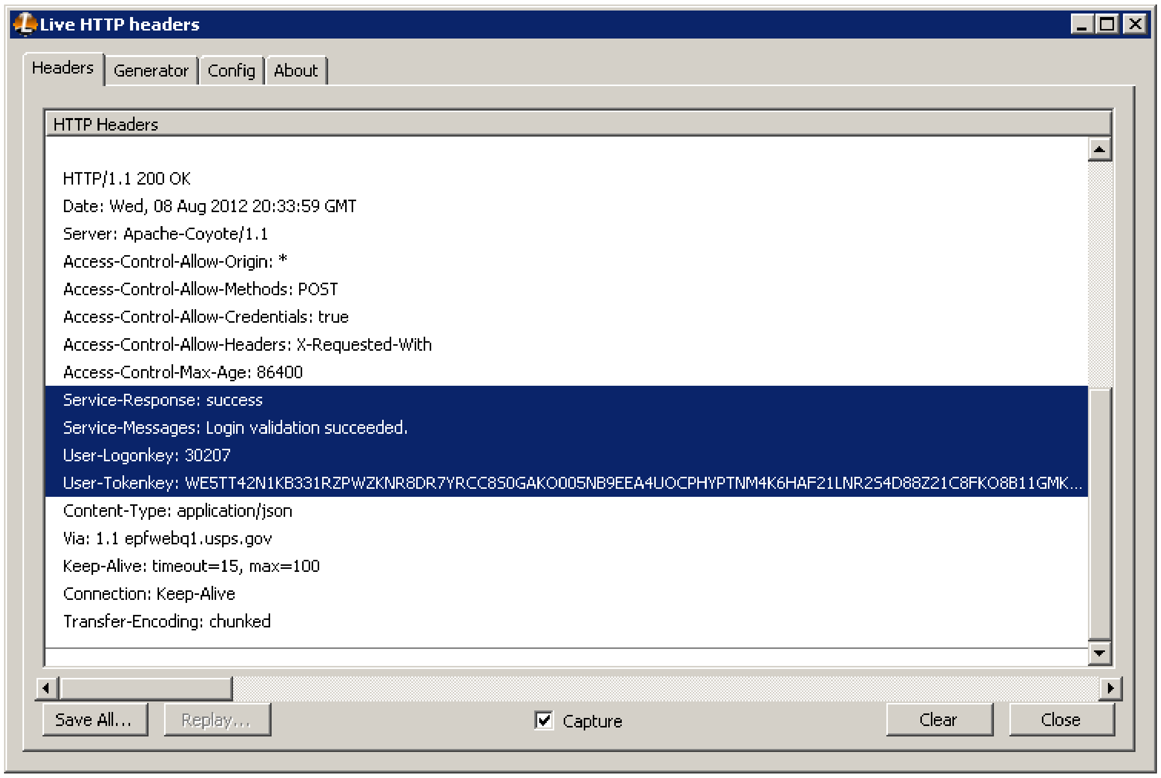
Akamai-File-Request	(file path including the file name)
logonkey	(epf user logon key)
tokenkey	(security token key)
fileid	(file id from the listings)

Java Example:

```
connection.setRequestProperty("Akamai-File-Request", view.getFilePath());
connection.setRequestProperty("logonkey", view.getLogonKey());
connection.setRequestProperty("tokenkey", view.getTokenKey());
connection.setRequestProperty("fileid", view.getFileID());
```

OUTPUT: File is returned by Akamai after validation by USPS. The logon key and token key have to be extracted from the response headers as the file is returned instead of a JSON object.

Response Header **User-Logonkey** = 123456
Response Header **User-Tokenkey** = ASDJH7Y7Y76767S8DASJKNASKLDJIUY



```
HTTP/1.1 200 OK
Date: Wed, 08 Aug 2012 20:33:59 GMT
Server: Apache-Coyote/1.1
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: POST
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: X-Requested-With
Access-Control-Max-Age: 86400
Service-Response: success
Service-Messages: Login validation succeeded.
User-Logonkey: 30207
User-Tokenkey: WE5TT42N1KB331RZPWZKNR8DR7YRCC850GAKO005NB9EEA4UOCPHYPTNM4K6HAF21LNR254D88Z21C8FKO8B11GMK...
Content-Type: application/json
Via: 1.1 epfwebq1.usps.gov
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
```

JSON Object Field Descriptions (may be case sensitive):

- **logonkey** – user login key returned from the login call. This key will not change, but must be used in every call for security validation.
- **tokenkey** – security token returned from the login call. This key must be used in every function call with a newly refreshed security token being return to use in your next call. Threading should not be used, as the security token could get out of synch, only serial calls are supported.
- **productcode** – This is the product code assigned to products. You will need to know your product codes and only one product code can be requested per list call.
- **productid** – This is a sub-key of the product code. You will need to know your product ID and only one product code + product ID can be requested per list call. This basically acts as a filter to limit the size of the list being returned.
- **status** – (optional) the current download status of the listed file. This can be used as an additional filter with most requesters being interested in only newly released files. After the file is downloaded, status should be set to complete.
- **fulfilled** – (optional) the fulfillment date can be used as an additional filter. Date format is 'CCYY-MM-DD' and will be ignored if passed in an incorrect format.

File Statuses: (optional - leave out for ALL to be returned)

- "N" = new file available
- "S" = download started
- "X" = download canceled
- "C" = download completed successfully