LANDFIRE’s Download Web Service User’s Guide

LANDFIRE provides a set of web services to the developer that can be incorporated into custom applications.

- **Request Validation Web Service** verifies and validates the dataset information and a user-defined area of interest and provides a fully parameterized URL(s) that can be used by the Download Service.
- **Download Web Service** initiates a request for products, queries the system to obtain a job status, and returns the requested products to the user.

This document discusses how to use the Download Service web service after the parameterized URL(s) are received from the Request Validation Web Service.

Download Web Service

The Download Web Service is used to make a request for full resolution products from the LANDFIRE Data Distribution Site (DDS). The DDS performs real-time clipping of data.

The URL to the Download Service Web Service Description Language (WSDL) page is:

https://landfire.cr.usgs.gov/axis2/services/DownloadService?wsdl

There are four methods that must be called in the correct sequence:

- initiateDownload( ) – used to submit a request
- getDownloadStatus( ) – used to check the current status of the request
- getData( ) – used to obtain the finished download bundle from LANDFIRE web servers
- setDownloadComplete( ) – used to force immediate cleanup of old requests

initiateDownload( ) Method

In the wsdl file there are over 40 potential parameters that can be passed to this method. Therefore, it’s best to use the Request Validation Service to compose this URL because the URL obtained from the Request Validation Service is guaranteed to correctly pass the initiateDownload requirements. This method call will submit the request to the job queue.

The response returned from the initiateDownload method call will include a unique id to be used as input for all successive calls to the Download Service. If there is an error when using this method, a string containing the word “error” will be returned. If successful, the response will look something like this:

- <ns:initiateDownloadResponse>
- <ns:return>**20090608.150009516.152061160045**</ns:return>
- </ns:initiateDownloadResponse>

The download request is assigned a priority value based on the number of requests in the user’s job queue. If there is only one request in the queue, it will always be assigned the highest or “best” priority. All jobs of the same priority are worked via a “First In, First Out” basis. If there is already one unfinished job in the queue, then the second job is assigned a priority of one less. The third unfinished job is assigned a priority of two less than the “best” priority. This is so that one single user cannot use all the resources when others are also contending for those resources. Therefore, writing the application so that only a certain number of jobs are in the queue at any time or monitoring the time it takes to work off the jobs and submit new jobs accordingly, might be helpful.

getDownloadStatus( ) Method

This method obtains the current status of a request. LANDFIRE products are stored on a variety of systems in a variety of different formats. It will take a short amount of time to find the requested product(s), reformat (if necessary), package (with metadata), and move the finished download bundle to a LANDFIRE web server where it is then available to download for one hour. After one hour, the download bundle is automatically deleted from the web server. The job identifier that was returned from initiateDownload method call must be used.

https://landfire.cr.usgs.gov/axis2/services/DownloadService/getDownloadStatus?downloadID=20090608.150009516.152061160045

To avoid unnecessary status queries slowing down the servers, we recommend calling this method no more often than every 30 seconds. As the request passes through different stages of processing, the status code and text message will indicate the current progress.

<ns:return>210,Extracting Data</ns:return>

If an error has occurred during one of the processing stages, a string containing the text “error” will be received.

This method checks the status field from the job record that was submitted in the initiateDownload method call. One hour after the job record was first submitted, it is automatically deleted from the database, even if the job has not finished processing. Therefore, staggering requests and/or submitting jobs during off-hours could prove beneficial. If a job has returned an error, try it again. Repeated errors to the same product could indicate a problem with the dataset at EROS and LANDFIRE Customer Services should be notified so the issue can be investigated.

getData( ) Method

Once a status of 400 or greater is received, the finished download bundle has been placed on a LANDFIRE web server and is ready to be downloaded. The download bundle can be retrieved using the getData method call and the job identifier:

https://landfire.cr.usgs.gov/axis2/services/DownloadService/getData?downloadID=20090608.150009516.152061160045

Note that the finished download bundle will be available on the web server for one hour, after which an automated process will delete the download bundle to make room for other users. If an application misses the one-hour window, the request must be re-submitted starting at the initiateDownload step.

Depending on what programming language is being used, an application will receive the product(s) as a series of bytes through some kind of connection object. If the request is for a zip, the bytes will need to be written to a file called xxxxxxxx.zip. If the request is for a tar-gzip file as the bundle format, then the bytes will need to be written to a file called xxxxxxxx.tgz. The product(s) and all support files are contained within the .zip file or the .tgz file. At this point, the user is responsible for extracting the information needed from the download bundle.

setDownloadComplete( ) Method

Once the download bundle has finished downloading to the user’s computer, a download complete message needs to be sent back to the Download Service so that the job is properly logged and the cleanup process on our web servers can begin. This is done by using the setDownloadComplete method call and the job identifier:

https://landfire.cr.usgs.gov/axis2/services/DownloadService/setDownloadComplete?downloadID=20090608.150009516.152061160045

A successful response returned will look like:
<ns:return>**Successfully updated 20090608.150009516.152061160045 to Complete**</ns:return>

This method is a courtesy call to allow our system to begin immediate cleanup of the job. If this step is not fulfilled, the request will automatically be cleaned up one hour after the initiateDownload method call was submitted.

Code Examples for Calling the Download Web Service

Sample Python Code

```
#  submit a job to the Download Service
chunkUrl =
“https://landfire.cr.usgs.gov/axis2/services/DownloadService/initiateDownload?siz=
1&key=F4M&ras=1&pfm=ArcGRID_with_attribs&imsurl=-1&ms=-1&att=-1&lay=-1&fid=-
1&dlpre=lf&lft=-98.49394726686275&rgt=-
98.48708695676768&top=29.426617875735545&bot=29.420069397917526&wmd=1&mur=https://
landfire.cr.usgs.gov/distmeta/servlet/gov.usgs.edc.MetaBuilder&mcd=F4M&mdf=HTML&ar
c=ZIP&sde=LANDFIRE.US_110EVT&msd=LANDFIRE.US_110_SPATPLYGN_MASTER&zun=METERS&prj=1
02039&rsp=0&bnd=&bndnm=&csx=30.0&csy=30.0&ics=&ORIG=RVS”

try:
   page = urllib2.urlopen(chunkUrl)
except IOError, e:
   if hasattr(e, 'reason'):
        print 'We failed to reach a server.'
        print 'Reason: ', e.reason
   elif hasattr(e, 'code'):
        print 'The server couldn\'t fulfill the request.'
        print 'Error code: ', e.code
   else:
        result = page.read()
        print result
        # parse response for request id
        if result.find("VALID>false") > -1:
            # problem with initiateDownload request string
            # handle that here
        else:   #  downloadRequest successfully entered into queue
            startPos = result.find("<ns:return>") + 11
            endPos = result.find("</ns:return>")
            requestID = result[startPos:endPos]
            print requestID

# call Download service with request id to get status
downloadStatusUrl =
https://landfire.cr.usgs.gov/axis2/services/DownloadService/getStatus?downloadID=2
0090608.150009516.152061160045”
try:
     page2 = urllib2.urlopen(downloadStatusUrl)
except IOError, e:
    if hasattr(e, 'reason'):
        print 'We failed to reach a server.'
        print 'Reason: ', e.reason
    elif hasattr(e, 'code'):
        print 'The server couldn\'t fulfill the request.'
        print 'Error code: ', e.code
    else:
        result = page2.read()
```

```
        # remove carriage returns
        result = result.replace("&#xd;\n"," ")

        # parse out status code and status text
        startPos = result.find("<ns:return>") + 11
        endPos = result.find("</ns:return>")
        status = result[startPos:endPos]
        print status

#  once a status of 400 has been received this code will retrieve the file and
store it locally
        getFileUrl =
“https://landfire.cr.usgs.gov/axis2/services/DownloadService/getData?downloadID=20
090608.150009516.152061160045”
try:
   page = urllib2.urlopen(getFileUrl)
except IOError, e:
   if hasattr(e, 'reason'):
       print 'We failed to reach a server.'
       print 'Reason: ', e.reason
   elif hasattr(e, 'code'):
       print 'The server couldn\'t fulfill the request.'
       print 'Error code: ', e.code
   else:
      # store file in a local folder
      downloadFile = open('D:/localdownloads/20090608.150009516.152061160045.zip’,
'wb')
      while True:
          data = page.read(8192)
          if data == "":
              break
          downloadFile.write(data)
          downloadFile.close()
      finally:
          page.close()


#  send complete message back to server so it can cleanup the job
setStatusUrl =
“https://landfire.cr.usgs.gov/axis2/services/DownloadService/setDownloadComplete?d
ownloadID=20090608.150009516.152061160045”

try:
    page = urllib2.urlopen(setStatusUrl)
except IOError, e:
    if hasattr(e, 'reason'):
        print 'We failed to reach a server.'
        print 'Reason: ', e.reason
    elif hasattr(e, 'code'):
        print 'The server couldn\'t fulfill the request.'
        print 'Error code: ', e.code
    else:
        result = page.read()

        # remove carriage returns
        result = result.replace("&#xd;\n"," ")
```

```
        # parse out status code and status text
        startPos = result.find("<ns:return>") + 11
        endPos = result.find("</ns:return>")
        status = result[startPos:endPos]
        print status

finally:
    page.close()
```

Sample Java Code

```
#  use this sample block to call initiateDownload, getStatus or
setDownloadComplete

import java.net.*;

String completeURL =
https://landfire.cr.usgs.gov/axis2/services/DownloadService/initiateDownload?siz=1
&key=F4M&ras=1&pfm=ArcGRID_with_attribs&imsurl=-1&ms=-1&att=-1&lay=-1&fid=-
1&dlpre=lf&lft=-98.49394726686275&rgt=-
98.48708695676768&top=29.426617875735545&bot=29.420069397917526&wmd=1&mur=https://
landfire.cr.usgs.gov/distmeta/servlet/gov.usgs.edc.MetaBuilder&mcd=F4M&mdf=HTML&ar
c=ZIP&sde=LANDFIRE.US_110EVT&msd=LANDFIRE.US_110_SPATPLYGN_MASTER&zun=METERS&prj=1
02039&rsp=0&bnd=&bndnm=&csx=30.0&csy=30.0&ics=&ORIG=RVS”;

try {
    URL extURL = new URL(completeURL);
    URLConnection connection = extURL.openConnection();
    connection.setDoOutput(false);
    connection.setDoInput(true);
    connection.setConnectTimeout(10000);
    connection.setReadTimeout(10000);
    connection.connect();

    // Save the XML response as a string.
    InputStream in = connection.getInputStream();
    BufferedReader bin = new BufferedReader(new InputStreamReader(in));
    String ln;
    String tempString="";
    while ((ln = bin.readLine()) != null) {
        if (tempString == null) {
            tempString = ln;
        } else {
            tempString = tempString + ln;
        }
    }
    bin.close();

    // Parse response here
    System.out.println(tempString);

}
catch (SocketTimeoutException ste) {
    System.err.println(ste.getMessage());
```

```
}
catch(MalformedURLException mue) {
    System.err.println(mue.getMessage());
}
catch(IOException ioe) {
    System.err.println(ioe.getMessage());
}

//  use this sample block to retrieve the requested file and store it locally.
import javax.activation.DataHandler;
import java.net.*;

String completeURL =
"https://landfire.cr.usgs.gov/axis2/services/DownloadService/getData?downloadID=20
100222.132733625.152061165003";

try {
    URL extURL = new URL(completeURL);

    DataHandler dhSource = new DataHandler(extURL);
    InputStream is = dhSource.getInputStream();

   dhSource.writeTo(new
java.io.FileOutputStream("d:/download_folder/20100222.132733625.152061165003.zip))
;
  is.close();

}
catch (SocketTimeoutException ste) {
    System.err.println(ste.getMessage());
}
catch(MalformedURLException mue) {
    System.err.println(mue.getMessage());
}
catch(IOException ioe) {
    System.err.println(ioe.getMessage());
}
```

Acronyms and Abbreviations

Acronym	Definition
WMS	Web Map Service
URI	Uniform Resource Identifier
REST	Representational State Transfer
WSDL	Web Service Description Language