Developing Algorithms for the HTRC Framework

The HTRC Analytics Gateway comes with a set of 4 algorithms created by members of the HTRC software development team. Allowing HTRC users to submit their own algorithms that can be run on the HTRC Analytics Gateway will provide a significant enhancement to HTRC services. User-contributed algorithms will be vetted by HTRC before they are made part of the HTRC Analytics Gateway. Here we describe the software components that HTRC needs for every user-contributed algorithm. We also describe the constraints to which these software components have to adhere to work with the job submission module of the HTRC Analytics Gateway, and the runtime environment in which jobs are executed.

Runtime Environment

Algorithms submitted through the HTRC Analytics Gateway are launched by the job submission module as **batch jobs** on HPC systems. At the time of the writing of this document, jobs are run on the Karst computing cluster. Nodes of Karst are IBM NeXtScale nx360 M4 servers, each equipped with two Intel Xeon E5-2650 v2 8-core processors. All nodes run Red Hat Enterprise Linux (RHEL) 6. An HTRC job is assigned up to 8 processors on a single node, and around 10 GB of memory. Algorithms should be parallelized to take advantage of the assigned processing capacity.

All 9 algorithms on the HTRC Analytics Gateway are Java applications. However, the HPC systems on which HTRC algorithms run do provide other software such as Python, and R. See here for a complete list of modules available on Karst.

The specific HPC system used to execute HTRC jobs is subject to change.

Software Components

For an algorithm that is intended to be run through the HTRC Analytics Gateway, there are 3 required software components.

1. the algorithm executable, and any other runtime dependencies
2. a shell script wrapper that invokes the executable
3. an XML file that describes the algorithm

Items 1 and 2 are used in the development and testing phase. Once the algorithm source is finalized, an XML file describing the algorithm is created, and all 3 items submitted to HTRC for incorporation into the HTRC Analytics Gateway. In other words, the XML file is not needed for algorithm development. It is simply a means for the algorithm developer to communicate details about the algorithm to different HTRC services, such as the HTRC Analytics Gateway and the job submission module.

We expect that users will perform algorithm development and testing on Data Capsule VMs.

Algorithm Executable and Dependencies

It should be possible to run the algorithm using the provided algorithm executable and dependencies in the job runtime environment. Examples of executables include jar files, Python scripts, and R scripts. Dependencies may include additional libraries used by the executable at runtime.

As an example, consider an algorithm, Marc_Downloader, that downloads MARC records for an input workset, The algorithm executable is the jar file obtained by compiling the Java source along with its dependencies.

The job submission module of the HTRC Analytics Gateway plays a vital role in algorithm execution. The algorithm has to be built in such a way as to work with the environment provided by the job submission module, and to communicate correctly with the module (e.g., obtain input parameters from the module, and point the module to job results). Below, we describe aspects of the job submission module that are pertinent to the algorithm.

> (a) When a user runs an algorithm on the HTRC Analytics Gateway, the job submission module first creates a new job directory corresponding to the new run of the algorithm. We refer to this directory as the job's working directory. The job submission module positions the algorithm in the job working directory before launching the algorithm.
>
> (b) The job submission module provides the user-supplied values of input parameters to the algorithm through a property file that is placed in the job working directory. Property files contain key-value pairs that map variables to their values. Examples of property files are included later in this document.
>
> (c) Worksets are specified as input parameters to algorithms on the HTRC Analytics Gateway using names of the form worksetName@author, e.g., Edwin_Drood1870@imbeths. The job submission module obtains the volume ids, along with any other properties that might be defined for the volumes in the workset, that comprise each input workset and places them in files named after the worksets (i.e., worksetName@author) in the job's working directory. The algorithm reads these files to get the lists of volume ids in the worksets.
>
> Two example files representing worksets with and without additional fields are shown below. The algorithm should be designed to **handle both types of workset files**. The first file contains the list of volume ids in the workset Edwin_Drood1870@imbeths. The first line is a header line describing the contents of the file. Every subsequent line contains just one volume id.
>
> The second example is a workset file in CSV form, corresponding to a workset which has additional properties defined for each volume. The first line is a header line describing the contents of the file. Each subsequent line has a leading volume id, followed by other fields, which may be empty.

File containing volume ids in workset Edwin_Drood1870@imbeths

```
volume_id
hvd.hwjlz9
wu.89073545923
njp.32101065271965
mdp.39015055276359
hvd.hn1irz
hvd.32044090305574
mdp.39015019076705
mdp.39015031229050
uc2.ark:/13960/t3pv6fz90
uc2.ark:/13960/t0bv7d73q
```

File containing volume ids and other fields in workset WorksetWithProperties@props

```
volume_id,title,author,date,rights,OCLC,LCCN,ISBN,catalog_url,handle_url
miua.4925052.1928.001,"The public papers and addresses of Franklin D. Roosevelt. Volume one, The genesis of the New Deal, 1928-1932 with a special introduction and explanatory notes by President Roosevelt. ","Roosevelt, Franklin D. 1882-1945.",1938-00-00,5,,,,,
miua.4925383.1934.001,"The public papers and addresses of Franklin D. Roosevelt. Volume three, The advance of recovery and reform, 1934 with a special introduction and explanatory notes by President Roosevelt. ","Roosevelt, Franklin D. 1882-1945.",1938-00-00,5,,,,,
mdp.49015002221860,Public papers of the presidents of the United States. 1930,United States. President.,1930-00-00,1,1198154,58061050,,,
mdp.49015002221886,Public papers of the presidents of the United States. 1931,United States. President.,1931-00-00,1,1198154,58061050,,,
mdp.49015002221878,Public papers of the presidents of the United States. 1932/33,United States. President.,1933-00-00,1,1198154,58061050,,,
```

(d) The job submission module directs the standard error and standard output streams of the algorithm to files stderr.txt and stdout.txt in the job's working directory. These file names should not be reused in the algorithm code.

(e) The job submission module creates a new folder for the job results in the job working directory. All algorithm results are expected to be placed in this folder. The algorithm reads the name of the job result folder from the property file.

In accordance with item (b) in the list above, the algorithm should read values of its input parameters from a property file, as opposed to the command line, or other methods. Apart from input parameters, there are 3 values which, if used by the algorithm, should be read from the property file:

(i) the Data API URL
(ii) the JWT (JSON Web Token), used to access the Data API
(iii) the folder in which the result files are to be placed

If the algorithm accesses the Data API, then it should access the one specified in the property file. The JWT used by the algorithm to access the Data API should be the one specified in the property file. Finally, the algorithm should place all result files in the output folder specified in the property file. There are no restrictions on the names of keys in the property file used by the algorithm, e.g., dataApiURL and data_api_url are both valid key names for the Data API URL. Similarly, there is no restriction on the property file name. The job submission module obtains the mappings to key names and the property file name from the algorithm XML file described later.

Apart from input parameters and the 3 values described above, the algorithm may include any other configuration parameters also in the property file.

When an algorithm is launched through the HTRC Analytics Gateway, the job submission module creates the property file for the job, using definitions in the algorithm's XML file, and the input values submitted by the user through the Analytics Gateway. During algorithm development, it is up to the developer to create the property file from which the algorithm reads the values of its input parameters. Note that this also implies obtaining a recent, valid JWT to access the Data API.

The property file of a Marc_Downloader job is shown below as an example.

Properties file, collection.properties, of a Marc_Downloader job

```
collectionLocation = Dickens_as_Authors@skowalczyk
SolrProxy = http://chinkapin.pti.indiana.edu:9994/solr
outputDir = job_results
outputFile = result.zip
```

In this example, the value of collectionLocation is the input workset on which Marc_Downloader is run. Marc_Downloader has only one input field (the input workset). The algorithm developer may safely ignore the "job name" input parameter, seen in algorithm submission pages in the Analytics Gateway, since this is only meaningful in the context of the job submission framework on the HTRC Analytics Gateway.

outputDir is value (iii) described earlier. SolrProxy and outputFile are variables used by the algorithm which are not input parameters or one of the 3 special variables mentioned earlier. With the values as they are, the Marc_Downloader algorithm creates a job_results folder in the job's working directory, and places its results in result.zip in the job_results folder.

Marc_Downloader downloads MARC records from the Solr proxy. It does not access the Data API, and so neither the Data API URL nor the JWT are in the property file. Recall that the JWT is only used to access the Data API.

As another example, we include the property file of a run of Word_Count, an algorithm that obtains counts of the top n words in terms of frequency.

Properties file, wc.properties, of a Word_Count job

```
data.api.epr = https://silvermaple.pti.indiana.edu:25443/
volume.list.file = Early19thWestIndian@MPaolillo
auth.token = 688b996c113dc71e9d5e5c2421ae295a
volume.limit = 25
top.n = 100
concat = True
```

The property file of Word_Count contains values for the Data API URL, and the JWT needed to access the Data API. Apart from this it contains values for 3 input parameters: the input workset (volume.list.file), the no. of words to display (top.n), and whether the pages of a volume should be concatenated (concat). volume.limit is a configuration parameter used by Word_Count. The output folder is not specified because the output is placed in stdout.txt in the job's working directory. This is a special case. In general, algorithm developers should place their results in an output folder specified in the property file.

Algorithms should handle workset parameters in accordance with item (c) in the list of tasks performed by the job submission module included earlier. Input worksets are specified in the property file using names of the form worksetName@author. Examples include

```
collectionLocation = Dickens_as_Authors@skowalczyk
```

in the Marc_Downloader property file, and

```
volume.list.file = Early19thWestIndian@MPaolillo
```

in the Word_Count property file shown earlier.

The algorithm reads the list of volume ids that constitue an input workset from a file named after the workset in the job's working directory. In the Marc_Downloader example, the algorithm reads the volume ids from a file named Dickens_as_Authors@skowalczyk in the job working directory.

As in the case of the property file, it is up to the algorithm developer to create needed workset files during the algorithm development phase. Once the algorithm is on the HTRC Analytics Gateway, it is placed in the job working directory by the job submission module.

As per item (e) in the list of tasks performed by the job submission module included earlier, the algorithm should place any result files in the result folder specified in the property file. Further note that the result folder is created by the job submission module before the job is launched, and not by the algorithm. During algorithm development, the developer should make similar arrangements before running the algorithm.

Shell Script Wrapper

The algorithm developer should provide a shell script that invokes the algorithm executable with the required dependencies. Examples of simplified shell scripts wrappers are shown below. Details in these scripts that are only relevant to the HTRC Analytics Gateway's job submission module have been elided.

run_marc.sh, the shell script wrapper for Marc_Downloader

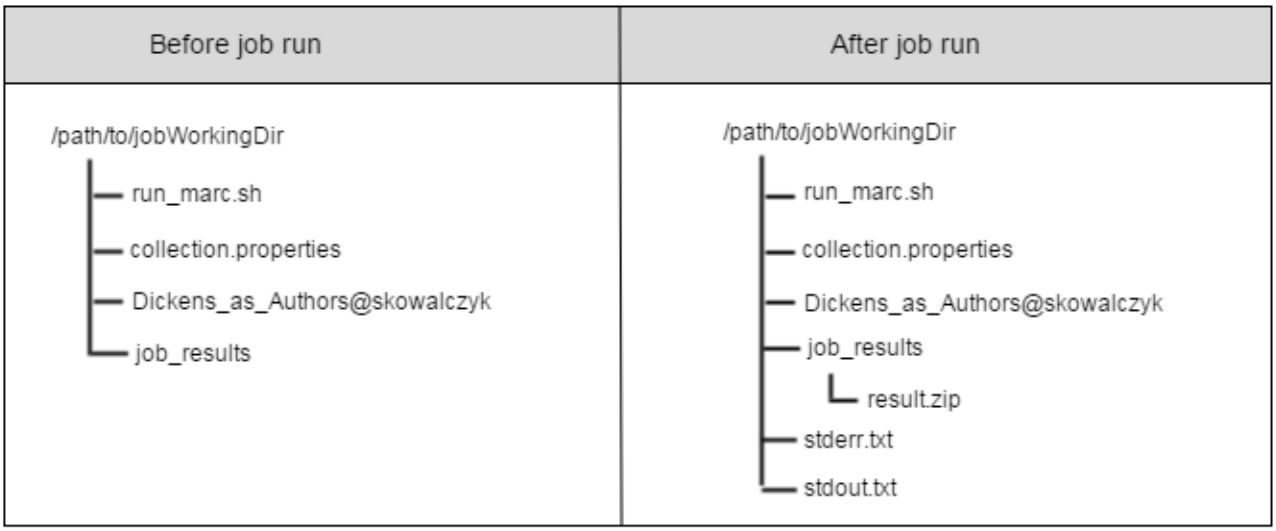
```
java -jar /path/to/MarcDownloader1.7.jar
```

runwc.sh, the shell script wrapper for Word_Count

```
java -classpath /path/to/htrc-uncamp-deplwc-1.4.jar htrc.uncamp.DeployableWordCountClient
```

The shell script wrappers are copied to the job working directory by the job submission module.

An example of a Marc_downloader job working directory before and after the job run is included below.



Algorithm XML File

The algorithm XML file describes different aspects of the algorithm such as the input parameters it expects, the job results it creates, and the name of the property file. The algorithm XML file is used by HTRC services such as the Analytics Gateway and the job submission module. The Analytics Gateway uses the XML file to determine what input fields the user needs to provide to run the algorithm. The XML file plays a key role in the communication needed between the job submission module and the algorithm, e.g., what input parameters need to be written to the property file, the names of the shell script and the property file, and the names of the result files produced by the algorithm. The XML file is only needed at the time of submission of the algorithm to HTRC.

The entries in an algorithm XML file are:

- Algorithm name
- Short name for the algorithm (at most 8 characters in length)
- Version
- Brief text description
- Authors
- User input parameters, name, type, label, brief description
- Name of the shell script wrapper
- Name of the property file expected
- Key-value pairs for each property to be written to the property file
    - Bash-style variables are used to refer to input parameters and system-provided values
- Names and types of result files, and whether they are optional

An example algorithm XML file is provided below.

Example of an algorithm XML file

```
 <algorithm>

  <info>
    <name>TestAlgorithm</name>
    <short_name>TestAlgo</short_name>
    <version>1.0</version>
    <description>TestAlgorithm performs ABC analysis and produces XYZ results.</description>
    <authors>
        <author name="Htrc Superuser"/>
    </authors>
    <supportUrl>http://help.me/please</supportUrl>

    <parameters>
        <param
            name="input_collection"
            type="collection"
            required="true"
            size_limit="3000"
                        defaultValue="default_collection">
            <label>Please select a collection for analysis</label>
            <description>A collection argument example.</description>
        </param>
        <param
            name="n"
            type="integer"
            required="true"
                        defaultValue="10">
            <label>Select a number</label>
            <description>An example of a non-collection argument.</description>
        </param>
    </parameters>
  </info>

  <!-- walltime should have the form hhh:mm:ss or hh:mm:ss -->
  <execution_info>
    <input_size min="0" max="500">
      <number_of_nodes>1</number_of_nodes>
      <number_of_processors_per_node>8</number_of_processors_per_node>
      <walltime>03:00:00</walltime>
      <java_max_heap_size>10g</java_max_heap_size>
    </input_size>
    <input_size min="501" max="3000">
      <number_of_nodes>1</number_of_nodes>
      <number_of_processors_per_node>16</number_of_processors_per_node>
      <walltime>05:00:00</walltime>
      <java_max_heap_size>28g</java_max_heap_size>
    </input_size>
  </execution_info>

  <run_script>run_TestAlgorithm.sh</run_script>
  <properties_file_name>TestAlgorithm.properties</properties_file_name>

  <system_properties>
      <e key="volume_id">$input_collection</e>
      <e key="num">$n</e>
      <e key="max_volumes">2000</e>
      <e key="token">$auth_token</e>
      <e key="data_api_url">$data_api_url</e>
      <e key="output_folder">$output_dir</e>
  </system_properties>

  <results>
      <result type="text/html" name="output.html"/>
      <!-- if the "optional" attribute is true, then the result may or may not be produced in a successful job
run; "optional" is false by default -->
      <result type="text/plain" name="volume_errors.txt" optional="true"/>
  </results>

</algorithm>
```

Parameters

The input parameters expected by an algorithm are described in the <parameters> element. This section is used by the HTRC Analytics Gateway to display required input fields for the algorithm. Each parameter has six pieces of key information:

- **name -** Later on in the file, in the <system_properties> section, this name can be used to refer to the user-provided value for this parameter.
- **type -** Currently there are only two "real" types: `collection` and `not_collection`. A collection argument needs to be treated differently, as the volume list for the collection must be downloaded to a file while a non-collection argument does not.
- **required -** This is a boolean flag to indicate whether or not this argument must be provided by the user.
- **label -** This is used by the Analytics Gateway to label the UI widget that asks for this value.
- **description -** This is a longer text description of what the input is. It should still be reasonably short to keep the Analytics Gateway UI clean.
- **defaultValue** - Optional attribute to specify default value to use for this parameter.

Parameters of type "collection" may have an additional attribute, **size_limit**, which specifies the maximum no. of volumes allowed in the input workset. If the input workset has more than size_limit volumes, then the job does not execute, and an appropriate error message is returned. If size_limit is not specified for a parameter of type "collection", then no check is made by the job submission module on the number of volumes in the input workset, and the algorithm is allowed to run on worksets of all sizes.

Execution information

<execution_info> is used to provide resource allocation information needed for optimal execution of the algorithm. This section is usually not provided by algorithm developers, and is filled in by HTRC staff. Resource allocation is parameterized by the total size of input worksets to the job. In the example shown below, resource allocation is specified for 2 ranges: 0 to 500 volumes, and 501 to 3000 volumes. Note that input_size refers to the total number of volumes in the input worksets, i.e., the total size of all input worksets, allowing for algorithms that have multiple input worksets.

Example of <execution_info> in an algorithm XML file

```
<execution_info>
  <input_size min="0" max="500">
    <number_of_nodes>1</number_of_nodes>
    <number_of_processors_per_node>8</number_of_processors_per_node>
    <walltime>03:00:00</walltime>
    <java_max_heap_size>10g</java_max_heap_size>
  </input_size>
  <input_size min="501" max="3000">
    <number_of_nodes>1</number_of_nodes>
    <number_of_processors_per_node>16</number_of_processors_per_node>
    <walltime>05:00:00</walltime>
    <java_max_heap_size>28g</java_max_heap_size>
  </input_size>
</execution_info>
```

The <number_of_nodes> element specifies how many nodes on the HPC system are allocated to the algorithm. So far, all algorithms run on a single node. If

The <number_of_processors_per_node> element is used to specify how many processors on a single node should be requested to run the algorithm. The number of processors is determined by the parallelism factor in the algorithm, and limited by the no. of cores on a node of the HPC system on which jobs are executed. The HPC system, Karst, used for HTRC jobs at this time, has 16 cores per node. If the algorithm does not contain any parallel computation, then <number_of_processors_per_node> should be set to 1. One might set <number_of_processors_per_node> to a number larger than 8, not to access as many processors, but rather to gain access to all of the other resources on the node. For example, by setting this number to 16, one gains access to the whole node, and therefore all the memory on the node. Nodes on Karst have 32 GB RAM.

<walltime> specifies the amount of time that will be allocated to the algorithm on the HPC system on which it is executed. If it is unable to complete in the allocated walltime, the algorithm is killed by the HPC system.

<java_max_heap_size> refers to the maximum heap size to be allocated on the JVM used to run the algorithm. This applies only to algorithms that run on JVMs.

If any of these 4 items are not specified, then default values are used. If <execution_info> itself is not provided, default values are used, irrespective of the input workset size(s).

Shell script and property file names

The <run_script> tag is used to specify the name of the shell script wrapper of the algorithm. <properties_file_name> tells the job submission module what the property file used by the algorithm should be named.

Property file content

The <system_properties> element contains details used by the job submission module to create the property file needed by the algorithm. It contains key names, and the values with which keys are to be associated in the property file. Valid values include (a) the values of input parameters specified in the <parameters> element (e.g., the value "$input_collection" for the key "volume_id" in the example XML file shown above), (b) values of system-provided variables (e.g., value "$auth_token" for the key "token"), and (c) string literals (e.g., the value "2000" for the key "max_volumes").

Values of variables are obtained by prefixing the variable names with a '$'. In some cases you may wish to insert a variable directly into a string. The "output_folder" key in the example file does this. To avoid ambiguous parsing the variable name following the '$' is placed inside curly braces. 'foo${input} baz' would become 'foobarbaz' if 'input' had been bound to 'bar'.

The list of system-provided variables, whose values are supplied at runtime by the job submission module, are:

- data_api_url
- auth_token
- output_dir

"output_dir" refers to the folder in which the algorithm is expected to place its result files. The other variables are self-explanatory.

Job results

The <results> element describes all the result files produced by the algorithm. Each <result> element specifies the name and MIME type of the result file. In addition, an "optional" attribute may also be specified. If "optional" is true, then the result may or may not be produced in a successful job run. "optional" is false by default. A value of false indicates that the result is produced in successful job runs.