

Templeton

Table of contents

| | |
|------------------|---|
| 1 Templeton..... | 2 |
|------------------|---|

1 Templeton

1.1 Templeton

1.1.1 Introduction

Templeton provides a REST-like web API for [HCatalog](#) and related Hadoop components. As shown in the figure below, developers make HTTP requests to access [Hadoop MapReduce](#), [Pig](#), [Hive](#), and [HCatalog DDL](#) from within applications. Data and code used by Templeton is maintained in [HDFS](#). HCatalog DDL commands are executed directly when requested. MapReduce, Pig, and Hive jobs are placed in queue by Templeton and can be monitored for progress or stopped as required. Developers specify a location in HDFS into which Templeton should place Pig, Hive, and MapReduce results.



1.1.2 URL format

Templeton resources are accessed using the following URL format:

```
http://yourserver/templeton/v1/resource
```

where "yourserver" is replaced with your server name, and "resource" is replaced by the Templeton resource name.

For example, to check if the Templeton server is running you could access the following URL:

```
http://www.myserver.com/templeton/v1/status
```

1.1.3 Security

The current version of Templeton supports two types of security:

- Default security (without additional authentication)
- Authentication via [Kerberos](#)

1.1.3.1 Standard Parameters

Every Templeton resource can accept the following parameters to aid in authentication:

- `user.name`: The user name as a string. Only valid when using default security.
- SPNEGO credentials: When running with Kerberos authentication.

1.1.3.2 Security Error Response

If the `user.name` parameter is not supplied when required, Templeton returns the following error:

```
{
  "error": "No user found. Missing user.name parameter."
}
```

1.1.4 WebHDFS and Code Push

Data and code that are used by Templeton resources must first be placed in Hadoop. The current version of Templeton does not attempt to integrate or replace existing web interfaces that can perform this task, like [WebHDFS](#). (Integration of these functions in some way, perhaps forwarding, is planned for a future release.) When placing files into HDFS is required you can use whatever method is most convenient for you.

1.1.5 Error Codes and Responses

The Templeton server returns the following HTTP status codes.

- **200 OK**: Success!
- **400 Bad Request**: The request was invalid.
- **401 Unauthorized**: Credentials were missing or incorrect.
- **404 Not Found**: The URI requested is invalid or the resource requested does not exist.
- **500 Internal Server Error**: We received an unexpected result.
- **503 Busy, please retry**: The server is busy.

Other data returned directly by Templeton is currently returned in JSON format. JSON responses are limited to 1MB in size. Responses over this limit must be stored into HDFS using provided options instead of being directly returned. If an HCatalog DDL command

might return results greater than 1MB, it's suggested that a corresponding Hive request be executed instead.

1.1.6 Log Files

The Templeton server creates three log files when in operation:

- **templeton.log** is the log4j log. This the main log the application writes to.
- **templeton-console.log** is what Java writes to stdout when Templeton is started. It is a small amount of data, similar to "hcat.out".
- **templeton-console-error.log** is what Java writes to stderr, similar to "hcat.err".

In the templeton-log4j.properties file you can set the location of these logs using the variable templeton.log.dir. This log4j.properties file is set in the server startup script.

1.1.7 Project Name

The Templeton project is named after the a character in the award-winning children's novel Charlotte's Web, by E. B. White. The novel's protagonist is a pig named Wilber. Templeton is a rat who helps Wilber by running errands and making deliveries.

1.2 Installation

1.2.1 Introduction

Templeton is deep in the middle of development and does not yet have a smooth install procedure. It is also designed to connect together services that are not normally connected and therefore has a complex configuration. As such, this version of Templeton should only be installed by expert developers.

1.2.2 Procedure

1. Ensure that the [required related installations](#) are in place, and place required files into the [Hadoop distributed cache](#).
2. Download and unpack the Templeton distribution.
3. Set the TEMPLETON_HOME environment variable to the base of the Templeton installation. This is used to find the Templeton configuration.
4. Review the [Templeton configuration](#) and update or create templeton-site.xml as required. Ensure that site specific component installation locations are accurate, especially the Hadoop configuration path. Configuration variables that use a filesystem path try to have reasonable defaults, but it's always safe to specify a full and complete path.
5. Build Templeton using the command `ant jar` from the top level Templeton directory.

6. Start the Templeton server with the command `bin/templeton_server.sh start`.
7. Check that your local install works. Assuming that Tomcat is running on port 8080, the following command would give output similar to that shown.

```
% curl -i http://localhost:50111/templeton/v1/status
HTTP/1.1 200 OK
Content-Type: application/json
Transfer-Encoding: chunked
Server: Jetty(7.6.0.v20120127)

{"status":"ok","version":"v1"}
%
```

1.2.3 Server Commands

- **Start the server:** `bin/templeton_server.sh start`
- **Stop the server:** `bin/templeton_server.sh stop`
- **End-to-end build, run, test:** `ant e2e`

1.2.4 Requirements

- [Ant](#), version 1.8 or higher
- [Hadoop](#), version 0.20.205.0
- [ZooKeeper](#) is required if you are using the ZooKeeper storage class. (Be sure to review and update the ZooKeeper related [Templeton configuration](#).)
- [HCatalog](#). Version 0.4.1 or higher is recommended for users who want to use the REST ddl commands. (Be sure to review and update the related [Templeton configuration](#).)
- [Hadoop Distributed Cache](#). To use the [Hive](#), [Pig](#), or [hadoop/streaming](#) resources, see instructions below for placing the required files in the Hadoop Distributed Cache.

1.2.5 Hadoop Distributed Cache

Templeton requires some files be accessible on the [Hadoop distributed cache](#). For example, to avoid the installation of Pig and Hive everywhere on the cluster, Templeton gathers a version of Pig or Hive from the Hadoop distributed cache whenever those resources are invoked. After placing the following components into HDFS please update the site configuration as required for each.

- **Hive:** [Download](#) the HCatalog tar.gz file and place it in HDFS. (If you need a version that is not yet released, you may need to build it yourself following the HCatalog instructions.)

```
hadoop fs -put /tmp/hcatalog-0.3.0.tar.gz /user/templeton/hcatalog-0.3.0.tar.gz
```

- **Pig:** [Download](#) the Pig tar.gz file and place it into HDFS. For example:

```
hadoop fs -put /tmp/pig-0.9.2.tar.gz /user/templeton/pig-0.9.2.tar.gz
```

- **Hadoop Streaming:** Place `hadoop-streaming.jar` into HDFS. For example, use the following command, substituting your path to the jar for the one below.

```
hadoop fs -put $HADOOP_PREFIX/hadoop-0.20.205.0/contrib/streaming/hadoop-  
streaming-0.20.205.0.jar \  
/user/templeton/hadoop-streaming.jar
```

- **Override Jars:** Place override jars required (if any) into HDFS. *Note:* As of this writing, all released versions of Hadoop require a patch to properly run Templeton. This patch is distributed with Templeton (located at `templeton/src/hadoop_temp_fix/ugi.jar`) and should be placed into HDFS, as reflected in the current default configuration.

```
hadoop fs -put ugi.jar /user/templeton/ugi.jar
```

The location of these files in the cache, and the location of the installations inside the archives, can be specified using the following Templeton configuration variables. (See the [Configuration](#) documentation for more information on changing Templeton configuration parameters.)

| Name | Default | Description |
|--------------------------------|---|--|
| templeton.pig.archive | <code>hdfs:///user/templeton/ pig-0.9.2.tar.gz</code> | The path to the Pig archive. |
| templeton.pig.path | <code>pig-0.9.2.tar.gz/ pig-0.9.2/bin/pig</code> | The path to the Pig executable. |
| templeton.hive.archive | <code>hdfs:///user/templeton/ hcatalog-0.3.0.tar.gz</code> | The path to the Hive archive. |
| templeton.hive.path | <code>hcatalog-0.3.0.tar.gz/ hcatalog-0.3.0/bin/hive</code> | The path to the Hive executable. |
| templeton.streaming.jar | <code>hdfs:///user/templeton/ hadoop-streaming.jar</code> | The path to the Hadoop streaming jar file. |
| templeton.override.jars | <code>hdfs:///user/templeton/ ugi.jar</code> | Jars to add to the <code>HADOOP_CLASSPATH</code> for all Map Reduce jobs. These jars must exist on HDFS. |

1.3 Configuration

The configuration for Templeton merges the normal Hadoop configuration with the Templeton specific variables. Because Templeton is designed to connect services that are not normally connected, the configuration is more complex than might be desirable.

The Templeton specific configuration is split into three layers:

1. **templeton-default.xml** - All the configuration variables that Templeton needs. This file sets the defaults that ship with Templeton and should only be changed by Templeton developers. Do not copy this file and/or change it to maintain local installation settings. Because templeton-default.xml is present in the Templeton war file, editing a local copy of it will not change the configuration.
2. **templeton-dist.xml** - The (possibly empty) configuration file that can set variables for a particular distribution, such as an RPM file.
3. **templeton-site.xml** - The (possibly empty) configuration file in which the system administrator can set variables for their Hadoop cluster. Create this file and maintain entries in it for configuration variables that require you to override default values based on your local installation.

The configuration files are loaded in this order with later files overriding earlier ones.

Note: the Templeton server will require restart after any change to the configuration.

To find the configuration files, Templeton first attempts to load a file from the CLASSPATH and then looks in the directory specified in the TEMPLETON_HOME environment variable.

Configuration files may access the special environment variable `env` for all environment variables. For example, the pig executable could be specified using:

```
${env.PIG_HOME}/bin/pig
```

Configuration variables that use a filesystem path try to have reasonable defaults. However, it's always safe to specify the full and complete path if there is any uncertainty.

Note: The location of the log files created by Templeton and some other properties of the logging system are set in the templeton-log4j.properties file.

1.3.1 Variables

| Name | Default | Description |
|------------------------------------|--|---------------------------------------|
| templeton.port | 50111 | The HTTP port for the main server. |
| templeton.hadoop.config.dir | <code>\$(env.HADOOP_CONFIG_DIR)</code> | The path to the Hadoop configuration. |

| Name | Default | Description |
|-----------------------------------|--|---|
| templeton.jar | <code>\${env.TEMPLETON_HOME}/templeton/templeton-0.1.0-dev.jar</code> | The path to the Templeton jar file. |
| templeton.libjars | <code>\${env.TEMPLETON_HOME}/lib/zookeeper-3.3.4.jar</code> | Jars to add to the classpath. |
| templeton.override.jars | <code>hdfs:///user/templeton/ugi.jar</code> | Jars to add to the HADOOP_CLASSPATH for all Map Reduce jobs. These jars must exist on HDFS. |
| templeton.override.enabled | <code>true</code> | Enable the override path in templeton.override.jars |
| templeton.streaming.jar | <code>hdfs:///user/templeton/hadoop-streaming.jar</code> | The hdfs path to the Hadoop streaming jar file. |
| templeton.hadoop | <code>\${env.HADOOP_PREFIX}/bin/hadoop</code> | The path to the Hadoop executable. |
| templeton.pig.archive | <code>hdfs:///user/templeton/pig-0.9.2.tar.gz</code> | The path to the Pig archive. |
| templeton.pig.path | <code>pig-0.9.2.tar.gz/pig-0.9.2/bin/pig</code> | The path to the Pig executable. |
| templeton.hcat | <code>\${env.HCAT_PREFIX}/bin/hcat</code> | The path to the Hcatalog executable. |
| templeton.hive.archive | <code>hdfs:///user/templeton/hcatalog-0.3.0.tar.gz</code> | The path to the Hive archive. |
| templeton.hive.path | <code>hcatalog-0.3.0.tar.gz/hcatalog-0.3.0/bin/hive</code> | The path to the Hive executable. |
| templeton.hive.properties | <code>hive.metastore.local=false hive.metastore.uris=thrift://localhost:9933, hive.metastore.sasl.enabled=false</code> | Properties to set when running hive. |
| templeton.exec.encoding | <code>UTF-8</code> | The encoding of the stdout and stderr data. |
| templeton.exec.timeout | <code>10000</code> | How long in milliseconds a program is allowed to run on the Templeton box. |

| Name | Default | Description |
|---|-------------------------------|--|
| templeton.exec.max-procs | 16 | The maximum number of processes allowed to run at once. |
| templeton.exec.max-output-bytes | 1048576 | The maximum number of bytes from stdout or stderr stored in ram. |
| templeton.exec.envs | HADOOP_PREFIX, HADOOP_HOME | The environment variables passed through to exec. |
| templeton.zookeeper.hosts | 127.0.0.1:2181 | ZooKeeper servers, as comma separated host:port pairs |
| templeton.zookeeper.session-timeout | 30000 | ZooKeeper session timeout in milliseconds |
| templeton.callback.retry.interval | 10000 | How long to wait between callback retry attempts in milliseconds |
| templeton.callback.retry.attempts | 5 | How many times to retry the callback |
| templeton.storage.class | org.apache.hcatalog.templeton | The class to use as storage |
| templeton.storage.root | /templeton-hadoop | The path to the directory to use for storage |
| templeton.hdfs.cleanup.interval | 43200000 | The maximum delay between a thread's cleanup checks |
| templeton.hdfs.cleanup.maxage | 604800000 | The maximum age of a templeton job |
| templeton.zookeeper.cleanup.interval | 43200000 | The maximum delay between a thread's cleanup checks |
| templeton.zookeeper.cleanup.maxage | 604800000 | The maximum age of a templeton job |

1.4 Reference

1.4.1 Templeton Resources

| Resource | Description |
|--------------------------|---|
| :version | Returns a list of supported response types. |
| status | Returns the Templeton server status. |

| Resource | Description |
|--|---|
| version | Returns the a list of supported versions and the current version. |
| ddl | Performs an HCatalog DDL command. |
| ddl/database | List HCatalog databases. |
| ddl/database/:db (GET) | Describe an HCatalog database. |
| ddl/database/:db (PUT) | Create an HCatalog database. |
| ddl/database/:db (DELETE) | Delete (drop) an HCatalog database. |
| ddl/database/:db/table | List the tables in an HCatalog database. |
| ddl/database/:db/table/:table (GET) | Describe an HCatalog table. |
| ddl/database/:db/table/:table (PUT) | Create a new HCatalog table. |
| ddl/database/:db/table/:table (POST) | Rename an HCatalog table. |
| ddl/database/:db/table/:table (DELETE) | Delete (drop) an HCatalog table. |
| ddl/database/:db/table/:existingtable/like/:newtable (PUT) | Create a new HCatalog table like an existing one. |
| ddl/database/:db/table/:table/partion | List all partitions in an HCatalog table. |
| ddl/database/:db/table/:table/partion/:partition (GET) | Describe a single partition in an HCatalog table. |
| ddl/database/:db/table/:table/partion/:partition (PUT) | Create a partition in an HCatalog table. |
| ddl/database/:db/table/:table/partion/:partition (DELETE) | Delete (drop) a partition in an HCatalog table. |
| ddl/database/:db/table/:table/column | List the columns in an HCatalog table. |
| ddl/database/:db/table/:table/column/:column (GET) | Describe a single column in an HCatalog table. |
| ddl/database/:db/table/:table/column/:column (PUT) | Create a column in an HCatalog table. |
| ddl/database/:db/table/:table/property (GET) | List table properties. |
| ddl/database/:db/table/:table/property/:property (GET) | Return the value of a single table property. |
| ddl/database/:db/table/:table/property/:property (PUT) | Set a table property. |
| mapreduce/streaming | Creates and queues Hadoop streaming MapReduce jobs. |

| Resource | Description |
|---------------------------------------|---|
| mapreduce/jar | Creates and queues standard Hadoop MapReduce jobs. |
| pig | Creates and queues Pig jobs. |
| hive | Runs Hive queries and commands. |
| queue | Returns a list of all jobids registered for the user. |
| queue/:jobid (GET) | Returns the status of a job given its ID. |
| queue/:jobid (DELETE) | Kill a job given its ID. |

1.4.2 GET :version

1.4.2.1 Description

Returns a list of the response types supported by Templeton.

1.4.2.2 URL

`http://www.myserver.com/templeton/:version`

1.4.2.3 Parameters

| Name | Description | Required? | Default |
|-----------------|---|-----------|---------|
| :version | The Templeton version number. (Currently this must be "v1") | Required | None |

1.4.2.4 Results

| Name | Description |
|----------------------|--|
| responseTypes | A list of all supported response types |

1.4.2.5 Example

Curl Command

```
% curl -s 'http://localhost:50111/templeton/v1'
```

JSON Output

```
{
```

```
"responseTypes": [
  "application/json"
]
```

JSON Output (error)

```
{
  "error": "null for uri: http://localhost:50111/templeton/v2"
}
```

1.4.3 GET status

1.4.3.1 Description

Returns the current status of the Templeton server. Useful for heartbeat monitoring.

1.4.3.2 URL

`http://www.myserver.com/templeton/v1/status`

1.4.3.3 Parameters

Only the [standard parameters](#) are accepted.

1.4.3.4 Results

| Name | Description |
|----------------|---|
| status | "ok" if the Templeton server was contacted. |
| version | String containing the version number similar to "v1". |

1.4.3.5 Example

Curl Command

```
% curl -s 'http://localhost:50111/templeton/v1/status'
```

JSON Output

```
{
  "status": "ok",
  "version": "v1"
}
```

1.4.4 GET version

1.4.4.1 Description

Returns a list of supported versions and the current version.

1.4.4.2 URL

`http://www.myserver.com/templeton/v1/version`

1.4.4.3 Parameters

Only the [standard parameters](#) are accepted.

1.4.4.4 Results

| Name | Description |
|--------------------------|-----------------------------------|
| supportedVersions | A list of all supported versions. |
| version | The current version. |

1.4.4.5 Example

Curl Command

```
% curl -s 'http://localhost:50111/templeton/v1/version'
```

JSON Output

```
{
  "supportedVersions": [
    "v1"
  ],
  "version": "v1"
}
```

1.4.5 ddl

1.4.5.1 Templeton DDL Resources

| Resource | Description |
|--|-----------------------------------|
| ddl | Performs an HCatalog DDL command. |
| ddl/database | List HCatalog databases. |
| ddl/database/:db (GET) | Describe an HCatalog database. |

| Resource | Description |
|--|---|
| ddl/database/:db (PUT) | Create an HCatalog database. |
| ddl/database/:db (DELETE) | Delete (drop) an HCatalog database. |
| ddl/database/:db/table | List the tables in an HCatalog database. |
| ddl/database/:db/table/:table (GET) | Describe an HCatalog table. |
| ddl/database/:db/table/:table (PUT) | Create a new HCatalog table. |
| ddl/database/:db/table/:table (POST) | Rename an HCatalog table. |
| ddl/database/:db/table/:table (DELETE) | Delete (drop) an HCatalog table. |
| ddl/database/:db/table/:existingtable/like/:newtable (PUT) | Create a new HCatalog table like an existing one. |
| ddl/database/:db/table/:table/partion | List all partitions in an HCatalog table. |
| ddl/database/:db/table/:table/partion/:partition (GET) | Describe a single partition in an HCatalog table. |
| ddl/database/:db/table/:table/partion/:partition (PUT) | Create a partition in an HCatalog table. |
| ddl/database/:db/table/:table/partion/:partition (DELETE) | Delete (drop) a partition in an HCatalog table. |
| ddl/database/:db/table/:table/column | List the columns in an HCatalog table. |
| ddl/database/:db/table/:table/column/:column (GET) | Describe a single column in an HCatalog table. |
| ddl/database/:db/table/:table/column/:column (PUT) | Create a column in an HCatalog table. |
| ddl/database/:db/table/:table/property (GET) | List table properties. |
| ddl/database/:db/table/:table/property/:property (GET) | Return the value of a single table property. |
| ddl/database/:db/table/:table/property/:property (PUT) | Set a table property. |

1.4.5.2 POST ddl

Description

Performs an [HCatalog DDL](#) command. The command is executed immediately upon request. Responses are limited to 1MB. For requests which may return longer results consider using the [Hive resource](#) as an alternative.

URL

`http://www.myserver.com/templeton/v1/ddl`

Parameters

| Name | Description | Required? | Default |
|--------------------|---|-----------|---------|
| exec | The HCatalog ddl string to execute | Required | None |
| group | The user group to use when creating a table | Optional | None |
| permissions | The permissions string to use when creating a table. The format is "rwxrw-r-x". | Optional | None |

Results

| Name | Description |
|-----------------|--|
| stdout | A string containing the result HCatalog sent to standard out (possibly empty). |
| stderr | A string containing the result HCatalog sent to standard error (possibly empty). |
| exitcode | The exitcode HCatalog returned. |

Example**Curl Command**

```
% curl -s -d user.name=ctdean \
  -d 'exec=show tables;' \
  'http://localhost:50111/templeton/v1/ddl'
```

JSON Output

```
{
  "stdout": "important_table
            my_other_table
            my_table
            my_table_2
            pokes
            ",
  "stderr": "WARNING: org.apache.hadoop.metrics.jvm.EventCounter is deprecated...
            Hive history file=/tmp/ctdean/hive_job_log_ctdean_201111111258_2117356679.txt
            OK
}
```

```

        Time taken: 1.202 seconds
        ",
        "exitcode": 0
    }

```

JSON Output (error)

```

{
  "stdout": "",
  "stderr": "WARNING: org.apache.hadoop.metrics.jvm.EventCounter is deprecated...
            Hive history file=/tmp/ctdean/hive_job_log_ctdean_201204051246_689834593.txt
            FAILED: Parse Error: line 1:5 Failed to recognize predicate 'tab'...

        ",
  "exitcode": 11
}

```

1.4.5.3 GET ddl/database

Description

List the databases in HCatalog.

URL

<http://www.myserver.com/templeton/v1/ddl/database>

Parameters

| Name | Description | Required? | Default |
|-------------|---|-----------|----------------|
| like | List only databases whose names match the specified pattern | Optional | "*" (List all) |

Results

| Name | Description |
|------------------|--------------------------|
| databases | A list of database names |

Example

Curl Command

```
% curl -s 'http://localhost:50111/templeton/v1/ddl/database?user.name=ctdean&like=n*'
```

JSON Output


```
{
  "databases": [
    "newdb",
    "newdb2"
  ]
}
```

1.4.5.4 GET ddl/database/:db

Description

Describe a database. (Note: this resource has a "format=extended" parameter however the output structure does not change if it is used.)

URL

`http://www.myserver.com/templeton/v1/ddl/database/:db`

Parameters

| Name | Description | Required? | Default |
|------------|-------------------|-----------|---------|
| :db | The database name | Required | None |

Results

| Name | Description |
|-----------------|-------------------------|
| location | The database location |
| params | The database parameters |
| comment | The database comment |
| database | The database name |

Example

Curl Command

```
% curl -s 'http://localhost:50111/templeton/v1/ddl/database/newdb?user.name=ctdean'
```

JSON Output

```
{
  "location": "hdfs://localhost:9000/warehouse/newdb.db",
  "params": "{a=b}",
  "comment": "Hello there",
  "database": "newdb"
}
```

JSON Output (error)

```
{
  "error": "No such database: newdb",
  "errorCode": 404
}
```

1.4.5.5 PUT ddl/database/:db

Description

Create a database.

URL

`http://www.myserver.com/templeton/v1/ddl/database/:db`

Parameters

| Name | Description | Required? | Default |
|--------------------|--|-----------|---------|
| :db | The database name | Required | None |
| group | The user group to use | Optional | None |
| permissions | The permissions string to use | Optional | None |
| location | The database location | Optional | None |
| comment | A comment for the database, like a description | Optional | None |
| properties | The database properties | Optional | None |

Results

| Name | Description |
|-----------------|-------------------|
| database | The database name |

Example

Curl Command

```
% curl -s -X PUT -HContent-type:application/json \
  -d '{ "comment": "Hello there",
        "location": "hdfs://localhost:9000/user/hive/my_warehouse",
        "properties": { "a": "b" } }' \
```

```
'http://localhost:50111/templeton/v1/ddl/database/newdb?user.name=rachel'
```

JSON Output

```
{
  "database": "newdb"
}
```

1.4.5.6 DELETE ddl/database/:db

Description

Delete a database.

URL

`http://www.myserver.com/templeton/v1/ddl/database/:db`

Parameters

| Name | Description | Required? | Default |
|--------------------|--|-----------|---------|
| :db | The database name | Required | None |
| ifExists | Hive returns an error if the database specified does not exist, unless ifExists is set to true. | Optional | false |
| option | Parameter set to either "restrict" or "cascade". Restrict will remove the schema if all the tables are empty. Cascade removes everything including data and definitions. | Optional | None |
| group | The user group to use | Optional | None |
| permissions | The permissions string to use. The format is "rwxrw-r-x". | Optional | None |

Results

| Name | Description |
|-----------------|-------------------|
| database | The database name |

Example**Curl Command**

```
% curl -s -X DELETE "http://localhost:50111/templeton/v1/ddl/database/newdb?
user.name=ctdean"
```

JSON Output

```
{
  "database": "newdb"
}
```

JSON Output (error)

```
{
  "errorDetail": "
    NoSuchObjectException(message:There is no database named my_db)
    at org.apache.hadoop.hive.metastor...
  ",
  "error": "There is no database named newdb",
  "errorCode": 404,
  "database": "newdb"
}
```

1.4.5.7 GET ddl/database/:db/table**Description**

List the tables in an HCatalog database.

URL

`http://www.myserver.com/templeton/v1/ddl/database/:db/table`

Parameters

| Name | Description | Required? | Default |
|-------------|--|-----------|-----------------------|
| :db | The database name | Required | None |
| like | List only tables whose names match the specified pattern | Optional | "*" (List all tables) |

Results

| Name | Description |
|-----------------|-----------------------|
| tables | A list of table names |
| database | The database name |

Example**Curl Command**

```
% curl -s 'http://localhost:50111/templeton/v1/ddl/database/default/table?
user.name=ctdean&like=m*'
```

JSON Output

```
{
  "tables": [
    "my_table",
    "my_table_2",
    "my_table_3"
  ],
  "database": "default"
}
```

JSON Output (error)

```
{
  "errorDetail": "
    org.apache.hadoop.hive.ql.metadata.HiveException: ERROR: The database defaultsd does
    not exist.
      at org.apache.hadoop.hive.ql.exec.DDLTask.switchDatabase(DDLTask.java:3122)
      at org.apache.hadoop.hive.ql.exec.DDLTask.execute(DDLTask.java:224)
      at org.apache.hadoop.hive.ql.exec.Task.executeTask(Task.java:134)
      at org.apache.hadoop.hive.ql.exec.TaskRunner.runSequential(TaskRunner.java:57)
      at org.apache.hadoop.hive.ql.Driver.launchTask(Driver.java:1332)
      at org.apache.hadoop.hive.ql.Driver.execute(Driver.java:1123)
      at org.apache.hadoop.hive.ql.Driver.run(Driver.java:931)
      at org.apache.hcatalog.cli.HCatDriver.run(HCatDriver.java:42)
      at org.apache.hcatalog.cli.HCatCli.processCmd(HCatCli.java:247)
      at org.apache.hcatalog.cli.HCatCli.processLine(HCatCli.java:203)
      at org.apache.hcatalog.cli.HCatCli.main(HCatCli.java:162)
      at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
      at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
      at
      sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
      at java.lang.reflect.Method.invoke(Method.java:597)
      at org.apache.hadoop.util.RunJar.main(RunJar.java:156)
    ",
  "error": "FAILED: Error in metadata: ERROR: The database defaultsd does not exist.",
  "errorCode": 500,
  "database": "defaultsd"
}
```

}

1.4.5.8 GET ddl/database/:db/table/:table**Description**

Describe an HCatalog table. Normally returns a simple list of columns (using "desc table"), but the extended format will show more information (using "show table extended like").

URL

`http://www.myserver.com/templeton/v1/ddl/database/:db/table/:table`

`http://www.myserver.com/templeton/v1/ddl/database/:db/table/:table?format=extended`

Parameters

| Name | Description | Required? | Default |
|---------------|--|-----------|--------------|
| :db | The database name | Required | None |
| :table | The table name | Required | None |
| format | Set "format=extended" to see additional information (using "show table extended like") | Optional | Not extended |

Results

| Name | Description |
|---|----------------------------------|
| columns | A list of column names and types |
| database | The database name |
| table | The table name |
| partitioned (extended only) | True if the table is partitioned |
| location (extended only) | Location of table |
| outputFormat (extended only) | Output format |
| owner (extended only) | The owner's user name |
| partitionColumns (extended only) | List of the partition columns |
| inputFormat (extended only) | Input format |

Example**Curl Command (simple)**

```
% curl -s 'http://localhost:50111/templeton/v1/ddl/database/default/table/my_table?user.name=ctdean'
```

JSON Output (simple)

```
{
  "columns": [
    {
      "name": "id",
      "type": "bigint"
    },
    {
      "name": "user",
      "comment": "The user name",
      "type": "string"
    },
    {
      "name": "my_p",
      "type": "string"
    },
    {
      "name": "my_q",
      "type": "string"
    }
  ],
  "database": "default",
  "table": "my_table"
}
```

Curl Command (extended)

```
% curl -s 'http://localhost:50111/templeton/v1/ddl/database/default/table/test_table?user.name=ctdean&format=extended'
```

JSON Output (extended)

```
{
  "partitioned": true,
  "location": "hdfs://ip-10-77-6-151.ec2.internal:8020/apps/hive/warehouse/test_table",
  "outputFormat": "org.apache.hadoop.hive ql.io.RCFileOutputFormat",
  "columns": [
    {
      "name": "id",
      "type": "bigint"
    },
    {
      "name": "price",
      "comment": "The unit price",
      "type": "float"
    }
  ]
}
```

```

    }
  ],
  "owner": "ctdean",
  "partitionColumns": [
    {
      "name": "country",
      "type": "string"
    }
  ],
  "inputFormat": "org.apache.hadoop.hive ql.io.RCFileInputFormat",
  "database": "default",
  "table": "test_table"
}

```

JSON Output (error)

```

{
  "error": "Table xtest_table does not exist",
  "errorCode": 404,
  "database": "default",
  "table": "xtest_table"
}

```

1.4.5.9 PUT ddl/database/:db/table/:table

Description

Create a new HCatalog table. For more information, please refer to the [Hive documentation](#).

URL

`http://www.myserver.com/templeton/v1/ddl/database/:db/table/:table`

Parameters

| Name | Description | Required? | Default |
|--------------------|---|-----------|---------|
| :db | The database name | Required | None |
| :table | The new table name | Required | None |
| group | The user group to use when creating a table | Optional | None |
| permissions | The permissions string to use when creating a table. | Optional | None |
| external | Allows you to specify a location so that Hive does not use the default location for this table. | Optional | false |

| Name | Description | Required? | Default |
|------------------------|---|-----------|---------|
| ifNotExists | If true, you will not receive an error if the table already exists. | Optional | false |
| comment | Comment for the table | Optional | None |
| columns | A list of column descriptions, including name, type, and an optional comment. | Optional | None |
| partitionedBy | A list of column descriptions used to partition the table. Like the columns parameter this is a list of name, type, and comment fields. | Optional | None |
| clusteredBy | An object describing how to cluster the table including the parameters columnNames, sortedBy, numberOfBuckets. The sortedBy parameter includes the parameters columnName and order. For further information please refer to the examples below or to the Hive documentation . | Optional | None |
| format | Storage format description including parameters for rowFormat, storedAs and storedBy. For further information please refer to the examples below or to the Hive documentation . | Optional | None |
| location | The HDFS path | Optional | None |
| tableProperties | A list of table property names and values (key/value pairs) | Optional | None |

Results

| Name | Description |
|-----------------|--------------------|
| table | The new table name |
| database | The database name |

Example

Curl Command

```
% curl -s -X PUT -HContent-type:application/json -d '{
  "comment": "Best table made today",
  "columns": [
    { "name": "id", "type": "bigint" },
    { "name": "price", "type": "float", "comment": "The unit price" } ],
  "partitionedBy": [
    { "name": "country", "type": "string" } ],
  "format": { "storedAs": "rcfile" } }' \
'http://localhost:50111/templeton/v1/ddl/database/default/table/test_table?
user.name=ctdean'
```

Curl Command (using clusteredBy)

```
% curl -s -X PUT -HContent-type:application/json -d '{
  "comment": "Best table made today",
  "columns": [
    { "name": "id", "type": "bigint"},
    { "name": "price", "type": "float", "comment": "The unit price" } ],
  "partitionedBy": [
    { "name": "country", "type": "string" } ],
  "clusteredBy": {
    "columnNames": ["id"],
    "sortedBy": [
      { "columnName": "id", "order": "ASC" } ],
    "numberOfBuckets": 10 },
  "format": {
    "storedAs": "rcfile",
    "rowFormat": {
      "fieldsTerminatedBy": "\u0001",
      "serde": {
        "name": "org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe",
        "properties": {
          "key": "value" } } } }
  } ' \
'http://localhost:50111/templeton/v1/ddl/database/default/table/test_table_c?
user.name=ctdean'
```

JSON Output

```
{
  "table": "test_table",
```

```
"database": "default"
}
```

JSON Output (error)

```
{
  "statement": "use default; create table test_table_c(id bigint, price float comment ...",
  "error": "unable to create table: test_table_c",
  "exec": {
    "stdout": "",
    "stderr": "WARNING: org.apache.hadoop.metrics.jvm.EventCounter is deprecated...
      Hive history file=/tmp/ctdean/hive_job_log_ctdean_201204051335_2016086186.txt
      SLF4J: Class path contains multiple SLF4J bindings.
      SLF4J: Found binding in ...
      SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
      OK
      Time taken: 0.448 seconds
      FAILED: Error in semantic analysis: Operation not supported. HCatalog doesn't allow
      Clustered By in create table.
    ",
    "exitcode": 10
  }
}
```

1.4.5.10 POST ddl/database/:db/table/:table

Description

Rename an HCatalog table.

URL

<http://www.myserver.com/templeton/v1/ddl/database/:db/table/:table>

Parameters

| Name | Description | Required? | Default |
|--------------------|---|-----------|---------|
| :db | The database name | Required | None |
| :table | The existing (old) table name | Required | None |
| rename | The new table name | Required | None |
| group | The user group to use | Optional | None |
| permissions | The permissions string to use. The format is "rwxrw-r-x". | Optional | None |

Results

| Name | Description |
|-----------------|--------------------|
| table | The new table name |
| database | The database name |

Example**Curl Command**

```
% curl -s -d user.name=ctdean \
  -d rename=test_table_2 \
  'http://localhost:50111/templeton/v1/ddl/database/default/table/test_table'
```

JSON Output

```
{
  "table": "test_table_2",
  "database": "default"
}
```

JSON Output (error)

```
{
  "error": "Table test_table does not exist",
  "errorCode": 404,
  "database": "default",
  "table": "test_table_2"
}
```

1.4.5.11 DELETE ddl/database/:db/table/:table**Description**

Delete (drop) an HCatalog table.

URL

`http://www.myserver.com/templeton/v1/ddl/database/:db/table/:table`

Parameters

| Name | Description | Required? | Default |
|---------------|-------------------|-----------|---------|
| :db | The database name | Required | None |
| :table | The table name | Required | None |

| Name | Description | Required? | Default |
|--------------------|---|-----------|---------|
| ifExists | Hive 0.70 and later returns an error if the table specified does not exist, unless ifExists is set to true. | Optional | false |
| group | The user group to use | Optional | None |
| permissions | The permissions string to use. The format is "rwxrwx-r-x". | Optional | None |

Results

| Name | Description |
|-----------------|-------------------|
| table | The table name |
| database | The database name |

Example**Curl Command**

```
% curl -s -X DELETE 'http://localhost:50111/templeton/v1/ddl/database/default/table/test_table?user.name=ctdean'
```

JSON Output

```
{
  "table": "test_table",
  "database": "default"
}
```

1.4.5.12 PUT ddl/database/:db/table/:existingtable/like/:newtable**Description**

Create a new HCatalog table like an existing one

URL

```
http://www.myserver.com/templeton/v1/ddl/database/:db/
table/:existingtable/like/:newtable
```

Parameters

| Name | Description | Required? | Default |
|-----------------------|---|-----------|---------|
| :db | The database name | Required | None |
| :existingtable | The existing table name | Required | None |
| :newtable | The new table name. | Required | None |
| group | The user group to use when creating a table | Optional | None |
| permissions | The permissions string to use when creating a table. | Optional | None |
| external | Allows you to specify a location so that Hive does not use the default location for this table. | Optional | false |
| ifNotExists | If true, you will not receive an error if the table already exists. | Optional | false |
| location | The HDFS path | Optional | None |

Results

| Name | Description |
|-----------------|--------------------|
| table | The new table name |
| database | The database name |

Example

Curl Command

```
% curl -s -X PUT -HContent-type:application/json -d {} \
'http://localhost:50111/templeton/v1/ddl/database/default/table/test_table/like/
test_table_2?user.name=ctdean'
```

JSON Output

```
{
  "table": "test_table_2",
  "database": "default"
}
```

1.4.5.13 GET ddl/database/:db/table/:table/partition**Description**

List all the partitions in an HCatalog table.

URL

`http://www.myserver.com/templeton/v1/ddl/database/:db/table/:table/partition`

Parameters

| Name | Description | Required? | Default |
|---------------|-------------------|-----------|---------|
| :db | The database name | Required | None |
| :table | The table name | Required | None |

Results

| Name | Description |
|-------------------|---|
| partitions | A list of partition values and of partition names |
| database | The database name |
| table | The table name |

Example**Curl Command**

```
% curl -s 'http://localhost:50111/templeton/v1/ddl/database/default/table/my_table/partition?user.name=ctdean'
```

JSON Output

```
{
  "partitions": [
    {
      "values": [
        {
          "columnName": "dt",
          "columnValue": "20120101"
        },
        {
          "columnName": "country",
          "columnValue": "US"
        }
      ]
    }
  ],
}
```

```

    "name": "dt='20120101',country='US'"
  },
  "database": "default",
  "table": "my_table"
}

```

1.4.5.14 GET ddl/database/:db/table/:table/partition/:partition

Description

Describe a single partition in an HCatalog table.

URL

<http://www.myserver.com/templeton/v1/ddl/database/:db/table/:table/partition/:partition>

Parameters

| Name | Description | Required? | Default |
|-------------------|--|-----------|---------|
| :db | The database name | Required | None |
| :table | The table name | Required | None |
| :partition | The partition name, col_name='value' list. Be careful to properly encode the quote for http, for example, country=%27algeria%27. | Required | None |

Results

| Name | Description |
|---------------------|---|
| database | The database name |
| table | The table name |
| partition | The partition name |
| partitioned | True if the table is partitioned |
| location | Location of table |
| outputFormat | Output format |
| columns | list of column names, types, and comments |

| Name | Description |
|-------------------------|-------------------------------|
| owner | The owner's user name |
| partitionColumns | List of the partition columns |
| inputFormat | Input format |

Example

Curl Command

```
% curl -s \
  'http://localhost:50111/templeton/v1/ddl/database/default/table/mytest/partition/
country=%27US%27?user.name=ctdean'
```

JSON Output

```
{
  "partitioned": true,
  "location": "hdfs://ip-10-77-6-151.ec2.internal:8020/apps/hive/warehouse/mytest/loc1",
  "outputFormat": "org.apache.hadoop.hive ql.io.RCFileOutputFormat",
  "columns": [
    {
      "name": "i",
      "type": "int"
    },
    {
      "name": "j",
      "type": "bigint"
    },
    {
      "name": "ip",
      "comment": "IP Address of the User",
      "type": "string"
    }
  ],
  "owner": "rachel",
  "partitionColumns": [
    {
      "name": "country",
      "type": "string"
    }
  ],
  "inputFormat": "org.apache.hadoop.hive ql.io.RCFileInputFormat",
  "database": "default",
  "table": "mytest",
  "partition": "country='US'"
}
```

1.4.5.15 PUT ddl/database/:db/table/:table/partition/:partition**Description**

Create a partition in an HCatalog table.

URL

`http://www.myserver.com/templeton/v1/ddl/database/:db/table/:table/partition/:partition`

Parameters

| Name | Description | Required? | Default |
|--------------------|--|-----------|---------|
| :db | The database name | Required | None |
| :table | The table name | Required | None |
| :partition | The partition name, col_name='value' list. Be careful to properly encode the quote for http, for example, country=%27algeria%27. | Required | None |
| group | The user group to use | Optional | None |
| permissions | The permissions string to use | Optional | None |
| location | The location for partition creation | Required | None |
| ifNotExists | If true, return an error if the partition already exists. | Optional | False |

Results

| Name | Description |
|------------------|--------------------|
| partition | The partition name |
| table | The table name |
| database | The database name |

Example**Curl Command**

```
% curl -s -X PUT -HContent-type:application/json -d '{"location": "loc_a"}' \
'http://localhost:50111/templeton/v1/ddl/database/default/table/test_table/
partition/country=%27algeria%27?user.name=ctdean'
```

JSON Output

```
{
  "partition": "country='algeria'",
  "table": "test_table",
  "database": "default"
}
```

1.4.5.16 DELETE ddl/database/:db/table/:table/partition/:partition**Description**

Delete (drop) a partition in an HCatalog table.

URL

`http://www.myserver.com/templeton/v1/ddl/database/:db/table/:table/partition/:partition`

Parameters

| Name | Description | Required? | Default |
|-------------------|--|-----------|---------|
| :db | The database name | Required | None |
| :table | The table name | Required | None |
| :partition | The partition name, col_name='value' list. Be careful to properly encode the quote for http, for example, country=%27algeria%27. | Required | None |
| ifExists | Hive returns an error if the partition specified does not exist, unless ifExists is set to true. | Optional | false |
| group | The user group to use | Optional | None |

| Name | Description | Required? | Default |
|--------------------|--|-----------|---------|
| permissions | The permissions string to use. The format is "rwxrwx-r-x". | Optional | None |

Results

| Name | Description |
|------------------|--------------------|
| partition | The partition name |
| table | The table name |
| database | The database name |

Example

Curl Command

```
% curl -s -X DELETE \
    'http://localhost:50111/templeton/v1/ddl/database/default/table/test_table/
partition/country=%27algeria%27?user.name=ctdean'
```

JSON Output

```
{
  "partition": "country='algeria'",
  "table": "test_table",
  "database": "default"
}
```

1.4.5.17 GET ddl/database/:db/table/:table/column

Description

List the columns in an HCatalog table.

URL

`http://www.myserver.com/templeton/v1/ddl/database/:db/table/:table/column`

Parameters

| Name | Description | Required? | Default |
|------------|-------------------|-----------|---------|
| :db | The database name | Required | None |

| Name | Description | Required? | Default |
|---------------|----------------|-----------|---------|
| :table | The table name | Required | None |

Results

| Name | Description |
|-----------------|----------------------------------|
| columns | A list of column names and types |
| database | The database name |
| table | The table name |

Example**Curl Command**

```
% curl -s 'http://localhost:50111/templeton/v1/ddl/database/default/table/my_table/column?
user.name=ctdean'
```

JSON Output

```
{
  "columns": [
    {
      "name": "id",
      "type": "bigint"
    },
    {
      "name": "user",
      "comment": "The user name",
      "type": "string"
    },
    {
      "name": "my_p",
      "type": "string"
    },
    {
      "name": "my_q",
      "type": "string"
    }
  ],
  "database": "default",
  "table": "my_table"
}
```

1.4.5.18 GET ddl/database/:db/table/:table/column/:column**Description**

Describe a single column in an HCatalog table.

URL

`http://www.myserver.com/templeton/v1/ddl/database/:db/table/:table/column/:column`

Parameters

| Name | Description | Required? | Default |
|----------------|-------------------|-----------|---------|
| :db | The database name | Required | None |
| :table | The table name | Required | None |
| :column | The column name | Required | None |

Results

| Name | Description |
|-----------------|--|
| database | The database name |
| table | The table name |
| column | A JSON object containing the column name, type, and comment (if any) |

Example**Curl Command**

```
% curl -s 'http://localhost:50111/templeton/v1/ddl/database/default/table/test_table/column/price?user.name=ctdean'
```

JSON Output

```
{
  "database": "default",
  "table": "test_table",
  "column": {
    "name": "price",
    "comment": "The unit price",
    "type": "float"
  }
}
```

1.4.5.19 PUT ddl/database/:db/table/:table/column/:column**Description**

Create a column in an HCatalog table.

URL

`http://www.myserver.com/templeton/v1/ddl/database/:db/table/:table/column/:column`

Parameters

| Name | Description | Required? | Default |
|--------------------|---|-----------|---------|
| :db | The database name | Required | None |
| :table | The table name | Required | None |
| :column | The column name | Required | None |
| group | The user group to use | Optional | None |
| permissions | The permissions string to use | Optional | None |
| type | The type of column to add, like "string" or "int" | Required | None |
| comment | The column comment, like a description | Optional | None |

Results

| Name | Description |
|-----------------|-------------------|
| column | The column name |
| table | The table name |
| database | The database name |

Example

Curl Command

```
% curl -s -X PUT -HContent-type:application/json \
  -d '{"type": "string", "comment": "The brand name"}' \
  'http://localhost:50111/templeton/v1/ddl/database/default/table/test_table/column/brand?user.name=ctdean'
```

JSON Output

```
{
  "column": "brand",
  "table": "test_table",
```

```
"database": "default"
}
```

1.4.5.20 GET ddl/database/:db/table/:table/property

Description

List all the properties of an HCatalog table.

URL

`http://www.myserver.com/templeton/v1/ddl/database/:db/table/:table/property`

Parameters

| Name | Description | Required? | Default |
|---------------|-------------------|-----------|---------|
| :db | The database name | Required | None |
| :table | The table name | Required | None |

Results

| Name | Description |
|-------------------|--|
| properties | A list of the tables properties in name: value pairs |
| database | The database name |
| table | The table name |

Example

Curl Command

```
% curl -s 'http://localhost:50111/templeton/v1/ddl/database/default/table/test_table/property?user.name=ctdean'
```

JSON Output

```
{
  "properties": {
    "fruit": "apple",
    "last_modified_by": "ctdean",
    "hcat.osd": "org.apache.hcatalog.rcfile.RCFileOutputDriver",
    "color": "blue",
    "last_modified_time": "1331620706",
    "hcat.isd": "org.apache.hcatalog.rcfile.RCFileInputDriver",
    "transient_lastDdlTime": "1331620706",
```



```
    "comment": "Best table made today",
    "country": "Albania"
  },
  "table": "test_table",
  "database": "default"
}
```

1.4.5.21 GET ddl/database/:db/table/:table/property/:property

Description

Return the value of a single table property.

URL

`http://www.myserver.com/templeton/v1/ddl/database/:db/table/:table/property/:property`

Parameters

| Name | Description | Required? | Default |
|------------------|-------------------|-----------|---------|
| :db | The database name | Required | None |
| :table | The table name | Required | None |
| :property | The property name | Required | None |

Results

| Name | Description |
|-----------------|---|
| property | The requested property's name: value pair |
| database | The database name |
| table | The table name |

Example

Curl Command

```
% curl -s 'http://localhost:50111/templeton/v1/ddl/database/default/table/test_table/property/fruit?user.name=ctdean'
```

JSON Output

```
{
  "property": {
    "fruit": "apple"
  }
}
```

```

},
"table": "test_table",
"database": "default"
}

```

JSON Output (error)

```

{
  "error": "Table test_table does not exist",
  "errorCode": 404,
  "database": "default",
  "table": "test_table"
}

```

1.4.5.22 PUT ddl/database/:db/table/:table/property/:property

Description

Add a single property on an HCatalog table. This will also reset and existing property.

URL

`http://www.myserver.com/templeton/v1/ddl/database/:db/table/:table/property/:property`

Parameters

| Name | Description | Required? | Default |
|--------------------|-------------------------------|-----------|---------|
| :db | The database name | Required | None |
| :table | The table name | Required | None |
| :property | The property name | Required | None |
| group | The user group to use | Optional | None |
| permissions | The permissions string to use | Optional | None |
| value | The property value | Required | None |

Results

| Name | Description |
|-----------------|-------------------|
| database | The database name |
| table | The table name |
| property | The property name |

Example**Curl Command**

```
% curl -s -X PUT -HContent-type:application/json -d '{ "value": "apples" }' \
'http://localhost:50111/templeton/v1/ddl/database/default/table/test_table/property/
fruit?user.name=ctdean'
```

JSON Output

```
{
  "property": "fruit",
  "table": "test_table",
  "database": "default"
}
```

1.4.6 POST mapreduce/streaming**1.4.6.1 Description**

Create and queue an [Hadoop streaming MapReduce](#) job.

1.4.6.2 URL

`http://www.myserver.com/templeton/v1/mapreduce/streaming`

1.4.6.3 Parameters

| Name | Description | Required? | Default |
|----------------|--|-----------|-----------------|
| input | Location of the input data in Hadoop. | Required | None |
| output | Location in which to store the output data. If not specified, Templeton will store the output in a location that can be discovered using the queue resource. | Optional | See description |
| mapper | Location of the mapper program in Hadoop. | Required | None |
| reducer | Location of the reducer program in Hadoop. | Required | None |
| file | Add an HDFS file to the distributed cache. | Optional | None |

| Name | Description | Required? | Default |
|------------------|---|-----------|---------|
| define | Set an Hadoop configuration variable using the syntax <code>define=NAME=VALUE</code> | Optional | None |
| cmdenv | Set an environment variable using the syntax <code>cmdenv=NAME=VALUE</code> | Optional | None |
| arg | Set a program argument. | Optional | None |
| statusdir | A directory where Templeton will write the status of the Map Reduce job. If provided, it is the caller's responsibility to remove this directory when done. | Optional | None |
| callback | Define a URL to be called upon job completion. You may embed a specific job ID into this URL using <code>\$jobId</code> . This tag will be replaced in the callback URL with this job's job ID. | Optional | None |

1.4.6.4 Results

| Name | Description |
|-------------|--|
| id | A string containing the job ID similar to "job_201110132141_0001". |
| info | A JSON object containing the information returned when the job was queued. See the Hadoop documentation (Class TaskController) for more information. |

1.4.6.5 Example

Code and Data Setup

```
% cat mydata/file01 mydata/file02
```

```

Hello World Bye World
Hello Hadoop Goodbye Hadoop

% hadoop fs -put mydata/ .

% hadoop fs -ls mydata
Found 2 items
-rw-r--r--  1 ctdean supergroup          23 2011-11-11 13:29 /user/ctdean/mydata/file01
-rw-r--r--  1 ctdean supergroup          28 2011-11-11 13:29 /user/ctdean/mydata/file02

```

Curl Command

```

% curl -s -d user.name=ctdean \
      -d input=mydata \
      -d output=mycounts \
      -d mapper=/bin/cat \
      -d reducer="/usr/bin/wc -w" \
      'http://localhost:50111/templeton/v1/mapreduce/streaming'

```

JSON Output

```

{
  "id": "job_201111111311_0008",
  "info": {
    "stdout": "packageJobJar: [] [/Users/ctdean/var/hadoop/hadoop-0.20.205.0/share/
hadoop/contrib/streaming/hadoop-streaming-0.20.205.0.jar...
               templeton-job-id:job_201111111311_0008
               ",
    "stderr": "11/11/11 13:26:43 WARN mapred.JobClient: Use GenericOptionsParser for
parsing the arguments
               11/11/11 13:26:43 INFO mapred.FileInputFormat: Total input paths to
process : 2
               ",
    "exitcode": 0
  }
}

```

Results

```

% hadoop fs -ls mycounts
Found 3 items
-rw-r--r--  1 ctdean supergroup           0 2011-11-11 13:27 /user/ctdean/mycounts/_SUCCESS
drwxr-xr-x  - ctdean supergroup           0 2011-11-11 13:26 /user/ctdean/mycounts/_logs
-rw-r--r--  1 ctdean supergroup          10 2011-11-11 13:27 /user/ctdean/mycounts/
part-00000

% hadoop fs -cat mycounts/part-00000
8

```

1.4.7 POST mapreduce/jar

1.4.7.1 Description

Creates and queues a standard [Hadoop MapReduce](#) job.

1.4.7.2 URL

`http://www.myserver.com/templeton/v1/mapreduce/jar`

1.4.7.3 Parameters

| Name | Description | Required? | Default |
|------------------|---|-----------|---------|
| jar | Name of the jar file for Map Reduce to use. | Required | None |
| class | Name of the class for Map Reduce to use. | Required | None |
| libjars | Comma separated jar files to include in the classpath. | Optional | None |
| files | Comma separated files to be copied to the map reduce cluster | Optional | None |
| arg | Set a program argument. | Optional | None |
| define | Set an Hadoop configuration variable using the syntax <code>define=NAME=VALUE</code> | Optional | None |
| statusdir | A directory where Templeton will write the status of the Map Reduce job. If provided, it is the caller's responsibility to remove this directory when done. | Optional | None |
| callback | Define a URL to be called upon job completion. You may embed a specific job ID into this URL using <code>\$jobId</code> . This tag will be replaced in the callback | Optional | None |

| Name | Description | Required? | Default |
|------|-----------------------------|-----------|---------|
| | URL with this job's job ID. | | |

1.4.7.4 Results

| Name | Description |
|-------------|--|
| id | A string containing the job ID similar to "job_201110132141_0001". |
| info | A JSON object containing the information returned when the job was queued. See the Hadoop documentation (Class TaskController) for more information. |

1.4.7.5 Example

Code and Data Setup

```
% hadoop fs -put wordcount.jar .
% hadoop fs -put transform.jar .

% hadoop fs -ls .
Found 2 items
-rw-r--r--  1 ctdean supergroup      23 2011-11-11 13:29 /user/ctdean/wordcount.jar
-rw-r--r--  1 ctdean supergroup      28 2011-11-11 13:29 /user/ctdean/transform.jar
```

Curl Command

```
% curl -s -d user.name=ctdean \
-d jar=wordcount.jar \
-d class=org.myorg.WordCount \
-d libjars=transform.jar \
-d arg=wordcount/input \
-d arg=wordcount/output \
'http://localhost:50111/templeton/v1/mapreduce/jar'
```

JSON Output

```
{
  "id": "job_201111121211_0001",
  "info": {
    "stdout": "templeton-job-id: job_201111121211_0001",
    "stderr": "",
    "exitcode": 0
  }
}
```

1.4.8 POST pig

1.4.8.1 Description

Create and queue a [Pig](#) job.

1.4.8.2 URL

`http://www.myserver.com/templeton/v1/pig`

1.4.8.3 Parameters

| Name | Description | Required? | Default |
|------------------|---|---|---------|
| execute | String containing an entire, short pig program to run. | One of either "execute" or "file" is required | None |
| file | HDFS file name of a pig program to run. | One of either "exec" or "file" is required | None |
| arg | Set a program argument. | Optional | None |
| files | Comma separated files to be copied to the map reduce cluster | Optional | None |
| statusdir | A directory where Templeton will write the status of the Pig job. If provided, it is the caller's responsibility to remove this directory when done. | Optional | None |
| callback | Define a URL to be called upon job completion. You may embed a specific job ID into this URL using <code>\$jobId</code> . This tag will be replaced in the callback URL with this job's job ID. | Optional | None |

1.4.8.4 Results

| Name | Description |
|-------------|--|
| id | A string containing the job ID similar to "job_201110132141_0001". |
| info | A JSON object containing the information returned when the job was queued. See the Hadoop documentation (Class TaskController) for more information. |

1.4.8.5 Example

Code and Data Setup

```
% cat id.pig
A = load 'passwd' using PigStorage(':');
B = foreach A generate $0 as id;
dump B;

% cat fake-passwd
ctdean:Chris Dean:secret
pauls:Paul Stolorz:good
carmas:Carlos Armas:evil
dra:Deirdre McClure:marvelous

% hadoop fs -put id.pig .
% hadoop fs -put fake-passwd passwd
```

Curl Command

```
% curl -s -d user.name=ctdean \
      -d file=id.pig \
      -d arg=-v \
      'http://localhost:50111/templeton/v1/pig'
```

JSON Output

```
{
  "id": "job_201111101627_0018",
  "info": {
    "stdout": "templeton-job-id:job_201111101627_0018",
    "stderr": "",
    "exitcode": 0
  }
}
```

1.4.9 POST hive

1.4.9.1 Description

Runs a [Hive](#) query or set of commands.

1.4.9.2 URL

`http://www.myserver.com/templeton/v1/hive`

1.4.9.3 Parameters

| Name | Description | Required? | Default |
|------------------|---|---|---------|
| execute | String containing an entire, short hive program to run. | One of either "execute" or "file" is required | None |
| file | HDFS file name of a hive program to run. | One of either "exec" or "file" is required | None |
| define | Set a Hive configuration variable using the syntax <code>define=NAME=VALUE</code> . | Optional | None |
| statusdir | A directory where Templeton will write the status of the Hive job. If provided, it is the caller's responsibility to remove this directory when done. | Optional | None |
| callback | Define a URL to be called upon job completion. You may embed a specific job ID into this URL using <code>\$jobId</code> . This tag will be replaced in the callback URL with this job's job ID. | Optional | None |

1.4.9.4 Results

| Name | Description |
|-----------|--|
| id | A string containing the job ID similar to "job_201110132141_0001". |

| Name | Description |
|-------------|--|
| info | A JSON object containing the information returned when the job was queued. See the Hadoop documentation (Class TaskController) for more information. |

1.4.9.5 Example

Curl Command

```
% curl -s -d user.name=ctdean \
-d execute="select**+from+pokes;" \
-d statusdir="pokes.output" \
'http://localhost:50111/templeton/v1/hive'
```

JSON Output

```
{
  "id": "job_2011111111311_0005",
  "info": {
    "stdout": "templeton-job-id:job_2011111111311_0005",
    "stderr": "",
    "exitcode": 0
  }
}
```

Results

```
% hadoop fs -ls pokes.output
Found 2 items
-rw-r--r--  1 ctdean supergroup      610 2011-11-11 13:22 /user/ctdean/pokes.output/
stderr
-rw-r--r--  1 ctdean supergroup      15 2011-11-11 13:22 /user/ctdean/pokes.output/
stdout

% hadoop fs -cat pokes.output/stdout
1      a
2      bb
3      ccc
```

1.4.10 GET queue

1.4.10.1 Description

Return a list of all job IDs registered to the user.

1.4.10.2 URL

`http://www.myserver.com/templeton/v1/queue`

1.4.10.3 Parameters

Only the [standard parameters](#) are accepted.

1.4.10.4 Results

| Name | Description |
|------------|---|
| ids | A list of all job IDs registered to the user. |

1.4.10.5 Example

Curl Command

```
% curl -s 'http://localhost:50111/templeton/v1/queue?user.name=ctdean'
```

JSON Output

```
{
  "job_201111111311_0008",
  "job_201111111311_0012"
}
```

1.4.11 GET queue/:jobid

1.4.11.1 Description

Check the status of a job and get related job information given its job ID. Substitute ":jobid" with the job ID received when the job was created.

1.4.11.2 URL

`http://www.myserver.com/templeton/v1/queue/:jobid`

1.4.11.3 Parameters

| Name | Description | Required? | Default |
|---------------|--|-----------|---------|
| :jobid | The job ID to check. This is the ID received when the job was created. | Required | None |

1.4.11.4 Results

| Name | Description |
|------------------------|---|
| status | A JSON object containing the job status information. See the Hadoop documentation (Class JobStatus) for more information. |
| profile | A JSON object containing the job profile information. See the Hadoop documentation (Class JobProfile) for more information. |
| id | The job ID. |
| parentId | The parent job ID. |
| percentComplete | The job completion percentage, for example "75% complete". |
| exitValue | The job's exit value. |
| user | User name of the job creator. |
| callback | The callback URL, if any. |
| completed | A string representing completed status, for example "done". |

1.4.11.5 Example

Curl Command

```
% curl -s 'http://localhost:50111/templeton/v1/queue/job_201112212038_0003?
user.name=ctdean'
```

JSON Output

```
{
  "status": {
    "startTime": 1324529476131,
    "username": "ctdean",
    "jobID": {
      "jtIdentifier": "201112212038",
      "id": 4
    },
    "jobACLs": {
    },
    "schedulingInfo": "NA",
    "failureInfo": "NA",
    "jobId": "job_201112212038_0004",
    "jobPriority": "NORMAL",
    "runState": 2,
  }
}
```

```

        "jobComplete": true
    },
    "profile": {
        "url": "http://localhost:50030/jobdetails.jsp?jobid=job_201112212038_0004",
        "jobID": {
            "jtIdentifier": "201112212038",
            "id": 4
        },
        "user": "ctdean",
        "queueName": "default",
        "jobFile": "hdfs://localhost:9000/tmp/hadoop-ctdean/mapred/staging/
ctdean/.staging/job_201112212038_0004/job.xml",
        "jobName": "PigLatin:DefaultJobName",
        "jobId": "job_201112212038_0004"
    },
    "id": "job_201112212038_0004",
    "parentId": "job_201112212038_0003",
    "percentComplete": "100% complete",
    "exitValue": 0,
    "user": "ctdean",
    "callback": null,
    "completed": "done"
}

```

1.4.12 DELETE queue/:jobid

1.4.12.1 Description

Kill a job given its job ID. Substitute ":jobid" with the job ID received when the job was created.

1.4.12.2 URL

`http://www.myserver.com/templeton/v1/queue/:jobid`

1.4.12.3 Parameters

| Name | Description | Required? | Default |
|---------------|---|-----------|---------|
| :jobid | The job ID to delete. This is the ID received when the job was created. | Required | None |

1.4.12.4 Results

| Name | Description |
|---------------|---|
| status | A JSON object containing the job status information. See the Hadoop documentation (Class JobStatus) for more information. |

| Name | Description |
|------------------------|---|
| profile | A JSON object containing the job profile information. See the Hadoop documentation (Class JobProfile) for more information. |
| id | The job ID. |
| parentId | The parent job ID. |
| percentComplete | The job completion percentage, for example "75% complete". |
| exitValue | The job's exit value. |
| user | User name of the job creator. |
| callback | The callback URL, if any. |
| completed | A string representing completed status, for example "done". |

1.4.12.5 Example

Curl Command

```
% curl -s -X DELETE 'http://localhost:50111/templeton/v1/queue/job_201111111311_0009?
user.name=ctdean'
```

JSON Output

```
{
  "status": {
    "startTime": 1321047216471,
    "username": "ctdean",
    "jobID": {
      "jtIdentifier": "201111111311",
      "id": 9
    },
    "jobACLs": {
    },
    "schedulingInfo": "NA",
    "failureInfo": "NA",
    "jobId": "job_201111111311_0009",
    "jobPriority": "NORMAL",
    "runState": 1,
    "jobComplete": false
  },
  "profile": {
    "url": "http://localhost:50030/jobdetails.jsp?jobid=job_201111111311_0009",
    "user": "ctdean",
    "jobID": {
```

```

        "jtIdentifier": "201111111311",
        "id": 9
      },
      "queueName": "default",
      "jobFile": "hdfs://localhost:9000/tmp/hadoop-ctdean/mapred/staging/
ctdean/.staging/job_201111111311_0009/job.xml",
      "jobName": "streamjob3322518350676530377.jar",
      "jobId": "job_201111111311_0009"
    }
    "id": "job_201111111311_0009",
    "parentId": "job_201111111311_0008",
    "percentComplete": "10% complete",
    "exitValue": 0,
    "user": "ctdean",
    "callback": null,
    "completed": "false"
  }
}

```

Note: The job is not immediately deleted, therefore the information returned may not reflect deletion, as in our example. Use [GET queue/:jobid](#) to monitor the job and confirm that it is eventually deleted.

1.5 PDF