Mesos Executor HTTP API design documentation

Terminology/Motivation

Much of the terminology/motivation in this document has been borrowed over from the Mesos HTTP API design document.

Unacknowledged Status Updates: An executor is expected to keep running when disconnected from the agent. The disconnection can happen due to an agent process failure. The status update messages produced by the executor during the time it was disconnected are known as unacknowledged status updates.

Unacknowledged Tasks: An agent process can die after sending its intent to launch a task to the executor or the message can be lost due to network intermediaries. An executor is expected to maintain a list of tasks for which it has not yet generated any status updates. This helps the agent in marking the tasks as lost when the executor subscribes with the agent again after a disconnection.

Solution Overview

Following the design of the new scheduler HTTP API, the executor now interacts with Mesos via `"api/v1/executor"` endpoint hosted by the Mesos agent. The fully qualified URL of the endpoint is:

http://agent:5051/api/v1/executor/

This endpoint accepts **HTTP POST** requests with data encoded in JSON (**Content-Type: application/json**) or binary Protobuf (**Content-Type: application/x-protobuf**).
The first request that an executor sends to "/api/v1/executor" endpoint is called **SUBSCRIBE** and results in a streaming response. The agent keeps the subscription connection open (barring errors in network, software, hardware etc) and streams the events via HTTP response ("**200 OK**" status code) using chunked encoding (**Transfer-Encoding: chunked**). Executors are expected to keep the subscription connection open as long as possible (barring errors in network, software, hardware etc.) and incrementally process the response (NOTE: HTTP client libraries that can only parse the response after the connection is closed cannot be used). For the encoding used, please refer to **"Events"** section below.

All the subsequent (non subscribe) requests to "/api/v1/executor" endpoint (see details below in **Calls** section) ***must*** be sent using a different connection(s) than the one being used for subscription. Agent responds to these HTTP POST requests with "**202 Accepted**" status codes (or, for unsuccessful requests, with 4xx or 5xx status codes; details in later sections). In line with the HTTP standard, a "202 Accepted" response just means that a request has been accepted for processing, ***not*** that the processing of the request has been completed. The request might or might not be acted upon by Mesos (e.g., agent fails during the processing of the request). Any asynchronous responses to these requests will be streamed on long-lived subscription connection.

Summary of Messages

Calls sent by Executor to Agent

SUBSCRIBE	Mandatory initial request; opens a long-lived streaming connection
UPDATE	Notifies the scheduler that a task has transitioned from one state to another
MESSAGE	Arbitrary binary message proxied by Master to `Scheduler`

Events sent by Agent to Executor (via streaming connection)

SUBSCRIBED	Received when an executor is successfully subscribed
LAUNCH	Received when the scheduler attempts to launch a task
KILL	Received when the scheduler wants to kill a specific task
ACKNOWLEDGED	Received when a status update is acknowledged
MESSAGE	Arbitrary binary message from `Executor`, proxied back to `Scheduler`
ERROR	Any asynchronous error events generated by the agent
SHUTDOWN	Received when the scheduler/agent asks the executor to shutdown/kill itself

Calls

The following calls would be accepted by Mesos Agent. The canonical source of this information is include/mesos/v1/executor/executor.proto (NOTE: The exact protobuf definitions are subject to change before this doc is finalized).
Calls made without subscription result in a `` `403 Forbidden` `` response instead of the `` `202 Accepted` `` response.

SUBSCRIBE

This is the very first step in the communication process between the executor and the agent. This is also to be considered as **subscription** to the "`/api/v1/executor`" events stream.

To subscribe to the agent, the executor sends an HTTP POST request with an encoded SUBSCRIBE message with the required ExecutorID, FrameworkID fields. These values are set based on the environment variables *MESOS_EXECUTOR_ID*, *MESOS_FRAMEWORK_ID* provided by the agent when the executor is launched.

The HTTP response in this case is a stream with RecordIO encoding, with the first event being SUBSCRIBED event (see details in **Events** section).

Subscription Request (JSON)

```
POST /api/v1/executor HTTP/1.1
Host: agent0:5051
Content-Type: application/json
Accept: application/json
Connection: keep-alive
```

```
{
  "type" : "SUBSCRIBE",
  "executor_id" :  {"value" : "387aa966-8fc5-4428-a794-5a868a60d3eb"},
  "framework_id" : {"value" : "49154f1b-8cf6-4421-bf13-8bd11dccd1f1"},
  “subscribe” : {}
}
```

Subscription Response (JSON)

```
HTTP/1.1 200 OK
Content-Type: application/json
Transfer-Encoding: chunked
Connection: keep-alive

<event-length>
{
  "type" : "SUBSCRIBED",

  "subscribed" : {
    "executor_info" : {
      "executor_id" : {"value" : "387aa966-8fc5-4428-a794-5a868a60d3eb"},
      "command" : {
        "value" : "/path/to/executor"
      }
    },
    "framework_id" : {"value" : "49154f1b-8cf6-4421-bf13-8bd11dccd1f1"},
    "framework_info" : {
```

```
      "user" : "foo",
      "name" : "my_framework"
    },
    "agent_id" : {"value" : "f1c9cdc5-195e-41a7-a0d7-adaa9af07f81"},
    "agent_info" : {
      "host" : "agent0",
      "port" : 5051
    }
  }
}
<more events>
```

Alternatively, if the executor is connecting to the agent after a disconnection, it can also send a list of unacknowledged tasks and status updates.

Subscription Request (JSON)

```
POST /api/v1/executor HTTP/1.1
Host: agent0:5051
Content-Type: application/json
Accept: application/json
Connection: keep-alive

{
  "type" : "SUBSCRIBE",
  "executor_id" :  {"value" : "387aa966-8fc5-4428-a794-5a868a60d3eb"},
  "framework_id" : {"value" : "49154f1b-8cf6-4421-bf13-8bd11dccd1f1"},
  “subscribe” : {
    "tasks" : [
      {
        "name" : "dummy-task",
        "task_id" : {"value" : "d40f3f3e-bbe3-44af-a230-4cb1eae72f67"},
        "agent_id" : {"value" : "f1c9cdc5-195e-41a7-a0d7-adaa9af07f81"},
        "command" : {
        "value" : "ls",
        "arguments" : ["-l", "/tmp"]
      }
    ],
    "updates" : [
      {
        "framework_id" : {"value" : "49154f1b-8cf6-4421-bf13-8bd11dccd1f1"},
        "status" : {
          "source" : "EXECUTOR",
          "task_id" : {"value" : "d40f3f3e-bbe3-44af-a230-4cb1eae72f67"},
          "state" : "TASK_RUNNING"
        },
        "timestamp" : 12345.543,
        "uuid" : "ZDQwZjNmM2UtYmJlMy00NGFmLWEyMzAtNGNiMWVhZTcyZjY3Cg=="
      }
    ]
  },
```

```
}
```

Subscription Response (JSON)

```
HTTP/1.1 200 OK
Content-Type: application/json
Transfer-Encoding: chunked
Connection: keep-alive

<event-length>
{
  "type" : "SUBSCRIBED",

  "subscribed" : {
    "executor_info" : {
      "executor_id" : {"value" : "387aa966-8fc5-4428-a794-5a868a60d3eb"},
      "command" : {
        "value" : "/path/to/executor"
      }
    },
    "framework_id" : {"value" : "49154f1b-8cf6-4421-bf13-8bd11dccd1f1"},
    "framework_info" : {
      "user" : "foo",
      "name" : "my_framework"
    },
    "agent_id" : {"value" : "f1c9cdc5-195e-41a7-a0d7-adaa9af07f81"},
    "agent_info" : {
      "host" : "agent0",
      "port" : 5051
    }
  }
}
<more events>
```

If subscription fails for whatever reason, a HTTP 4xx response is returned with the error message as part of the body and the connection is closed.

If the persistent HTTP connection is closed for any reason, except after a `SHUTDOWN` event, the executor should prepare for agent recovery. To do so, it should attempt to reconnect with the agent using a Subscribe request. The reconnection retry mechanisms depend on whether framework has ‘checkpointing’ enabled.

- If checkpointing is enabled, executors retry subscription for ‘MESOS_RECOVERY_TIMEOUT’ seconds (environment variable) set by the agent. If the executor is unable to reconnect with the agent within this time, it should gracefully shutdown. It is recommended to attempt reconnection for a finite number of times using an exponential backoff strategy.

- If checkpointing is not enabled, executors are encouraged to gracefully shutdown as soon as they detect a disconnection. This is similar to the existing semantics for the old ExecutorDriver implementation.

Note that the older version of the API supported bidirectional communication which allowed the agent to notify its executors once it was back on-line. With the new client–server approach this is no longer possible, and the executor is responsible now to detect agent failures and attempt reconnection.

UPDATE

Sent by the executor to reliable communicate the state of the managed tasks. It is crucial that a terminal update (e.g., `TASK_FINISHED`, `TASK_KILLED` or `TASK_FAILED`) is sent to the scheduler as soon as the task terminates, in order to allow Mesos to release the resources allocated to the task.

The scheduler must explicitly respond to this call through an `ACKNOWLEDGE` message (see the `ACKNOWLEDGE` event). The executor must keep a list of unacknowledged updates. The executor should resend the unacknowledged messages until they are acknowledged by the agent. If there's a communication failure, these messages must be sent in the subscribe request in the `updates` field.

Note: Sending a `TASK_STAGING` is invalid behavior and in the old API this generated an error in the log followed by the executor shutdown. With the HTTP API sending an invalid status update `TASK_STAGING` will result in a 400 Bad Request.

UPDATE request (JSON)

```
POST /executor/call HTTP/1.1
Host: agent0:5051
Content-Type: application/json

{
  "type" : "UPDATE",
   "framework_id" : {"value" : "49154f1b-8cf6-4421-bf13-8bd11dccd1f1"},
   "executor_id" : {"value" : "387aa966-8fc5-4428-a794-5a868a60d3eb"},
    "update" : {
      "status" : {
        "source" : "SOURCE_EXECUTOR",
        "task_id" : {"value" : "d40f3f3e-bbe3-44af-a230-4cb1eae72f67"},
        "state" : "TASK_RUNNING"
      },
      "timestamp" : 12345.543,
      "uuid" : "ZDQwZjNmM2UtYmJlMy00NGFmLWEyMzAtNGNiMWVhZTcyZjY3Cg=="
   }
```

```
}
```

UPDATE response

```
HTTP/1.1 202 Accepted
```

MESSAGE

Sent by the executor in order to deliver arbitrary binary data to the scheduler. Note that Mesos neither interprets this data nor it makes any guarantees about the delivery of this message to the scheduler. This can be used by frameworks to exchange data without the overhead of reliability.

MESSAGE request (JSON)

```
POST /executor/call HTTP/1.1
Host:  agent0:5051
Content-Type: application/json

{
  "type" : "MESSAGE",
  "framework_id" : {"value" : "49154f1b-8cf6-4421-bf13-8bd11dccd1f1"},
  "executor_id" : {"value" : "387aa966-8fc5-4428-a794-5a868a60d3eb"},
  "message" : {
    "agent_id" : {"value" : "f1c9cdc5-195e-41a7-a0d7-adaa9af07f81"},
    "data" : "VGhpcyBpcyBhIHVucmVsaWFibGUgc2VudCBtZXNzYWdlCg=="
  }
}
```

MESSAGE response

```
HTTP/1.1 202 Accepted
```

Events

The executor is expected to keep a persistent connection open to the `/api/v1/executor` endpoint, even after getting a `SUBSCRIBED` HTTP response event chunk. This is indicated by the `Connection: keep-alive` and `Transfer-Encoding: chunked` headers, and by **not** setting the `Content-Length` header. All subsequent events that are relevant to this executor generated by Mesos are streamed on this connection. Agent encodes each Event in RecordIO format, i.e., string representation of length of the event followed by JSON or binary Protobuf (possibly compressed) encoded event. Note that the value of length will never be '0' and the size of the length will be the size of unsigned integer (i.e., 64 bits). Also note that the RecordIO encoding should be decoded by the executor whereas the underlying HTTP chunked

encoding is typically invisible at the application (executor) layer. The type of content encoding used for the events will be determined by the accept header of the POST request (e.g., **Accept: application/json**).

LAUNCH

Sent by the framework whenever it needs to assign a new task to the executor. The executor is required to send an `UPDATE` response back to the agent indicating the success or failure of the task initialization. Note that the `id` field of the `framework_info` must always be set.

LAUNCH (JSON)

```
<event-length>
{
  "type" : "LAUNCH",

  "launch" : {
    "framework_info" : {
      "id" : {"value" : "49154f1b-8cf6-4421-bf13-8bd11dccd1f1"},
      "user" : "foo",
      "name" : "my_framework"
    },
    "task" : {
      "name" : "dummy-task",
      "task_id" : {"value" : "d40f3f3e-bbe3-44af-a230-4cb1eae72f67"},
      "agent_id" : {"value" : "f1c9cdc5-195e-41a7-a0d7-adaa9af07f81"},
      "command" : {
        "value" : "sleep",
        "arguments" : ["100"]
      }
    }
  }
}
```

KILL

The `KILL` message is sent whenever the scheduler needs to stop execution of a specific task. The executor is required to send `TASK_KILLED` (or `TASK_FAILED`) terminal update back to the scheduler once it has stopped/killed the task. Mesos will mark the resources as freed once the terminal update is received.

KILL (JSON)

```
<event-length>
{
  "type" : "KILL",
```

```
  "kill" : {
    "framework_id" : {"value" : "49154f1b-8cf6-4421-bf13-8bd11dccd1f1"},
    "agent_id" : {"value" : "f1c9cdc5-195e-41a7-a0d7-adaa9af07f81"},
    "task_id" : {"value" : "d40f3f3e-bbe3-44af-a230-4cb1eae72f67"}
  }
}
```

ACKNOWLEDGED

Sent by the agent in order to signal the executor that a status update was received as part of the reliable message passing mechanism. Acknowledged updates must not be retried.

ACKNOWLEDGED Event (JSON)

```
<event-length>
{
  "type" : "ACKNOWLEDGE",

  "acknowledge" : {
    "framework_id" : {"value" : "49154f1b-8cf6-4421-bf13-8bd11dccd1f1"},
    "agent_id" : {"value" : "f1c9cdc5-195e-41a7-a0d7-adaa9af07f81"},
    "task_id" : {"value" : "d40f3f3e-bbe3-44af-a230-4cb1eae72f67"},
    "uuid" : "ZDQwZjNmM2UtYmJlMy00NGFmLWEyMzAtNGNiMWVhZTcyZjY3Cg=="
  }
}
```

MESSAGE

Custom message generated by the scheduler and forwarded all the way to the executor. These messages are delivered "as-is" by Mesos and they come with no delivery guarantees. It is up to the scheduler to retry if a message is dropped for any reason.

MESSAGE Event (JSON)

```
<event-length>
{
  "type" : "MESSAGE",

  "message" : {
    "framework_id" : {"value" : "49154f1b-8cf6-4421-bf13-8bd11dccd1f1"},
    "agent_id" : {"value" : "f1c9cdc5-195e-41a7-a0d7-adaa9af07f81"},
    "executor_id" : {"value" : "387aa966-8fc5-4428-a794-5a868a60d3eb"},
    "data" : "VGhpcyBpcyBhIHVucmVsaWFibGUgc2VudCBtZXNzYWdlCg=="
  }
}
```

SHUTDOWN

Sent by the scheduler in order to shutdown the executor. Once an executor gets a shutdown event it is required to kill all its tasks, send `TASK_KILLED` updates back to the scheduler and gracefully exit. If an executor doesn't terminate within a certain period after the event was emitted (*grace_period_seconds*, part of the SHUTDOWN event), the agent will forcefully destroy the container where the executor is running (which in turn will kill the executor and its tasks) and send an `TASK_FAILED` update to the master for all active tasks.

SHUTDOWN Event (JSON)

```
<event-length>
{
  "type" : "SHUTDOWN",
  “shutdown” : {
    “grace_period_seconds” : 5
  }
}
```

HTTP Response Codes

This is the list of expected (but not exhaustive) HTTP status codes for the HTTP API.

Status	Reason
200 OK	Returned for successful subscription request
202 Accepted	Returned for successful non-subscription requests
400 Bad Request	Returned for malformed requests
409 Conflict	Returned for requests with an incompatible version
403 Forbidden	Returned for non-subscription requests made before subscription request
401 Unauthorized	Returned for unauthenticated requests
406 Not Acceptable	Returned for requests with an unsupported accept header

Mandatory Environment Variables to be set by Agent

The following environment variables are mandatory and need to be set by the Agent when the executor starts-up:

- MESOS_FRAMEWORK_ID : FrameworkID needed as part of the SUBSCRIBE call.
- MESOS_EXECUTOR_ID : ExecutorID needed as part of the SUBSCRIBE call.
- MESOS_DIRECTORY : Path to the working directory for the executor.
- MESOS_RETRY_MAX_BACKOFF_FACTOR (new) : If set, denotes the maximum backoff duration to be used by the executor between two retries when disconnected.
- MESOS_AGENT_ENDPOINT (new) : A string containing the agent endpoint i.e. ip:port to be used by the executor to connect to the agent.
- MESOS_CHECKPOINT : If set, denotes if framework has checkpointing enabled.

The following mandatory environment variables were needed by the old executor driver but would no longer be needed:

- MESOS_SLAVE_ID : This would now be part of AgentInfo sent as part of SUBSCRIBED event by the agent.
- MESOS_SLAVE_PID : We no longer need a PID for exchanging messages with the agent with the new HTTP API.

We propose to rename the following environment variables so that they can be reused elsewhere in the Mesos code:

- MESOS_RECOVERY_TIMEOUT -> MESOS_*RETRY_INTERVAL_MAX* : This refers to the total time that executor should spend retrying before shutting itself down when it is disconnected from the agent. See the Backoff strategies section for more details.

More variables can be passed on by the operator to the executor if there is a need by decorator hooks.

Post MVP, we might think about moving to a JSON based object with all these variables instead of passing them around individually as part of the environment.

Agent Recovery

Currently, the agent supports two recovery mechanisms, specified by using the *--recover* flag at startup. Let's go over in detail on how each of them would be affected by the move to the HTTP API.

- ***reconnect*** : This mode allows the agent to reconnect with any of it's old live executors provided the framework has enabled checkpointing.

  In the old API (libprocess message passing , bi-directional communication), the agent used to send a *ReconnectExecutorMessage* to the Executor during the recovery

process. The executors, then, had a grace period of *EXECUTOR_REREGISTER_TIMEOUT* (2 seconds) to reconnect, failing, the container where the executor was running in was destroyed. The recovery for the slave was marked as complete after this step.

It's not possible to have this earlier mechanism owing to the unidirectional nature of the new API, the executors are encouraged to use the MESOS_RETRY_MAX_BACKOFF_FACTOR when retrying to ensure that the agent is able to receive at least one SUBSCRIBE request before marking the recovery as complete. If the agent, receives no SUBSCRIBE requests within this time, it assumes that the executor is hung and destroys the container it is running.

For the Initial MVP implementation, the agent would set the value of MESOS_RETRY_MAX_BACKOFF_FACTOR to just be equal to *EXECUTOR_REREGISTER_TIMEOUT*.

If the framework does not have checkpointing enabled, the executor should kill itself as soon as it notices a disconnection from the slave after a small grace period to clean up whatever pending tasks it is executing.

- ***cleanup*** : This mode kills any old live executors and then exits the agent. This is usually done by operators when making a non-compatible slave/executor upgrade.

  In the old API, the agent used to send a *ShutdownExecutorMessage* to the live executors, followed by a *executor_shutdown_grace_period (default: 5 seconds)*. The agent would then destroy the container where the executor is running in.

  In the new API, in the absence, of a bi-directional communication mechanism, for the executors connected via HTTP, the agent would first wait for the executor to SUBCRIBE again and would send it a SHUTDOWN event as soon as it reconnects. For hung executors, it would wait for *executor_shutdown_grace_period* and then kill the container where the executor is running in forcefully.

Backoff Strategies

The executors are encouraged to use a suitable backoff strategy like linear backoff when they notice a disconnection with the agent. Since in most cases for executor, the disconnection would be primarily TCP/IP level network errors i.e. an agent process going down/agent upgrade etc. These problems are generally temporary and tend to clear quickly. it is advisable to increase the delay in reconnects by e.g. 250ms for each attempt up to MESOS*_RETRY_INTERVAL_MAX*.

Furthermore, the executor should retry only if framework checkpointing is enabled. The executors can use the environment variable MESOS_RETRY_MAX_BACKOFF_FACTOR that denotes the maximum amount of time to wait in between two retries. This is specifically needed for Agent Recovery. The recovery of the agent is only marked complete once all the disconnected executors have connected and back/hung executors have been destroyed.

Hence, it is mandatory that every executor retries at least once within the maximum backoff period to ensure it is not shut-down by the agent due to being hung/unresponsive.