

# Task and Thread Actions

Task actions provide features for running and interacting with asynchronous tasks managed by the XINA Run application or AWS Lambda platform.

## Task Actions

### RUN

Run one or more asynchronous tasks.

#### Example

```
{
  "action" : "run",
  "tasks" : [ <task definition>, ... ]
}
```

#### Task Definition

A task definition may either be a numeric task ID of an existing task or a JSON object with the following format:

```
{
  "name"   : <string, task name>,
  "conf"   : <JSON object, format depends on task>,
  "parent" : <long, parent task ID, optional>,
  "thread" : <string, thread name, optional>,
  "auto"   : <boolean, optional, default false>,
  "archive" : <boolean, optional, default false>,
  "open"   : <boolean, optional, default false>,
  "desc"   : <string, optional>,
  "priority" : <int, optional>,
  "timeout" : <int, ms, optional>,
  "ref_id" : <long, optional>
}
```

---

## CONCLUDE

Explicitly conclude an asynchronous task. This is currently only used by AWS Lambda tasks, to notify the XINA server that the task has concluded. If a value is provided for "delay", the server will wait that many milliseconds before concluding the task. This supports tasks that may have a longer cleanup or import period following the immediate task completion.

### Example

```
{
  "action" : "conclude",
  "task"   : <task ID>,
  "delay"  : <number, ms, 0-5000, optional, default 0>
}
```

## CANCEL

Cancel one or more asynchronous tasks.

Only non-concluded tasks may be canceled. If "ignore" is false, and any specified tasks have concluded, an error will be thrown and no changes will occur. If "ignore" is true, all non-concluded specified tasks will be canceled, and any concluded tasks will be ignored.

### Example

```
{
  "action" : "cancel",
  "tasks"  : [ <task ID>, ... ],
  "ignore" : <boolean, optional, default false>
}
```

## CLEAN

Permanently delete one or more asynchronous task records and any associated files.

Only concluded tasks may be cleaned. If "ignore" is false, and any specified tasks have not concluded, an error will be thrown and no changes will occur. If "ignore" is true, all concluded specified tasks will be cleaned, and any non-concluded tasks will be ignored.

### Example

```
{
  "action" : "clean",
  "tasks"  : [ <long, task ID>, ... ],
  "ignore" : <boolean, optional, default false>
}
```

```
}
```

# DESTROY

Cancel and clean one or more asynchronous tasks. Unlike the CANCEL and CLEAN actions, this will apply to all tasks regardless of the current task state.

## Example

```
{  
  "action" : "clean",  
  "tasks" : [ <long, task ID>, ... ]  
}
```

# Thread Actions

## PAUSE

Pause execution of one or more asynchronous task threads. This does not affect any task currently running in the thread, but future tasks assigned to the thread will not run until the thread is resumed.

## Example

```
{  
  "action" : "pause",  
  "threads" : [ <string, thread name>, ... ]  
}
```

## RESUME

Resume execution of one or more asynchronous task threads. If "continue" is `true` and a continuable task is currently locked on the thread, that task will be continued.

## Example

```
{  
  "action" : "pause",  
  "threads" : [ <string, thread name>, ... ],  
  "continue" : <boolean, optional, default false>  
}
```

}

Revision #11

Created 12 January 2023 20:31:15 by Nick Dobson

Updated 12 January 2023 21:11:41 by Nick Dobson