Operations

This chapter describes the operations in the Scheduler Web Services SOAP API.

Operations

This section provides an overview of the Tidal Entersprise Scheduler Web Services SOAP API by describing the operations in the API. Tidal recommends that you review the WSDL and schema files for details of each operation and its parameters.

addrule

The **addrule** operation defines a new job rule (job definition). It can be followed by the **modrule** operation to further define other job parameters.

After **addrule** has been issued, use the **submit** operation to add the job to the production schedule.

```
<operation name="addrule">
      <input message="tns:addrule"/>
      <output message="tns:addruleResponse"/>
    </operation>
```

agent

The **agent** operation enables or disables a connection to an agent provided you have the correct security privileges for editing this connection.

```
<operation name="agent">
      <input message="tns:agent"/>
      <output message="tns:agentResponse"/>
    </operation>
```

alerts

The **alerts** operation displays all the operator alerts presently in the production schedule in table format. The columns included are:

- **ID** — The alert ID (needed for the **alertset** operation).
- **Job Number** — The job number ID of the job that issued the alert.
- **Type** — The kind of alert issued.
- **Level** — The severity level of the alert, either Critical, Warning, Error, or Information.
- **Status** — The status of the alert, either Open(1), Acknowledged(2), or Closed(3).
- **Description** — The alert message as defined in the operator alert action used to issue the alert.
- **Response** — Operator notes taken in response to the alert.
- **Time** — The time the operator alert was closed.
- **User** — The runtime user of the job that closed the alert.

```
<operation name="alerts">
    <input message="tns:alerts"/>
    <output message="tns:alertsResponse"/>
</operation>
```

alertset

The **alertset** operation lets you manually set the status of an alert specified by the alert ID. To obtain the job number alert ID, use the **alerts** operation.

```
<operation name="alertset">
      <input message="tns:alertset"/>
      <output message="tns:alertsetResponse"/>
    </operation>
```

calendars

The **calendars** operation displays a list of calendars accessible to the current user.

```
<operation name="calendars">
      <input message="tns:calendars"/>
      <output message="tns:calendarsResponse"/>
    </operation>
```

calrecalc

The **calrecalc** operation recalculates all calendar dates.

```
<operation name="calrecalc">
      <input message="tns:calrecalc"/>
      <output message="tns:calrecalcResponse"/>
    </operation>
```

compile

The **compile** operation compiles the production schedule for the dates specified.

```
<operation name="compile">
      <input message="tns:compile"/>
      <output message="tns:compileResponse"/>
    </operation>
```

delrule

The **delrule** operation deletes a job or job group definition. You can either specify the alias or the ID of the job or job group.

```
<operation name="delrule">
      <input message="tns:delrule"/>
      <output message="tns:delruleResponse"/>
    </operation>
```

depadd

The **depadd** operation adds a new job dependency or file dependency.

```
<operation name="depadd">
      <input message="tns:depadd"/>
      <output message="tns:depaddResponse"/>
    </operation>
```

depdel

The **depdel** operation deletes job dependencies and file dependencies. It then replaces that job's predone instances in the production schedule.

Refer to the **submit** operation.

```
<operation name="depdel">
      <input message="tns:depdel"/>
      <output message="tns:depdelResponse"/>
    </operation>
```

getmasterstatus

The **getmasterstatus** operation provides the operational status of the UNIX master being used.

```
<operation name="getmasterstatus">
      <input message="tns:getmasterstatus"/>
      <output message="tns:getmasterstatusResponse"/>
    </operation>
```

getmasterversion

The **getmasterversion** operation displays the version of the UNIX master being used.

```
<operation name="getmasterversion">
    <input message="tns:getmasterversion"/>
    <output message="tns:getmasterversionResponse"/>
  </operation>
```

grpupd

The **grpupd** operation updates inherit attributes for jobs in the specified group. You can obtain the group's Job ID by using the **grpupd** operation.

```
<operation name="grpupd">
      <input message="tns:grpupd"/>
      <output message="tns:grpupdResponse"/>
    </operation>
```

historyPurge

The **historypurge** operation will delete the job history that is stored in the database.

```
<operation name="historyPurge">
      <input message="tns:historyPurge"/>
      <output message="tns:historyPurgeResponse"/>
    </operation>
```

hosts

The **hosts** operation lists information about all Scheduler hosts defined in Scheduler.

```
<operation name="hosts">
      <input message="tns:hosts"/>
      <output message="tns:hostsResponse"/>
    </operation>
```

inactrule

The **inactrule** operation inactivates or disables a job or job group. When a job or job group is inactivated, its occurrences (if any) are pulled from the production schedule.

```
<operation name="inactrule">
      <input message="tns:inactrule"/>
      <output message="tns:inactruleResponse"/>
    </operation>
```

jobadd

The **jobadd** operation lets you add a job or job group to the production schedule. You can add a job either by alias or by ID. You can obtain the job occurrence ID and/or alias by using the **listrule** operation. Unlike the **submit** operation, the job is submitted adhoc. An adhoc job is not dependent on a calendar because a new instance is submitted manually into the schedule.

```
<operation name="jobadd">
      <input message="tns:jobadd"/>
      <output message="tns:jobaddResponse"/>
    </operation>
```

jobcancel

The **jobcancel** operation cancels a job occurrence with the specified job ID from the production schedule. You can also specify whether canceling the job affects other dependent jobs. You can obtain the ID by running the **jobmon** operation.

```
<operation name="jobcancel">
      <input message="tns:jobcancel"/>
      <output message="tns:jobcancelResponse"/>
    </operation>
```

jobdep

The **jobdep** operation displays all the dependencies of the specified type belonging to a job or job group.

```
<operation name="jobdep">
      <input message="tns:jobdep"/>
      <output message="tns:jobdepResponse"/>
    </operation>
```

jobgo

The **jobgo** operation overrides all dependencies for a job or job group, allowing the job or job group to launch. Obtain job run IDs by running the **jobmon** operation.

```
<operation name="jobgo">
      <input message="tns:jobgo"/>
      <output message="tns:jobgoResponse"/>
    </operation>
```

jobhold

The **jobhold** operation prevents a job occurrence with the specified job occurrence ID from running in the production schedule. Obtain the job occurrence ID by running the **jobmon** operation.

```
<operation name="jobhold">
      <input message="tns:jobhold"/>
```

```
      <output message="tns:jobholdResponse"/>
    </operation>
```

jobmod

The **jobmod** operation modifies a Job Occurrence. You can obtain the job run ID by using the **jobmon** operation.

```
<operation name="jobmod">
      <input message="tns:jobmod"/>
      <output message="tns:jobmodResponse"/>
    </operation>
```

jobmon

The **jobmon** operation enables you display job occurrence information. Command options allow you to filter the information displayed.

```
<operation name="jobmon">
      <input message="tns:jobmon"/>
      <output message="tns:jobmonResponse"/>
    </operation>
```

jobrelease

The **jobrelease** operation resumes a job occurrence that was held with the **jobhold** operation and releases a job in a **Waiting on Operator** status. You can obtain the job run ID by using the **jobmon** operation.

```
<operation name="jobrelease">
      <input message="tns:jobrelease"/>
      <output message="tns:jobreleaseResponse"/>
    </operation>
```

jobrermove

The **jobremove** operation removes job occurrences from the production schedule. The job must have a prelaunched status. Once a job reaches Launched, Active, or one of the Completed statuses, it cannot be removed from the schedule. You can obtain the job rule ID by running the **listrule** operation.

```
<operation name="jobremove">
      <input message="tns:jobremove"/>
      <output message="tns:jobremoveResponse"/>
    </operation>
```

jobrerun

The **jobrerun** operation lets you manually rerun a completed job or job group. You can obtain the job run ID by using the **jobmon** operation.

```
<operation name="jobrerun">
      <input message="tns:jobrerun"/>
      <output message="tns:jobrerunResponse"/>
    </operation>
```

jobset

The **jobset** operation lets you manually set the completion status of a job. You can obtain the job run ID by using the **jobmon** operation.

```
<operation name="jobset">
      <input message="tns:jobset"/>
      <output message="tns:jobsetResponse"/>
    </operation>
```

listrule

The **listrule** operation lists information about job and job group rules. You can choose the information to list. You can also select which records to display.

```
<operation name="listrule">
      <input message="tns:listrule"/>
      <output message="tns:listruleResponse"/>
    </operation>
```

liststat

The **liststat** operation lists all possible job statuses and their associated status number in a two column format.

```
<operation name="liststat">
      <input message="tns:liststat"/>
      <output message="tns:liststatResponse"/>
    </operation>
```

mastershutdown

The **mastershutdown** operation shuts down the UNIX master.

```
<operation name="masterShutDown">
      <input message="tns:masterShutDown"/>
      <output message="tns:masterShutDownResponse"/>
    </operation>
```

modrule

The **modrule** operation modifies a job or job group definition (rule). To modify the rule, either the job ID or an alias must be specified.

```
<operation name="modrule">
      <input message="tns:modrule"/>
      <output message="tns:modruleResponse"/>
    </operation>
```

output

The **output** operation displays the output of a job.

```
<operation name="output">
      <input message="tns:output"/>
      <output message="tns:outputResponse"/>
    </operation>
```

pause

The **pause** operation temporarily halts the production schedule thereby preventing jobs, even those whose dependencies have been met, from being launched. To restart the production schedule, use the **resume** operation.

```
</operation>
    <operation name="pause">
      <input message="tns:pause"/>
      <output message="tns:pauseResponse"/>
    </operation>
```

qlimit

The **qlimit** operation lets you manually set the limit of an existing queue.

```
<operation name="qlimit">
      <input message="tns:qlimit"/>
      <output message="tns:qlimitResponse"/>
    </operation>
```

resume

The **resume** operation resumes the production schedule, allowing jobs to be launched after using pause to temporarily stop the production schedule. This does not apply to jobs that are waiting on dependencies.

```
<operation name="resume">
      <input message="tns:resume"/>
      <output message="tns:resumeResponse"/>
```

```
</operation>
```

status

The **status** operation displays the status of a job or job group instance.

Status is the condition or state of a job occurrence throughout its life cycle. When a job has entered the schedule and is waiting to run or is actively running, possible statuses include:

- Active
- Waiting on Children
- Launched
- Waiting on Resource
- Waiting On Dependencies
- Held
- Agent Unavailable
- Agent Disabled
- Waiting on Group
- Timed Out for Day
- Waiting on Operator

When a job completes, possible status values are:

- Completed Normally
- Completed Abnormally
- Error Occurred
- Orphaned
- Skipped
- Aborted
- Cancelled
- Timed Out

If the status of a job occurrence is Externally Defined, then Scheduler is waiting for an external status update.

```
<operation name="status">
      <input message="tns:status"/>
      <output message="tns:statusResponse"/>
    </operation>
```

submit

The **submit** operation replaces predone job or job group instances in the production schedule according to its calendar. If there are job or job group instances already completed, these instances are not replaced.

```
<operation name="submit">
      <input message="tns:submit"/>
```

```
      <output message="tns:submitResponse"/>
    </operation>
```

varset

The **varset** operation lets you manually set the value of an already existing variable.

```
<operation name="varset">
      <input message="tns:varset"/>
      <output message="tns:varsetResponse"/>
    </operation>
```