How To - SSH Session Management

FEATURE AVAILABILITY STARTING FROM RELEASE 1.9

The SSH Session Management adds the possibility to end orphaned remote processes started by SSH jobs or orphaned JobScheduler tasks from SSH jobs.

Use Case

What happens if the connection to your remote host breaks while a script is still running? How can the JobScheduler job which started the remote script know about that?

What happens if a remote script has finished, but the JobScheduler task which started the process remotely cannot know about that? (e.g. because of a temporarily broken network connection).

The SSH Session Management provides a solution for dealing with that type of issues: it provides the possibility of configuring an additional job chain to check for orphaned processes on the remote host as well as check for orphaned tasks.

You can configure your existing SSH job chain to start the monitoring job chain, which then will monitor the task of your original job chain as well as the processes started on the remote host via SSH.

Mode of Operation

To configure your SSH job to use the SSH Session Management you have to configure your SSH job and define a cleanup job chain for the cleanup work.

The feature requires the use of the *JSch* implementation by JCraft. See How To - Usage of the SSH Job (JobSchedulerSSHJob) with JCraft's JSch for more information about configuring your SSH job to use the JSch implementation,

The SSH Session Management will carry out one of the following actions after checking the remote processes and JobScheduler tasks:

- if a remote process is running and the JobScheduler task is still alive:
    - do nothing, the cleanup job chain goes to a setback condition and waits for another start
- if a remote process is running but the JobScheduler task is no longer available:
    - the cleanup job tries to end the process on the remote host
- if a remote process is no longer available but the JobScheduler task is still running:
    - the cleanup job ends the task immediately
- if a remote process and the JobScheduler task are not available anymore
    - do nothing, the cleanup job ends

Related Development Issues

JITL-152 - Getting issue details... STATUS

JITL-147 - Getting issue details... STATUS

JITL-123 - Getting issue details... STATUS

JITL-112 - Getting issue details... STATUS

Configuration of the SSH job to be monitored

A number of additional parameters have to be added to the job configuration before the SSH job can be monitored:

runWithWatchdog

- format:
  - boolean
- default value:
  - `false`
- description:
  - If this parameter is set to true, the job itself is aware that it is monitored and generates a new order for the second job chain with all parameters of the job and the pid of the connected shell. If the parameter is set to false or is not present the SSH Session Management will not be processed.

cleanupJobchain

- format:
  - string
- default value:
  - empty
- description:
  - This parameters defines the path to the configuration files of the job chain to use for the cleanup work.

ssh_job_kill_pid_command

- format:
  - **`<command> \${pid}`**
- default value:
  - `kill -9 \${pid}`
- description:
  - The command to kill a process on the remote machine. The command depends on the OS of the remote host. If the command is not set, the cleanup job checks whether the remote host is running on a Linux or on a Windows system and uses the relevant appropriate commands.
    The **`${pid}`** placeholder will be substituted by the cleanup job. Note that the leading $ character has to be escaped with "\".

ssh_job_terminate_pid_command

- format:
  - **`<command> \${pid}`**
- default value:
  - `kill -15 \${pid}`
- description:
  - The command for terminating a process on the remote machine. This command depends on the OS of the remote host. If the command is not set, the cleanup job checks whether the remote host is running on a Linux or on a Windows system and uses the appropriate default commands.
    The **`${pid}`** placeholder will be substituted by the cleanup job. Note that the leading $ character has to be escaped with "\".

ssh_job_get_pid_command

- format:
  - **`<command>`**
- default value:
  - `echo $$`
- description:
  - A command or script that writes the pid of the connected shell to stdout of the remote host.

ssh_job_get_child_processes_command

- format:
  - **`<command> \${pid}`**

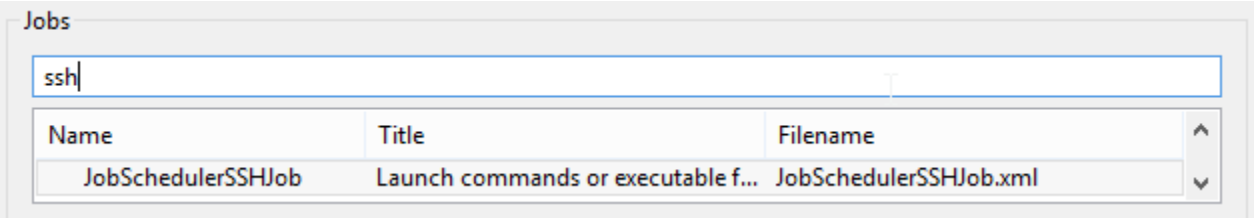
- default value:
  - `/bin/ps -ef | pgrep -P\${pid}`
- description:
  - The command or script determines the child processes of the given pid. The command or script depends on the OS of the remote host. If the command is not set, the default command for Linux is used.
    The **`${pid}`** placeholders will be substituted by the cleanup job. Note that the leading $ character has to be escaped with "\".

ssh_job_get_active_processes_command

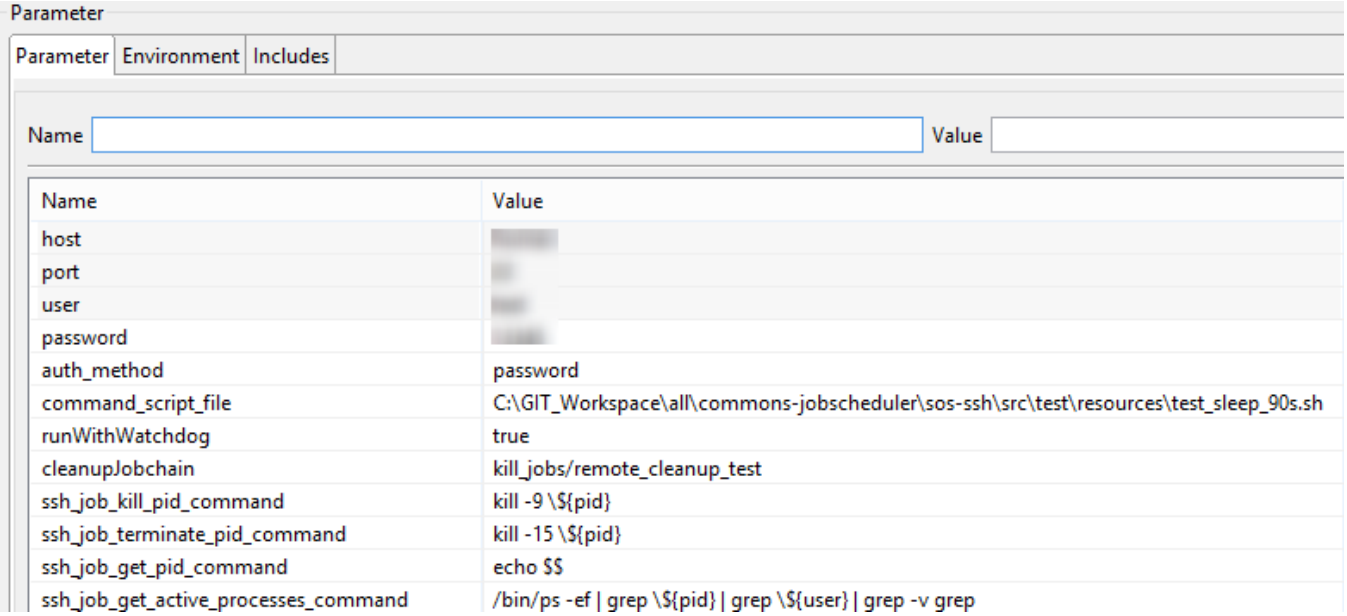
- format:
  - **`<command> \${pid} \${user}`**
- default value:
  - `/bin/ps -ef | grep \${pid} | grep \${user} | grep -v grep`
- description:
  - The command or script checks if the process on the remote host is still running. The cleanup job expects an exitcode = 0 if the process is still running or other than 0 if the process is not available anymore. The command or script depends on the OS of the remote host. If the command is not set, the default command for Linux is used.
    The **`${pid}`** and **`${user}`** placeholders will be substituted by the cleanup job. Note that the leading $ character has to be escaped with "\".
  - Example commands:
    - `/bin/ps -ef`
      - writes all running processes to stdout on the remote host
    - `| grep \${pid}`
      - filters the result to show only results containing the given ${pid}
    - `| grep \${user}`
      - filters the result to show only results containing the given ${user}
    - `| grep -v grep`
      - filters the grep command itself

Example: Configuration with JOE

Job



Parameters



Example: XML Configuration

job xml

```
<job title="Launch commands or executable files by SSH">
  <description>
    <include file="jobs/JobSchedulerSSHJob.xml"/>
  </description>
  <params>
    <param name="host" value="[HOST]"/>
    <param name="port" value="[SSHPORT]"/>
    <param name="user" value="[USERNAME]"/>
    <param name="password" value="[PASSWORD]"/>
    <param name="auth_method" value="password"/>
    <param name="command_script_file" value="[PATH_TO_SCRIPTFILE]\test_sleep_90s.sh"/>
    <param name="runWithWatchdog" value="true"/>
    <param name="cleanupJobchain" value="kill_jobs/remote_cleanup_test"/>
    <param name="ssh_job_kill_pid_command" value="kill -9 \${pid}"/>
    <param name="ssh_job_terminate_pid_command" value="kill -15 \${pid}"/>
    <param name="ssh_job_get_pid_command" value="echo $$"/>
    <param name="ssh_job_get_active_processes_command" value="/bin/ps -ef | grep \${pid} | grep \${user} | grep
-v grep"/>
  </params>
  <script java_class="sos.scheduler.job.SOSSSHJob2JSAdapter" language="java"/>
  <run_time/>
</job>
```

Configuration of the Cleanup job chain

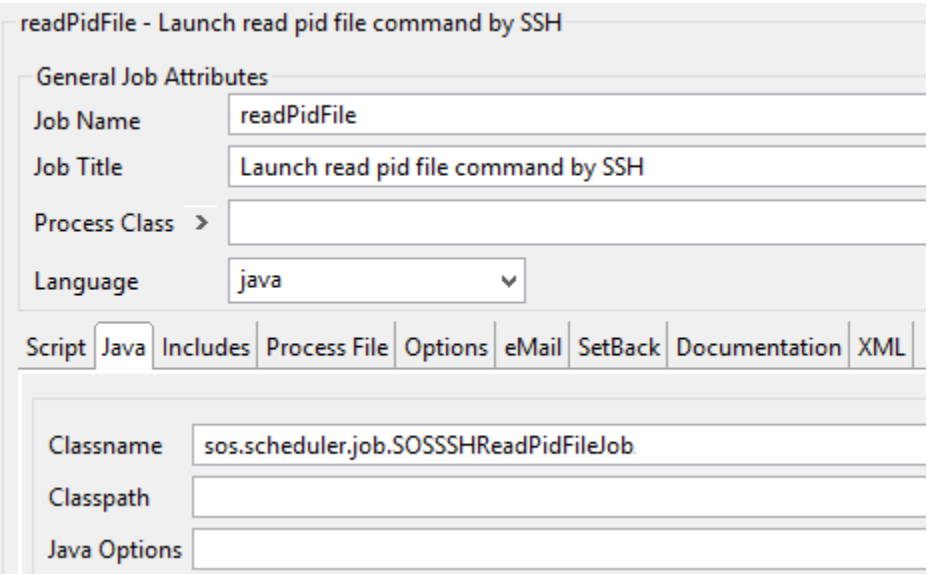
A cleanup job chain with two jobs has to be configured to process the cleanup of the remote processes or the JobScheduler task.

The cleanup job chain consists of two jobs, one to read the pid of the connected shell from a temporary file on the remote host and one to check if the process or the JobScheduler task is still running. The temporary file will be generated automatically and deleted after processing. The second job also ends either the remote process or the JobScheduler task as appropriate.

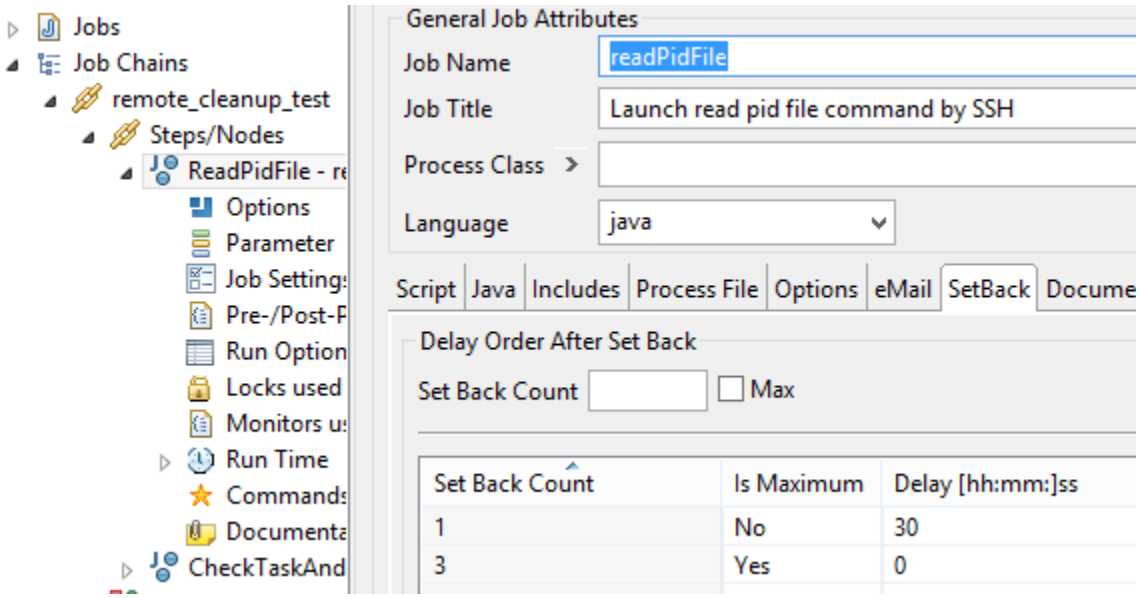
Job 1: The read-pid-from-temporary-file job

The first job reads the pid from the temporary file on the remote host. If your SSH job is configured as described above, the temporary file will have been created automatically on the remote host.

Configure the Class of the Job in JOE like this:



After choosing the relevant class name from the list, configure a setback for the job. The setback will be used to restart the job according to the conditions found by the job (described above).



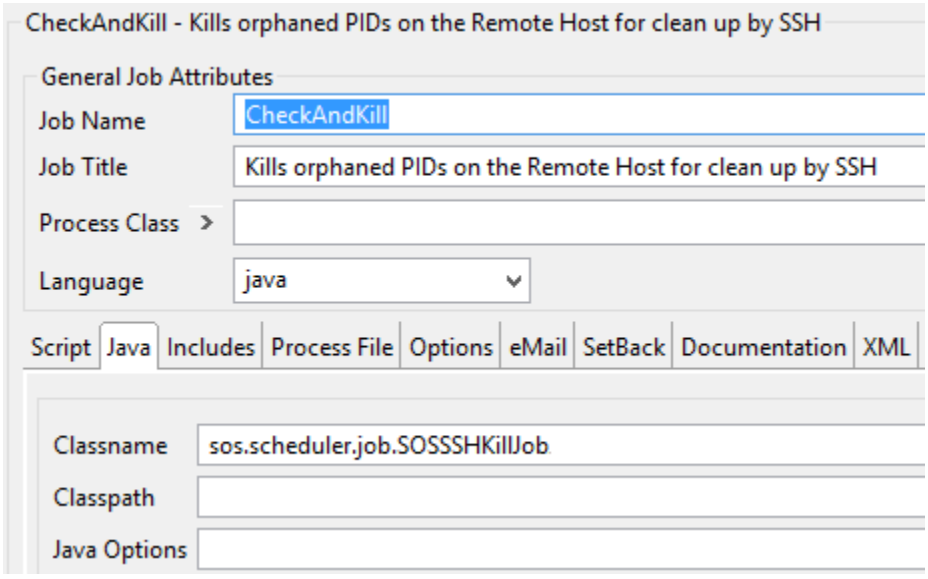
Job 2: The check-and-kill job

The second job checks:

- if the pid determined by the first job is still active
- if the JobScheduler task is still active
- kills the remote process or the JobScheduler task

depending on the conditions found. The actions taken are described in the Mode of Operation chapter above.

Configure the class of the job in JOE as follows:



Example: XML Configuration

Job1: ReadPidFileJob

```
<job order="yes" stop_on_error="no" title="Launch read pid file command by SSH">
  <description>
    <include file="jobs/SOSSSHReadPidFileJob.xml"/>
  </description>
  <script java_class="sos.scheduler.job.SOSSSHReadPidFileJobJSAdapter" language="java"/>
  <delay_order_after_setback delay="30" is_maximum="no" setback_count="1"/>
  <delay_order_after_setback delay="0" is_maximum="yes" setback_count="3"/>
  <run_time/>
</job>
```

Job2: CheckAndKillJob

```
<job order="yes" stop_on_error="no" title="Kills orphaned PIDs on the Remote Host for clean up by SSH">
  <description>
    <include file="jobs/SOSSSHKillJob.xml"/>
  </description>
  <script java_class="sos.scheduler.job.SOSSSHKillJobJSAdapter" language="java"/>
  <delay_order_after_setback delay="30" is_maximum="no" setback_count="1"/>
  <delay_order_after_setback delay="0" is_maximum="yes" setback_count="3"/>
  <run_time/>
</job>
```

Jobchain: Cleanup

```
<job_chain orders_recoverable="yes" visible="yes">
  <job_chain_node error_state="ERROR" job="readPidFile" next_state="CheckTaskAndRemoteProcessesAndKillIfNeeded"
on_error="setback" state="ReadPidFile"/>
  <job_chain_node error_state="ERROR" job="CheckAndKill" next_state="SUCCESS" on_error="setback" state="
CheckTaskAndRemoteProcessesAndKillIfNeeded"/>
  <job_chain_node state="ERROR"/>
  <job_chain_node state="SUCCESS"/>
</job_chain>
```

Change Management References

T	Key	Linked Issues	Fix Version/s	Status	P	Summary	Updated
+	JITL-206	YADE-306	1.10.1, 1.11	RELEASED	⏬	Add the PreferredAuthentications configuration to the SSH implementation with JSch	Dec 21, 2015
⧉	JITL-152		1.9	RELEASED	⏬	Documentation about the limitations of SSHJobs on Windows Systems	Nov 30, 2015
+	JITL-151		1.10	RELEASED	⏫	SSH jobs should support elliptic curves (ECDSA) for authentication	Jan 27, 2016
+	JITL-147	JITL-124 , JITL-123 , JITL-180 , JITL-181	1.9	RELEASED	⏬	Improve SSH Session management to monitor and kill remote sessions	Nov 20, 2015
+	JITL-112	JITL-123	1.9	RELEASED	⏬	Setting order parameters by shell scripts executed with SSH Jobs	Jan 27, 2016
+	JITL-53		1.9	RELEASED	⏫	SSH Job does not work with VMS	Nov 19, 2015

6 issues