How to prevent a job starting when another job is running

Introduction

You can use *locks* to prevent a job starting while another job is running. You can find the information about locks at http://www.sos-berlin.com/doc/en/scheduler.doc/lock.xml

Locks are defined in files in the hot folder (`live`). Lock file names follow the convention:

- the name of the lock followed by `.lock.xml`.
  E.g. *`lockSample.lock.xml`*

To use locks, you have to:

1. declare a lock
2. assign the lock to the required jobs

Example

A typical lock declaration would be:

```
   <locks>
       <lock name="lockSample"/>
   </locks>
```

To assign the lock to your jobs use:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<job order="yes"
    stop_on_error="no">
   <lock.use lock="lockSample"
             exclusive="yes"/>
   <script language="shell">
       <![CDATA[
echo "here is the job jobsSample_1"
echo "I'm not running in parallel with job jobsSample_2"
ping -n 60 localhost
        ]]>
   </script>
</job>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<job order="yes"
    stop_on_error="no">
   <lock.use lock="lockSample" exclusive="yes"/>
   <script language="shell">
       <![CDATA[
echo "here is the job jobsSample_2"
echo "I'm not running in parallel with job jobsSample_1"
ping -n 60 localhost
        ]]>
   </script>
</job>
```

Two job chains using the jobs

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<job_chain>
   <job_chain_node state="100"
                   job="jobsSample_1"
                   next_state="success"
                   error_state="error"/>
   <job_chain_node state="success"/>
   <job_chain_node state="error"/>
</job_chain>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<job_chain>
   <job_chain_node state="100"
                   job="jobsSample_2"
                   next_state="success"
                   error_state="error"/>
   <job_chain_node state="success"/>
   <job_chain_node state="error"/>
</job_chain>
```

Scope of locks

- Where are lock files saved?
  - Locks are stored directly in the `live` folder or any sub-folders.
- How do you reference locks in jobs?
  - Locks are identified by their path:
    - this is made up of the folder where the lock is stored and of the name of the lock.
  - Locks that are located in the same folder as the job can be addressed using the lock's name (omitting the folder).
- Example:
  - Lock Location
    - Folder `live/project_a` contains a set of jobs and a lock named `my_lock_a`
    - Folder `live/project_b` contains a set of jobs and a lock named `my_lock_b`
  - Lock Usage
    - `job_a` from the folder named `project_a` can reference the lock from its folder using for example:
      - `<lock.use name="my_lock_a" exclusive="true"/>`
    - Job `job_b` from folder `project_b` can be configured to use its local lock in a similar manner.
    - Should it be necessary to prevent jobs `job_a` and `job_b` from runnig in parallel then they have to use a common lock. This is achieved by referencing a common lock, e.g. job `job_b` could use a reference to `my_lock_a` from the folder `project_a` like this:
      - `<lock.use name="/project_a/my_lock_a" exlcusive="yes"/>`
    - Caveat: It is possible to use the same lock name in different folders to represent different locks. Jobs using relative lock addressing (i.e. just the locks' name) and omitting the folder name would reference the lock in their local folder. Consider using absolute lock addressing if you want jobs from different folders to make use of the same lock.
    - Hint: Locks that are located directly in the `live` folder can be addressed with a leading slash as follows:
      - `<lock.use name="/my_global_lock" exclusive="yes"/>`