

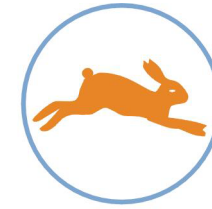
Open Source Job Scheduler



Architecture and Mode of Operation

<http://jobscheduler.sourceforge.net>

Contents

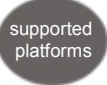


JOBSCHEDULER

- **Components**
- **Platforms & Databases**
- **Architecture**
- **Job Configuration**
- **Deployment**
- **Distributed Processing**
- **Security**
- **Failsafe Operation**



Icons used on these pages


 supported platforms Works with all supported platforms


 supported platforms Works with all supported platforms and Java


 any platform Any platform of your choice applicable

 supported database Works with all supported databases

 any database Any database of your choice applicable

 Job Jobs

 DB Managed Job Managed Jobs are stored in a database

 Javascript is required for Ajax GUI

Components

Run-time Components for Job Execution



- Jobs and job chains are executed in the Job Scheduler Engine in batch mode
- The built-in Operations GUI is accessible for browsers and is used to control jobs and job chains, e.g. start, stop, suspend, resume, and to access log files



- Backup Job Scheduler instances are used for failsafe operation
- Failover is performed automatically

Design-time Components for Job Configuration



- Hot Folders store job configurations and are monitored by Job Scheduler for changes that are immediately detected and automatically applied by the respective Job Scheduler instance



- Define jobs and job chains with a client GUI (same functionality as Managed Jobs GUI)
- Store job configurations in XML files on disk
- Manage job configurations with Hot Folders



- Define jobs and job chains by browser (same functionality as Job Editor GUI)
- Store job configurations in a database
- Submit jobs and job chains to one or more Job Scheduler instances

Platforms & Databases

Supported Operating Systems

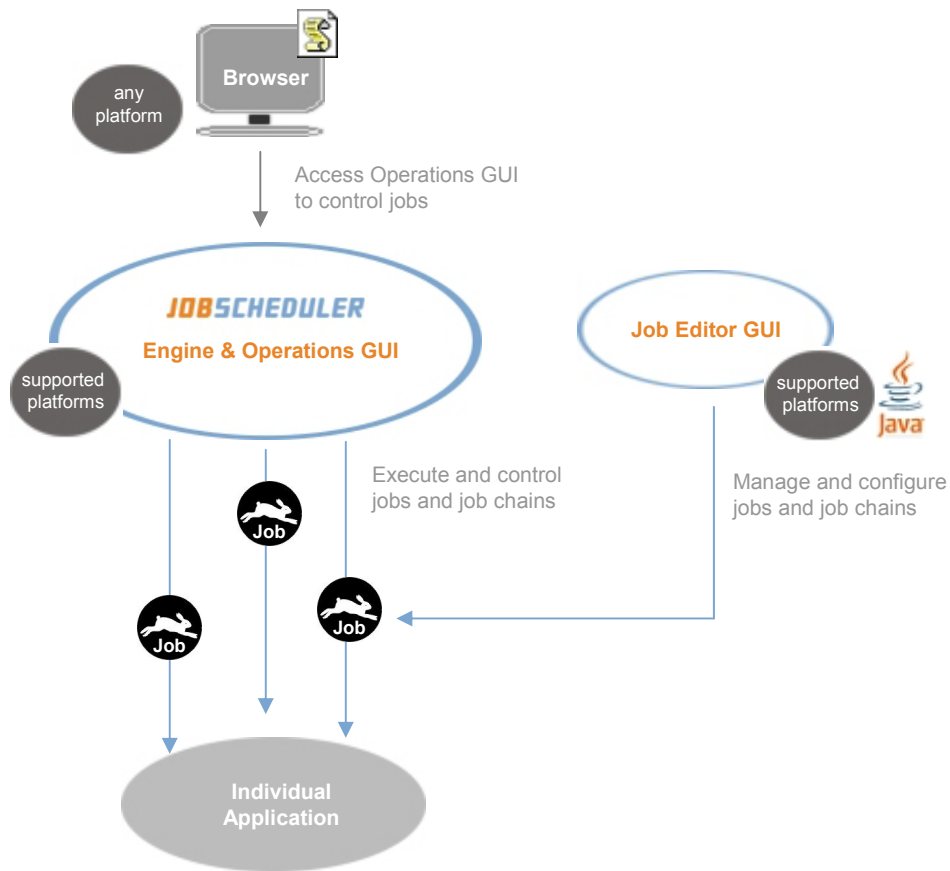
- Windows 2000, 2003, XP, Vista
- Linux starting with kernel 2.4
- Solaris 8, 9, 10
- HP-UX 11 (PA-RISC, IA-64)
- IBM AIX 5.3

Supported Databases

- DB2 8.x, 9.x
- Oracle 8.1.7, 9.x, 10.x
- SQL Server 2000, 2005
- Sybase ASE 15
- Firebird 1.5
- MySQL 4.1, 5.x
- PostgreSQL 8.x

Architecture

Single Server Operation



At a Glance:

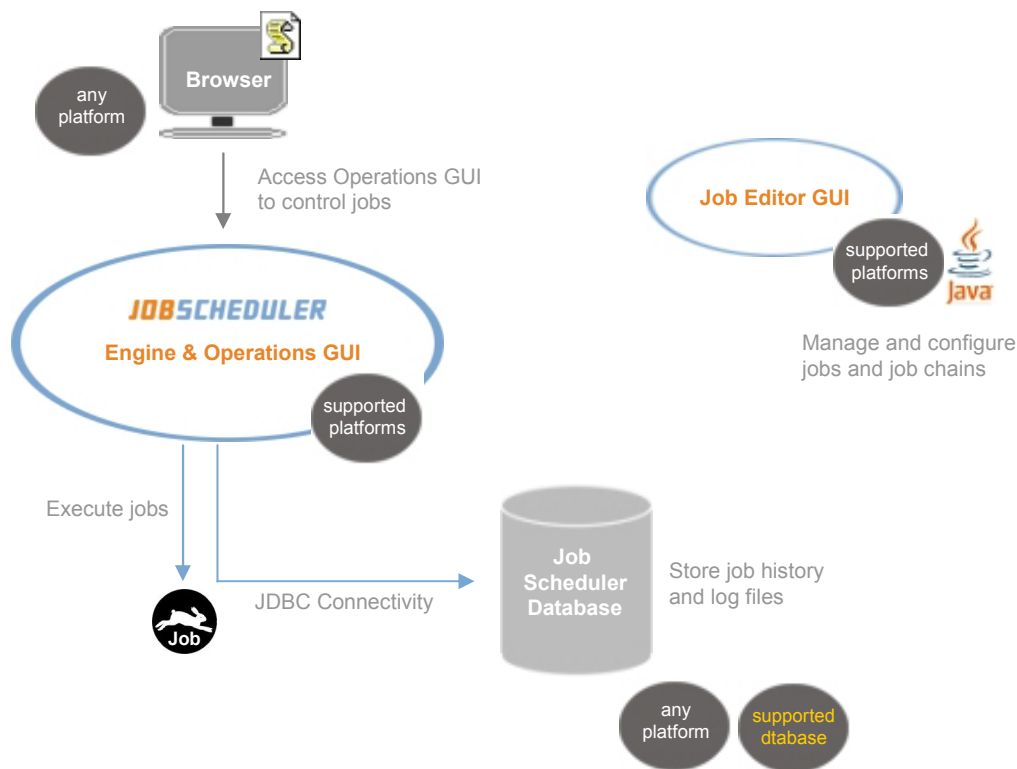
- You can operate the Job Scheduler on a single server without a database.
- If you want to operate Job Scheduler without a database, no additional software is required.
- The implementation of the Job Scheduler includes a built-in Operations GUI.
- You can use the Job Editor GUI to configure jobs that are stored as XML files.
- Use any of the supported platforms.

By the way!

- You can use any editor of your choice to manage job configurations in XML files.

Architecture

Using a Database with regard to Compliance



At a Glance:

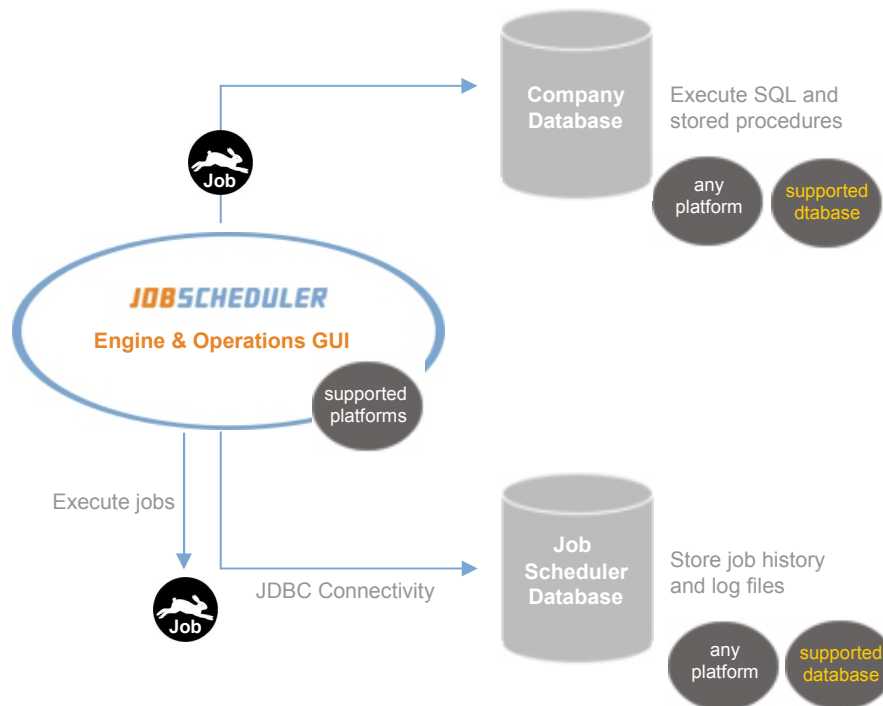
- If you decide to use a database, then you can locate it on any host and platform.
- Connectivity is established per JDBC.

By the way!

- The advantage of using a database is that you keep track of the job history and log files for compliance of your IT processes.
- Job history and log files can be accessed directly by the Operations GUI per browser.

Architecture

Accessing Company Databases

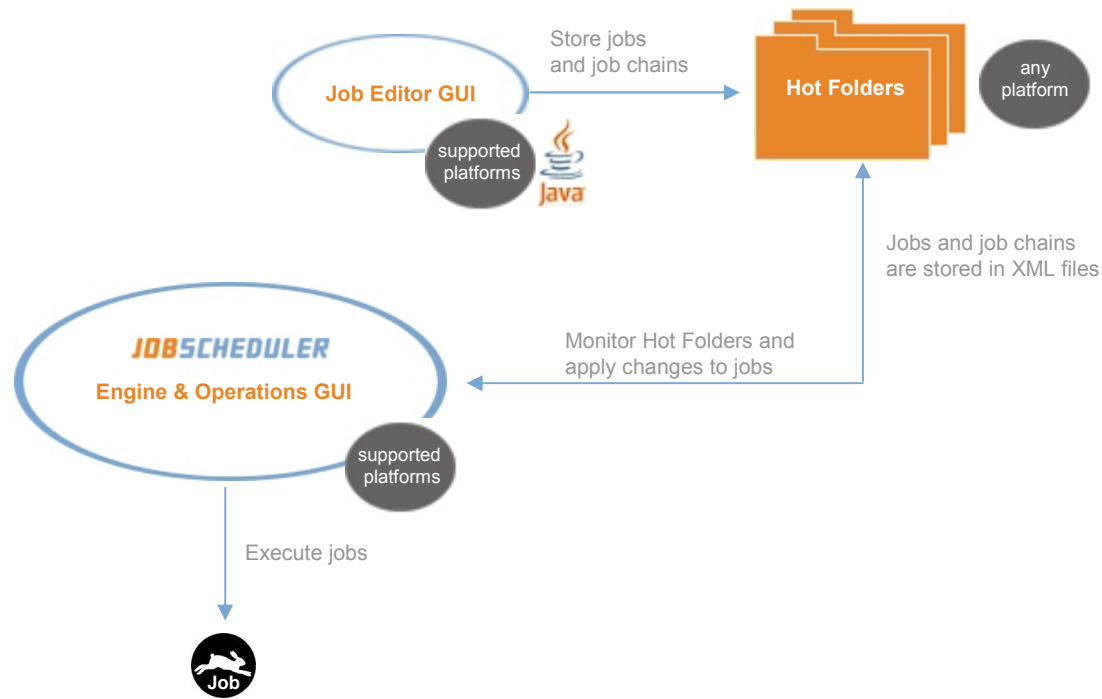


At a Glance:

- If you want to use a database, you can use different products for your architecture.
- For example you could add a company database for the execution of SQL statements and stored procedures by jobs.
- You can use a different database product for the job history and log files of Job Scheduler.

Job Configuration

Using Hot Folders



At a Glance:

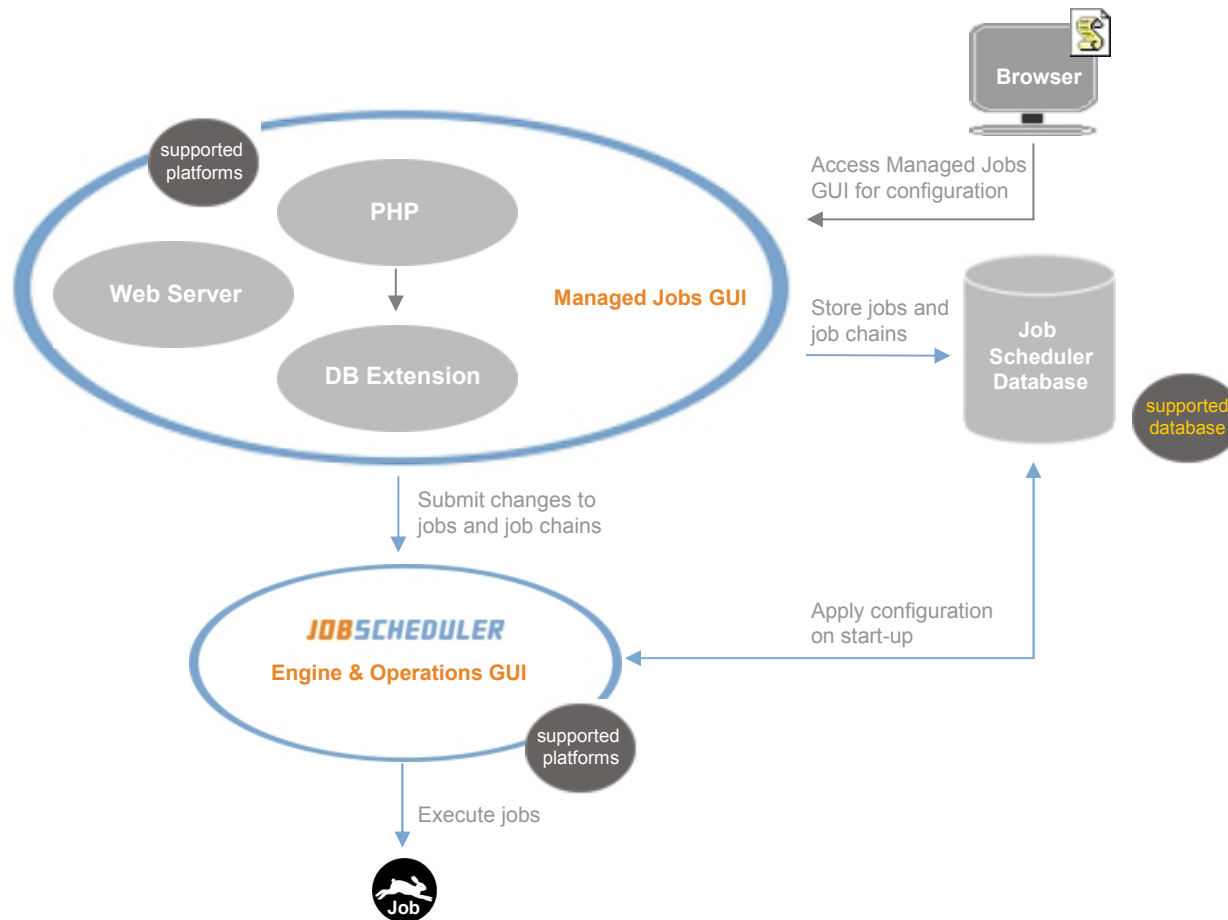
- If you use Hot Folders then Job Scheduler will automatically monitor changes to the configuration.
- The Job Editor GUI enables you to manage configurations in Hot Folders on disk.
- The Job Scheduler monitors the Hot Folders constantly. Any changes are immediately detected.
- The requested changes are applied right away without the need for a restart of Job Scheduler.

By the way!

- You could still use at any time your favorite editor in order to configure jobs in Hot Folders.

Job Configuration

Using Managed Jobs GUI



At a Glance:

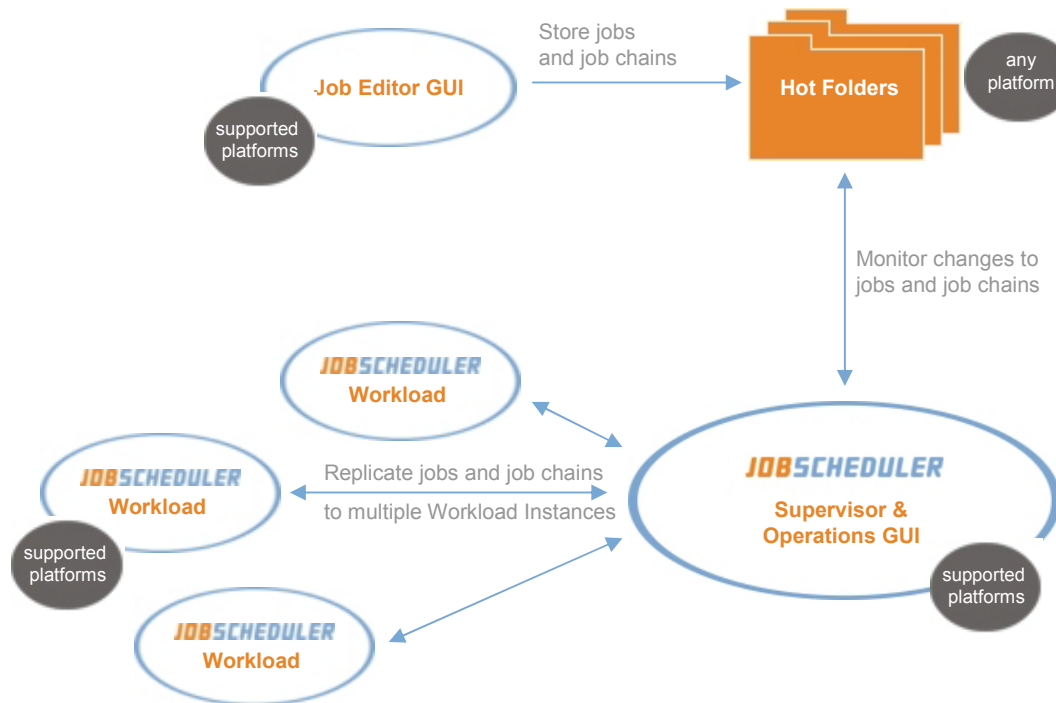
- If you want to use the Managed Jobs GUI with a web server, PHP and DB extension, you will require a database client to access the database.
- The Managed Jobs GUI is operated with Apache or IIS.
- The Jobs are stored in the Job Scheduler database.
- The Job Scheduler applies changes after submission by the Managed Jobs GUI.

By the way!

- LAMP or WAMP will facilitate this solution perfectly, configurations with PHP are easier to handle.

Deployment

Deploy Jobs and Job Chains by Hot Folders

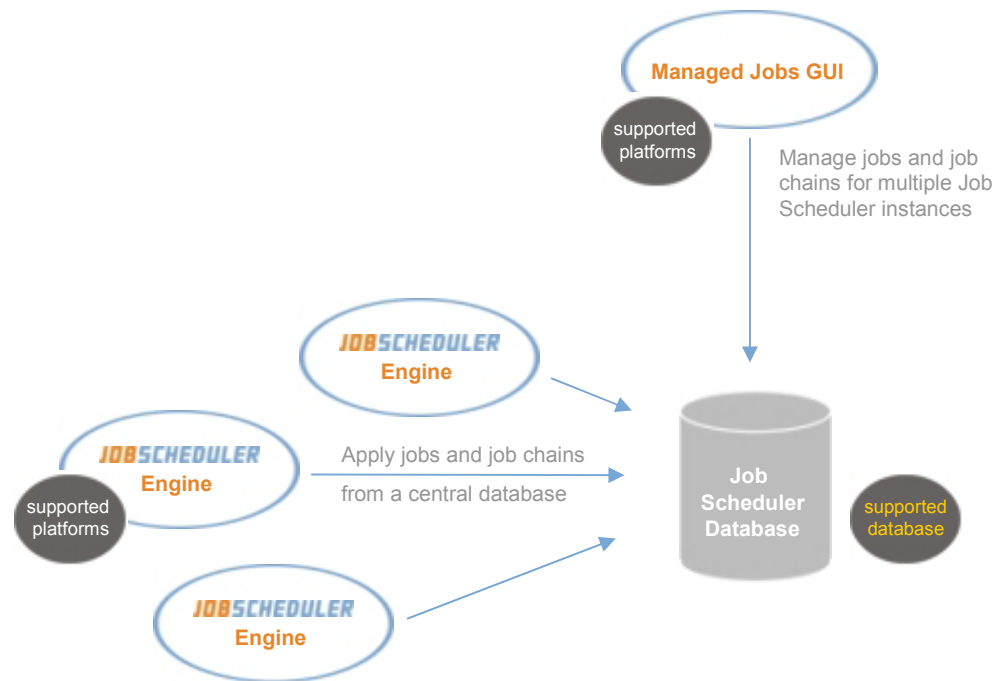


At a Glance:

- Job Scheduler Supervisor replicates job configurations from Hot Folders to the Workload Instances.
- Some jobs are deployed to all instances while other jobs are deployed to specific Workload Instances.
- Workload Instances connect on start-up to a Supervisor Instance and update their jobs and schedules.
- Workload instances can execute jobs independently of a Supervisor Instance.
- Workload Instances re-connect automatically to a Supervisor Instance after network outages.

Deployment

Deploy Jobs and Job Chains by Managed Jobs GUI



At a Glance:

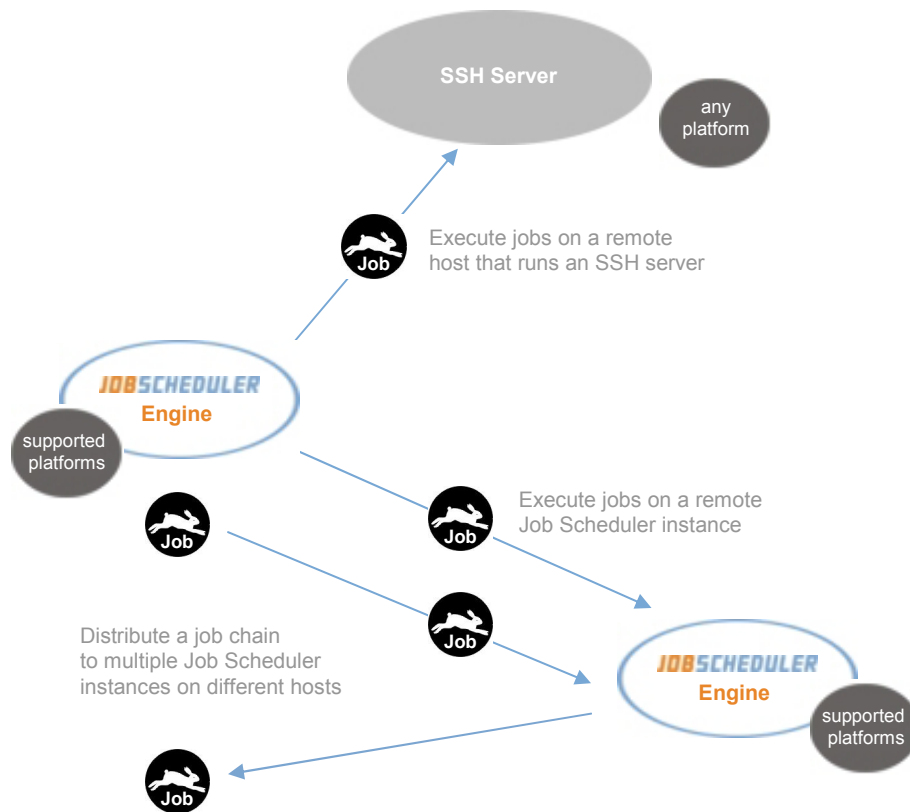
- Jobs for multiple Job Scheduler instances are stored in a central database.
- With the Managed Jobs GUI you can set up jobs and job chains and manage their configuration centrally.

By the way!

- This solution will be most suitable if you intend to operate an arbitrary number of Job Schedulers with a large number of jobs and job chains.

Distributed Processing

Remote Execution

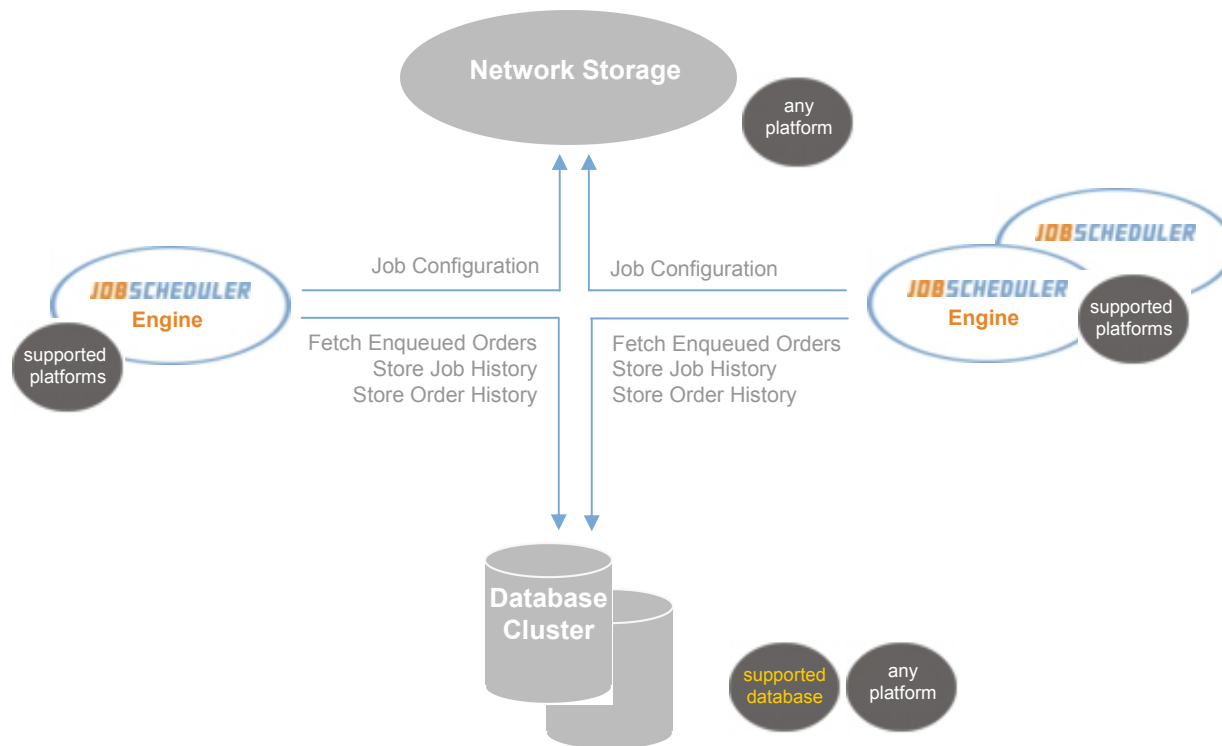


At a Glance:

- Job Scheduler supports the execution of jobs via SSH on remote hosts. No SSH client is required.
- Job Scheduler supports the execution of jobs on remote Job Scheduler instances.
- Job chains for execution of individual job steps on remote hosts are supported.

Distributed Processing

Load Balancing



At a Glance:

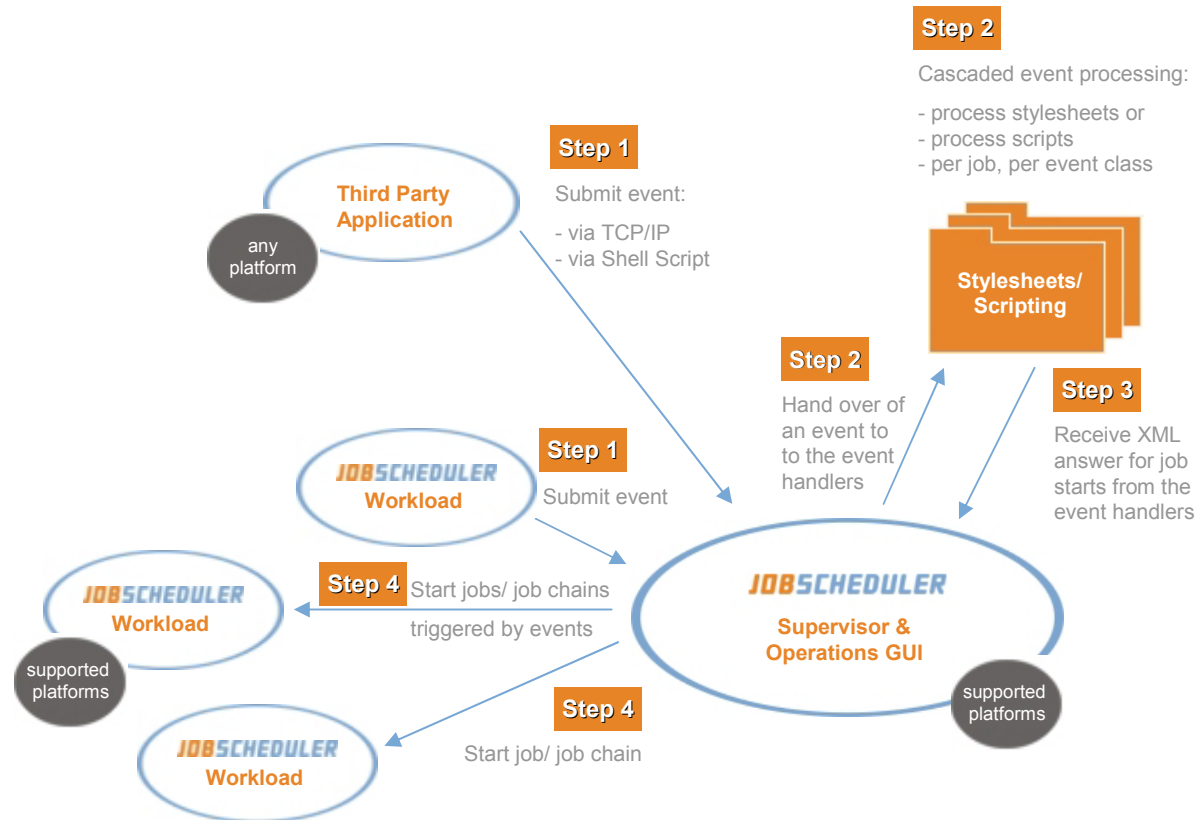
- For Load Balancing you can operate an arbitrary number of Job Schedulers in parallel.
- All Job Schedulers use the same job configurations and database.
- Job configurations are loaded from a network storage, alternatively from a clustered database (Managed Jobs).
- Job Schedulers concurrently execute jobs and job chains that are synchronized by a central database.

By the way!

- This solution fits well if you want to scale the same jobs for parallel processing on multiple hosts.
- Since you can process jobs parallelly you can achieve high efficiency in your processing time.

Distributed Processing

Event Processing

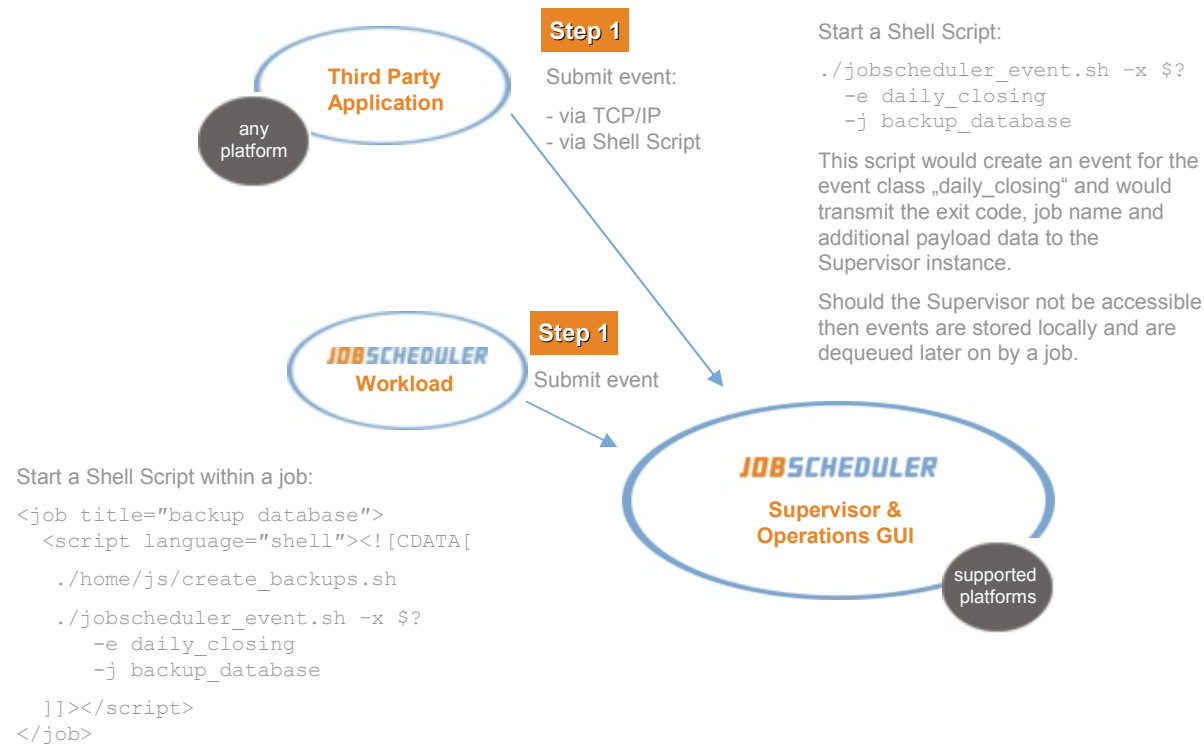


At a Glance:

- Any jobs or Third Party Applications could submit events to the Supervisor Job Scheduler (step 1).
 - The Supervisor hands over events to a cascade of event handlers that are implemented as XSLT stylesheets or scripts (step 2).
- Event handlers implement individual conditions for job starts and return the respective XML commands should a job or job chain be started (step 3).
- The Supervisor causes the respective jobs/ job chains to be started in the remote Workload instance (step 4).

Distributed Processing

Event Processing: Submitting Events



At a Glance:

- Any jobs or Third Party Applications could submit events to the Supervisor Job Scheduler (step 1).
- Sending events can be implemented by use of a shell script that hands over job data and additional payload data to the Supervisor instance.
- In case of failure to access the Supervisor instance events are stored locally and are dequeued by a job on a regular basis.

Distributed Processing

Event Processing: Cascading Event Handlers

Sample stylesheet that checks successfully terminated jobs and starts a job if the respective conditions apply:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:sos="http://www.sos-berlin.com" version="1.0">
  <xsl:import href="scheduler_event_functions.xsl.inc"/>
  <xsl:output indent="yes"/>

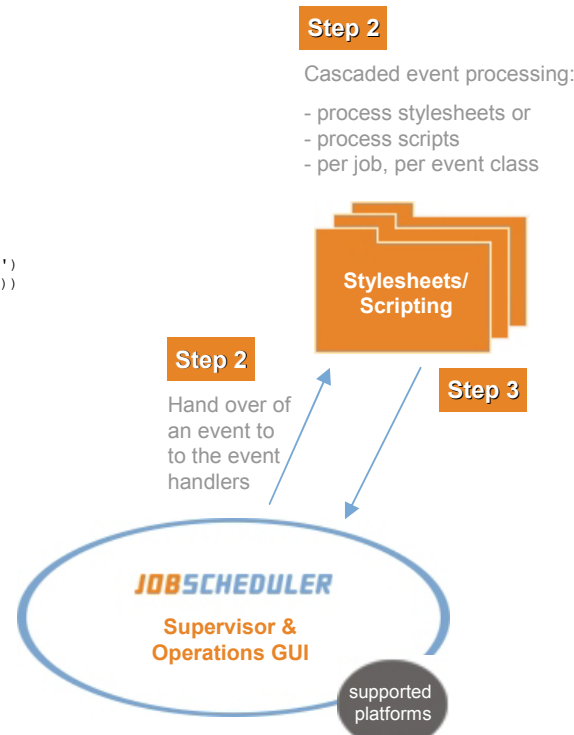
  <xsl:template match="events[
    (sos:d('0357export-exinanag') and sos:d('0357export-exinordi'))
    and sos:d('0357export-exinrice') and sos:d('0357export-exinveas'))
    or sos:d('0357fgio-EXPORT_DATABASE')
  ]">

    <xsl:call-template name="run_job">
      <xsl:with-param name="job">sample_jobs/0357fgio-
JEODF001</xsl:with-param>
      <xsl:with-param name="host">localhost</xsl:with-param>
    </xsl:call-template>

    <remove_event>
      <event event_class="JEODF001"/>
    </remove_event>
  </xsl:template>
</xsl:stylesheet>
```

Event handlers are stored on disk and are triggered according to a naming scheme:

- process event handler for a given job chain
- process event handler for a given job
- process event handler for a given event class
- process default event handler



Event handlers return XML elements for Job Scheduler commands that would cause a job chain or a job to start as in:

```
<start_job job="backup database"/>
<add_order job_chain="backup"/>
```

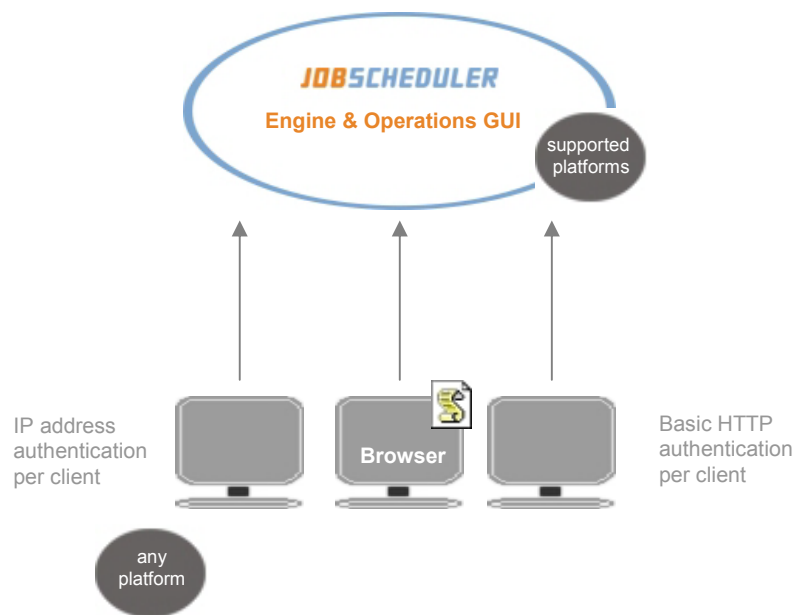
At a Glance:

- Event handlers are used in order to implement individual conditions for job execution. They are implemented as XSL stylesheets or with JavaScript..
- The Supervisor triggers event handlers according to the information given with the event. The more specific event handlers are triggered first.

Therefore a sequence of job chain, job, event class and finally the default event handler is triggered cascadingly.

Security

Basic Authentication



At a Glance:

- If you want to apply an individual authentication, you can choose between Basic HTTP authentication and IP address authentication.
- If you use Basic HTTP authentication the user specific passwords are stored MD5 encrypted in the Job Scheduler configuration.
- In case you use IP address authentication a fixed address per client or network range is specified in the configuration.

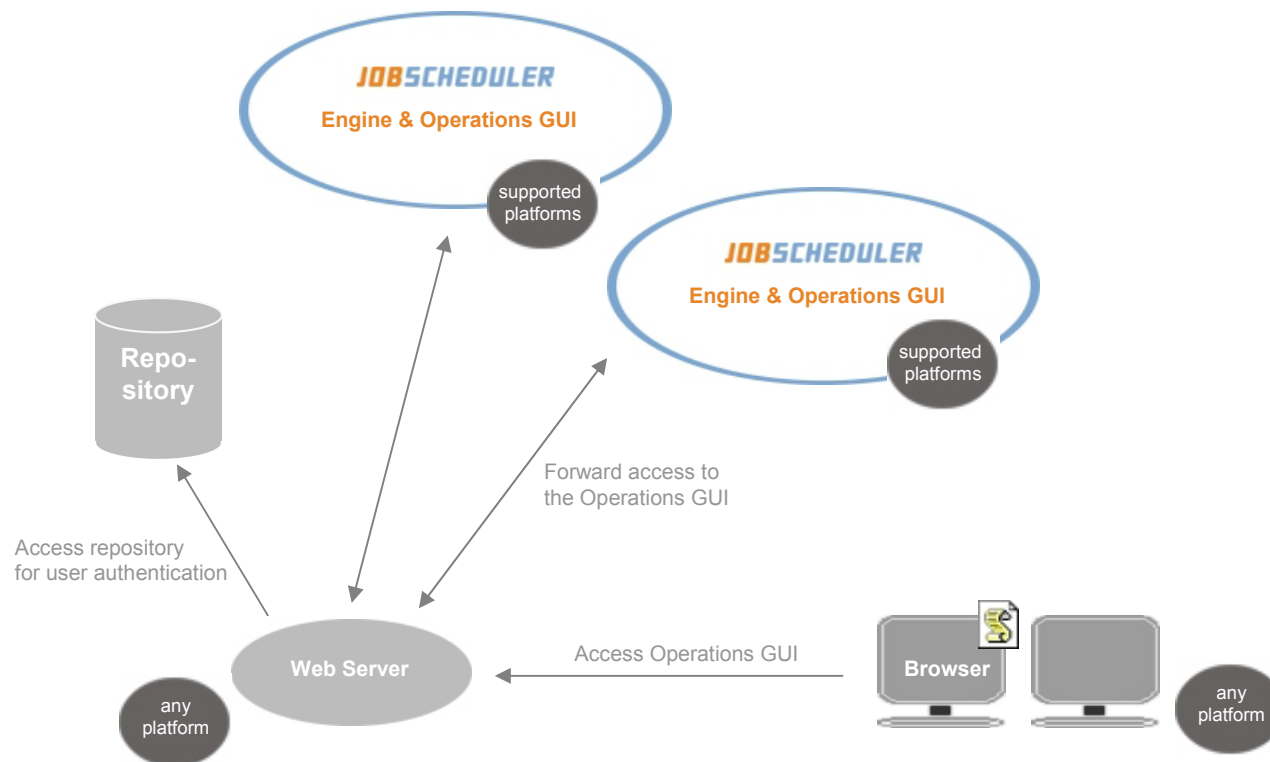
However, this does not apply for the use of terminal servers that assign the same address and for use of HTTP proxies.

By the way!

- If you have a network with dynamic IP assignment and lots of users then these authentication solutions would increase the required configuration efforts.

Security

External Authentication



At a Glance:

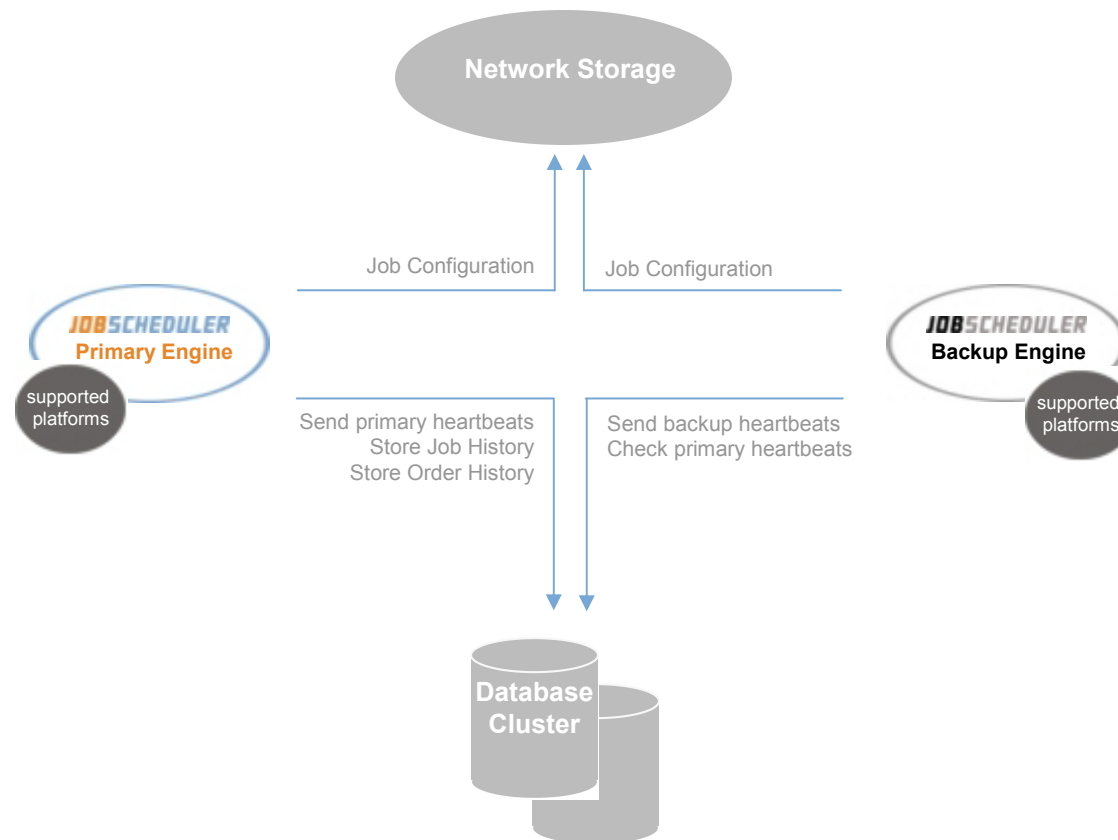
- You can choose to use a web server as a proxy between the clients and the Job Scheduler instances.
- An arbitrary number of Job Schedulers is addressed by one web server.
- LDAP, PAM and subsequent mechanisms supported by Apache can be used for individual authentication.

By the way!

- You could authenticate users against existing Unix accounts using passwd.

Failsafe Operation

Automatic Failover from Primary to Backup Job Schedulers



At a Glance:

- For automatic failover you have to set up one or more Backup Job Scheduler Instances.
- The Backup Instances use the same configuration and database as the Primary Job Scheduler Instance.
- Job configurations are loaded from a network storage, alternatively from a clustered database (Managed Jobs).
- The backup Job Schedulers constantly check if the Primary Job Scheduler is up and running and will assume control in case of failure of the primary instance.
- The Backup Job Schedulers will not execute any jobs, unless the heartbeats of the Primary Job Scheduler instance fail to be detected.

Open Source Job Scheduler



JOBSCHEDULER

Let the rabbit do the job

<http://jobscheduler.sourceforge.net>