Architecture and Mode of Operation
Open Source Users in more than 100 countries
100 commercial customers in 20 countries
Contents

Architecture Topics

- Components
- Supported Platforms and Databases
- Architecture
- Job Configuration
- Deployment
- Distributed Processing
- Security
- Failsafe Operation
Works with all supported platforms
Works with all supported platforms and Java
Any platform of your choice applicable
Works with all supported databases
Any database of your choice applicable

Managed Jobs are stored in a database
Javascript is required for Ajax GUI
Components

Overview

- Run-time components
- Design-time components
Run-time and design-time Components

- Run-time components for Job Execution
  - Jobs and job chains are executed in the JobScheduler Engine / Master in batch mode
  - The built-in Operations Center (JOC) is accessible for browsers and is used to control jobs and job chains, e.g. start, stop, suspend, resume, and to access log files

- Design-time components for Job Configuration
  - Hot Folders store job configurations and are monitored by JobScheduler Master for changes that are immediately detected and automatically applied by the respective JobScheduler instance
  - Define jobs and job chains with a client GUI
  - Store job configurations in XML files on disk
  - Manage job configurations with Hot Folders
Components

Run-time components

- JobScheduler Engine
- JOC: JobScheduler Operations Center
- Database
- File System: Live Folder
- Configuration Objects
Run-time Components

JobScheduler Engine

- Running in the background (Daemon / Service)
- C++ and Java and Scala
- Build-in http-Server or Jetty
- Internal and external API
- Modes of operation
  - Single instance, single server
  - Remote execution e.g. with Agents
  - Remote configuration (Supervisor)
- Load Sharing
- Failover
Run-time components

- Browser based application
- Ajax based application
- Connection to JobScheduler via TCP/IP
- Communication via JobScheduler Engine or Jetty
- Authentication via http
Run-time Components

Database

- Access via JDBC
- Usage
  - Restart capabilities
  - History and logs
  - Fail over
  - Load balancing
- House keeping jobs
  - Cleanup history
- Any host and platform
- Same database for multiple JobScheduler instances
Run-time Components

- JobScheduler Objects are flat files
- Structured by folders and subfolders
- XML based syntax of objects
- JobScheduler file watching active
- Version control by SVN, Git, …
Run-time Components

Configuration objects

- Job
- Job Chain
- Order
- Process Class
- Scheduler
- Lock
- Calendar (non-operated days)
- Parameter (include files)
Components

- JOE: JobScheduler Object Editor
- JITL: JobScheduler Integrated Template Library
- JID: JobScheduler Information Dashboard
- YADE: Managed File Transfer
Design-time Components

JOE: JobScheduler Object Editor
Design-time Components

- Predefined Jobs
- Platform independent
- File Operations
- Housekeeping
- File Transfer (YADE)
- Mail processing
- Agentless scheduling (SSH)
- Database: SQL*Plus, PL/SQL Jobs, JDBC connectivity
Design-time Components

- Managed File Transfer
- JobScheduler Adapter (YADE JITL Job)
- CLI
- Class Library
- FTP, SFTP, FTPS, local, CIFS, WebDav, HTTP, ...
- Transactional Transfer
- Atomic Transfer
- ...

YADE: Managed File Transfer
Design-time Components

- Single Point of Control
- History viewer
- Daily plan
- JOE integrated
- JOC integrated
- Event processor
- ...
Design-time Components

JID: JobScheduler Information Dashboard
Platforms and Databases

- **Supported Operating Systems** JobScheduler Master
  - Linux (Suse, RedHat, CentOS, …)

- **Supported Operating Systems** JobScheduler Universal Agent
  - Any platform that runs a Java Virtual Machine
  - Includes Windows, Linux, AIX, Solaris, HP-UX, …

- **Supported Databases**
  - DB2
  - Oracle
  - SQL Server
  - MySQL (MariaDB)
  - PostgreSQL
At a Glance:
- You can operate JobScheduler 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 JobScheduler includes a built-in Operations Center (JOC).
- You can use JOE to configure objects that are stored as flat files.

By the way!
- You can use any editor of your choice to manage object configurations in XML files.
Operation with a Database for compliant logging

Architecture

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 Center (JOC) per browser.
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 JobScheduler.
Using Hot Folders

Job Configuration

Store jobs and job chains

Hot Folders

JOE

JobScheduler Editor

supported platforms

Supported platforms

At a Glance:

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

By the way!

- You can still use at any time your favorite editor in order to configure jobs in the Hot Folders.
Remote Configuration Service: Deploy Objects

**Deployment**

**JobScheduler Editor (JOE)**
- Store jobs and job chains

**Hot Folders**
- Store jobs and job chains
- Monitor changes to jobs and job chains

**Workload**
- Replicate jobs and job chains to multiple workload instances

**Supervisor**
- any platform

**At a Glance:**
- JobScheduler RCS replicates object configurations from Live Folder to the Workload Instances.
- Some objects are deployed to all instances while other objects are deployed to specific Workload Instances.
- Workload Instances connect on start-up to a RCS Instance request for an update their objects in cache folder.
- Workload instances can execute jobs independently of a RCS Instance.
- Workload Instances re-connect automatically to a RCS Instance after network outages.
### Remote Execution

#### Distributed Processing

**At a Glance:**
- JobScheduler supports the execution of jobs via SSH on remote hosts. No SSH client is required.
- JobScheduler supports the execution of jobs on remote JobScheduler instances.
- Job chains for execution of individual job steps on remote hosts are supported.

---

[Diagram showing remote execution processes and supported platforms (Windows & Linux).]
Remote Execution by JobScheduler Universal Agents

Distributed Processing

At a Glance:
- JobScheduler supports the execution of jobs by Agents on remote hosts.
- JobScheduler supports the execution of jobs on remote JobScheduler instances.
- Job chains for execution of individual job steps on remote hosts are supported.

Host A
- Windows & Linux
- Execute job locally
- Engine

Host B
- any platform
- Execute jobs on a remote JobScheduler instance
- Universal Agent

Execute job locally
Agentless scheduling: Remote Execution by SSH

**Distributed Processing**

**At a Glance:**
- JobScheduler supports the execution of jobs via SSH on remote hosts. No SSH client is required.
- Job chains for execution of individual job steps on remote hosts are supported.

Cloud Engine

**Windows & Linux**

Host A

Execute job locally

Job

SSH Server

any platform

Host B

Execute job on a remote host that runs an SSH server

Job

Execute job locally

Host A

Host B
**Distributed Processing**

**Load Balancing**

**At a Glance:**
- For Load Balancing you can operate an arbitrary number of JobSchedulers in parallel.
- All Job Schedulers use the same job configurations and database.
- Job configurations are loaded from a network storage.
- 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 in parallel you can achieve high efficiency in your processing time.

![Diagram showing distributed processing architecture](image-url)

- **JobScheduler Supervisor**
  - Alternatively JobScheduler Supervisor or Network Storage can be used

- **Network Storage**
  - any platform

- **Job Configuration**
  - JobScheduler Engine
    - Windows & Linux
    - • Fetch enqueued orders
    - • Store Job History
    - • Store Order History

- **Database Cluster**
  - supported database
  - any platform
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.
- 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.
Basic Features
- Copy and move files by FTP, SFTP, FTPS to/from any host in the network.
- Execute commands by SSH.
- Password and public/private key authentication are supported for SFTP.
- The parameterization is effected by command line parameters, configuration files and job parameters.
- Logging and error handling are provided, errors are detected on a per file basis.
Advanced Features
- Transfer of partially matched files
- Recursive transfer of files in directories
- Atomic Transfer
- Transactional Behavior
- Check of transferred files for completeness (data integrity checking)
- Handling of empty files and renaming of files
- Use of alternative access data and credentials
- Support for UNC paths in target directories
- Track record of file transfers
- Configurable log structure and debug information
YADE follows a component architecture with layers.

The YADE Engine can handle transfer operations for any Virtual File System (VFS).

Currently VFS providers for FTP, SFTP, FTPS, WebDAV, CIFS, local files are available.

Transfer of files between VFS providers would not require any touchdown of files during up and running and will transfer operation.

YADE can be installed on one server and can handle file transfers for an arbitrary number of different hosts.
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.
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 password.
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.
Managed Jobs

Mixed Usage of Hot Folders and Managed Jobs

Access Job Editor GUI for configuration

<table>
<thead>
<tr>
<th>Job Editor GUI</th>
<th>supported platforms</th>
<th>Monitor Hot Folders and apply changes to Job Scheduler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Changes become active immediately</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Store jobs and job chains in XML-Files</td>
<td>Monitor Hot Folders and apply changes to Job Scheduler</td>
<td></td>
</tr>
<tr>
<td>Submit jobs/job chains to Hot Folders</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Job synchronizes Hot Folders automatically with the database</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Job Scheduler Database</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Access Managed Jobs GUI for configuration

| Managed Jobs GUI | supported platforms ||
|------------------|---------------------||
| Changes become active when submitted by Managed Jobs. | |
| Store jobs and job chains in Database | |
| Monitor Hot Folders and apply changes to Job Scheduler |
| Submit jobs/job chains to Hot Folders |

At a Glance:

- You can use both Hot Folders and the Managed Jobs GUI to manage configurations.
- Configurations in Hot Folders are automatically synchronized with Managed Jobs database.
- Configurations in the Managed Jobs database are automatically written to Hot Folders.
- 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.

Deprecated Feature
JobScheduler

Let the rabbit do the job

http://jobscheduler.sourceforge.net