

Volume

2



Technical Documentation

KETL[®] System Administration Guide 2.1

TECHNICAL DOCUMENTATION

KETL[®] System Administration Guide 2.1

© 1999 – 2008. All rights reserved worldwide.
Kinetic Networks, Inc.
33 New Montgomery Street • Suite 1200
Phone 415.358.5100 • Fax 415.358.5101

KETL is a registered trademark of Kinetic Networks, Inc.
UNIX is a registered trademark of The Open Group. Linux is a registered trademark of Linus Torvalds.
Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

Table of Contents

1	I N T R O D U C T I O N		5	K E T L O P E R A T I O N S	
	Scope	1		Start the KETL Server	9
	Audience	1		Run Jobs at the Operating System Level	10
	Related Documents	1		Any Job	10
				OS Job	11
2	O V E R V I E W			SQL Job	12
	Architecture	2		XML Job	12
	Job Executors	3		Enter the KETL Console Manager	13
	Dependency Model	3			
	Installation	4	6	K E T L C O N S O L E	
3	R E Q U I R E M E N T S			KETL Server	14
	System Requirements	5		Connect to the Metadata Database	15
	Operating System	5		Check Server Status	15
	Database	5		Pause the Server	16
	Backwards Compatibility	6		Resume the Server	16
	Conformance to Industry Standards	6		Shut Down the Server	17
	Product Requirements	6		Restart the Server	18
	Internationalization	6		Display Project List	18
	Installed Components	6		Display Server List	19
	Other Components	7		Display Available Commands	19
	Bundled Software	7		Exit the KETL Console Manager	19
	Documentation and Help System	7		Job Control	20
				Import a Job Definition	20
				Export a Job Definition	21
4	S E C U R I T Y			Display a Job Definition	21
	Database Connection	8		SQL Script	21
	Access Control List	8		XML	21

6	K E T L C O N S O L E		
	Import a Parameter List	22	
	Export a Parameter List	22	
	Start a Job	22	
	Submit a Job	22	
	Pause a Job	23	
	Resume a Job	23	
	Cancel a Job	23	
	Restart a Job	23	
	Skip a Job	24	
	Delete a Job	24	
	Check Status of a Job	25	
	Quick Reference	26	
	KETL Server	26	
	Job Management	27	
	Parameter List Management	27	
7	P E R F O R M A N C E C O N S I D E R A T I O N S		
	Hardware	28	
	Database	28	
	Parallel Jobs	28	
	KETL Clusters	29	
	Logging Level	29	
	Shared Machines	29	
8	B A C K U P A N D R E C O V E R Y		
	Backup		30
	Recovery		30
9	C O N F I G U R A T I O N P A R A M E T E R S		
	KETL Server		31
	Metadata Database		31
10	T R O U B L E S H O O T I N G		
	Job Instance		32
	Job Dependencies		32
	Job Output		33
	Log Files		33

Introduction

Kinetic Networks[®], Inc. (KNI) has developed a flexible and robust Java application to extract, transform, and load data (ETL). This application, named KETL[®], allows management of complex manipulation of data while leveraging an open source data integration platform. KETL is a metadata-driven solution that is scalable and platform-independent. The engine is built upon an open, multi-threaded, XML-based architecture. As such, complex ETL transformations can be executed quickly and efficiently.

Scope

This document explains how to perform basic administrative tasks on the KETL server. This document assumes that you have successfully installed KETL on your system with an initial metadata repository. Although this document provides examples in XML, it does not detail the XML job definition language.

Audience

This document is for the operations and administrative staff, typically those who install and configure enterprise software. Specifically, this document is for ETL developers, database administrators, and the operations team.

Related Documents

Refer to the following documents as needed:

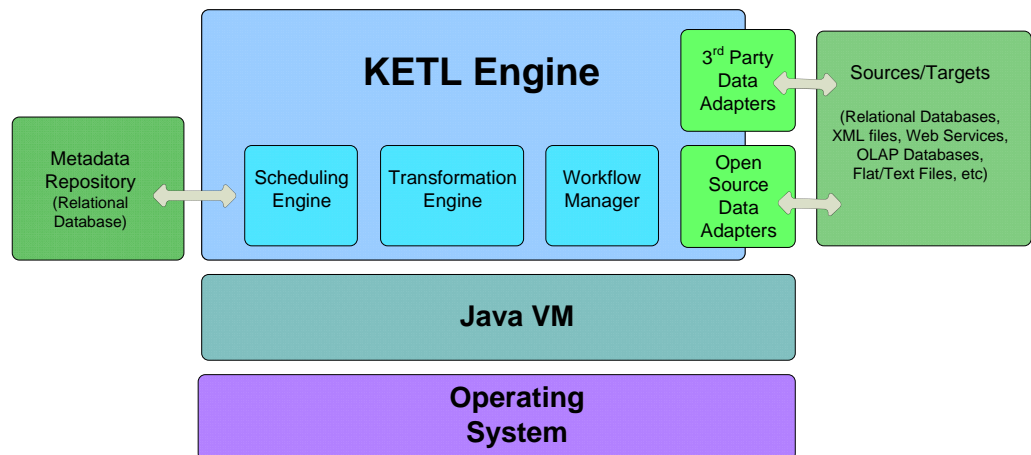
- KETL Installation Guide
- KETL Developer's Guide

Overview

The main goal of KETL is to allow for management of complex manipulation of data quickly and efficiently on an open source platform. This chapter provides an overview of the architecture and installation of KETL.

Architecture

KETL is platform-independent and can be installed on any operating system, as long as the appropriate Java runtime environment is in place. The following diagram shows the KETL engine as a layer on top of the Java virtual machine, along with different components, both optional and required.



The metadata repository connects to the KETL engine, with data flowing in both directions. Optional data adapters can also plug into KETL, with data coming from various sources and going to different targets.

Job Executors

The KETL engine consists of a multi-threaded server that manages various job executors. Each executor performs a specific function. The job executors fall into the following categories:

Executor Category	Description
SQL	Executes a hand coded SQL statement via JDBC.
OS	Executes an OS level command.
XML	Executes an XML defined job.
Sessionizer	Parses web log files.
Empty	Simple pass through mechanism, to assist in job management.

KETL combines a Java-based execution and scheduling manager with modular job execution architecture. All KETL programs, schedules, configuration data, and logs are stored within a central data repository. This repository allows for extensive reporting and integration with other metadata systems.

Dependency Model

Jobs are defined using XML and can be executed manually from the command line or through the KETL server. If using the KETL server, the job details must be defined in the KETL metadata repository and any associated job dependencies.

The dependency model used by KETL is a simple parent-child tree mechanism. This model allows for a high degree of parallelism, which is throttled according to the number of job executors allowed per server.

Dependencies can either be critical or non-critical. A critical dependency means that the job must wait for successful completion of another job before it executes. A non-critical dependency means that the job can still execute, even following an unsuccessful job.

Installation

You can install the KETL server on any platform, as long as you have the appropriate Java run-time environment and a relational database to store the KETL metadata. In general, you can use any relational database that supports row-level locking. For ease of use, however, KETL includes scripts for both PostgreSQL and Oracle, if you choose to use either of these databases.



Note

Refer to the KETL Installation Guide for more information.

Requirements

This chapter describes the [system requirements](#) as well as [product requirements](#) for KETL.

System Requirements

This section describes the following system requirements:

- [Operating System](#)
- [Database](#)
- [Backwards Compatibility](#)
- [Conformance to Industry Standards](#)

Operating System

The KETL server runs on any operating system that supports a Java run-time environment, version 1.5 or higher.

Database

To store the KETL metadata, you can use any relational database that supports row-level locking. KETL, however, has been tested on the following databases:

Database	Version
PostgreSQL	8.0 and higher
Oracle	9.1 and higher
MySQL	5.1 or higher using InnoDB for storage

In addition, KETL requires a thin client JDBC driver to connect to the appropriate metadata database. After you set up your metadata database, KETL also requires support for the procedural language.

Backwards Compatibility

Because all jobs are written in XML, they are backwards compatible. If a parameter is not used, then the system ignores it.

Class libraries are not backwards compatible and must be re-installed.

Conformance to Industry Standards

As per industry standards, KETL uses XML, JDBC, and SOAP.

Product Requirements

This section describes the following product requirements:

- [Internationalization](#)
- [Installed Components](#)
- [Other Components](#)
- [Bundled Software](#)
- [Documentation and Help System](#)

Internationalization

To date, KETL has not been fully internationalized, as it only supports the English language. When KETL does fulfill requirements for internationalization, KNI may look into the possibility of localization, by adding special features for a specific region (such as time zone and currency).

Installed Components

Some jar files have been included for installed components, but only those files that have licensing to allow for distribution. For example, the thin client JDBC driver for PostgreSQL has been included while drivers for other databases have been excluded.



Note

Refer to the KETL Installation Guide for more information.

Other Components

Other components, which may or may not be optional, have not been included in KETL. Complex transformations, for example, are recommended for your installation, but have been excluded due to licensing restrictions. This class library, however, is usually included in your Java installation.



Note

Refer to the KETL Installation Guide for more information.

Bundled Software

At this time, KETL does not include a metadata repository, but scripts have been included for both Oracle and PostgreSQL, if you choose to use either of these databases.

Documentation and Help System

The KETL documentation set consists of the following documents:

- KETL Installation Guide
- KETL System Administration Guide
- KETL Developer's Guide

The KETL system also has a simple help system, which displays available commands with syntax.

Security

This section describes the following security issues:

- [Database Connection](#)
- [Access Control List](#)

Database Connection

When connecting to the database, KETL can be configured to authenticate user ID and password. If the password is defined as type PASSWORD, then KETL automatically encrypts the string in the metadata database. Otherwise, KETL does not encrypt the string.

The following example shows data from a table that has two encrypted fields:

640	185	PASSWORD	vUpAqWmBQ2XN4/aFpabbw==
2,364	592	PASSWORD	vUpAqWmBQ2XN4/aFpabbw==

Access Control List

An access control list is a list of permissions attached to an object. This list describes who has access and what operations are allowed on the object. The list is a data structure (usually a table) with entries that specify individual user or group rights. These rights (or permissions) determine, for example, who has access to read, write, or execute an object.

To set user or group permissions, you must do so at the operating system. KETL does not define these privileges.

KETL Operations

This section describes the following KETL operations:

- [Start the KETL Server](#)
- [Run Jobs at the Operating System Level](#)
- [Enter the KETL Console Manager](#)

This section provides sample screenshots for reference only.



Important

For a Windows[®] environment, all commands have the .bat extension. For a UNIX[®]/Linux[®] environment, the commands do not have this extension. For example, the ketl_start command is ketl_start.bat in Windows. For the purpose of clarity, this document uses the UNIX/Linux version of the command.

Start the KETL Server

To start the KETL server, complete the following steps:

1. Log in as the dedicated operating system user for KETL.
2. Go to the following directory:

```
$KETDIR
```

3. Enter the following command:

```
ketl_start
```

This command starts the server and protects it from terminating after the user logs out.

```
[ketl@localhost ~]# ketl_start
Starting...
nohup: appending output to 'nohup.out'
Started.
[ketl@localhost ~]# _
```

Note

The KETL server remains running until you shut it down, so you do not need to start the server on a daily basis.

The following example shows that the ETL daemon is still running:

```
[ketl@localhost log]$ ps -f | grep ETL
ketl      3102      1 14 16:35 tty1      00:00:12 /usr/java/jdk1.5.0_10/bin/java -
server -XX:+UseParallelGC -Xms128m -Xmx1024m -Djava.awt.headless=true -Dlog4j.co
nfiguration=file:/home/ketl/ketl_dir/KETLBuild/conf/KETL.log.properties -classpa
th /home/ketl/ketl_dir/KETLBuild/lib/axis-ant.jar:/home/ketl/ketl_dir/KETLBuild/
lib/axis.jar:/home/ketl/ketl_dir/KETLBuild/lib/commons-discovery-0.2.jar:/home/k
etl/ketl_dir/KETLBuild/lib/commons-logging-1.0.4.jar:/home/ketl/ketl_dir/KETLBuild
lib/hsqldb.jar:/home/ketl/ketl_dir/KETLBuild/lib/jaxrpc.jar:/home/ketl/ketl_dir
/KETLBuild/lib/je-3.2.13.jar:/home/ketl/ketl_dir/KETLBuild/lib/junit.jar:/home
/ketl/ketl_dir/KETLBuild/lib/KETLGeoIP.jar:/home/ketl/ketl_dir/KETLBuild/lib/KET
LGPLExtensions.jar:/home/ketl/ketl_dir/KETLBuild/lib/KETL.jar:/home/ketl/ketl_dir
/KETLBuild/lib/KETLOracleExt.jar:/home/ketl/ketl_dir/KETLBuild/lib/KETLPostGreE
xt.jar:/home/ketl/ketl_dir/KETLBuild/lib/KETLSOAPExt.jar:/home/ketl/ketl_dir/KET
LBuild/lib/KETLXMLExt.jar:/home/ketl/ketl_dir/KETLBuild/lib/log4j-1.2.9.jar:/hom
e/ketl/ketl_dir/KETLBuild/lib/postgresql.jar:/home/ketl/ketl_dir/KETLBuild/lib/s
aaj.jar:/home/ketl/ketl_dir/KETLBuild/lib/saxon8-dom.jar:/home/ketl/ketl_dir/KET
LBuild/lib/saxon8.jar:/home/ketl/ketl_dir/KETLBuild/lib/saxon8-xpath.jar:/home/k
etl/ketl_dir/KETLBuild/lib/tools.jar:/home/ketl/ketl_dir/KETLBuild/lib/wsdl4j-1.
5.1.jar:/usr/java/jdk1.5.0_10/lib/tools.jar ETLDaemon
ketl      3137    2516  0 16:37 tty1      00:00:00 grep ETL
[ketl@localhost log]$ _
```

Run Jobs at the Operating System Level

You can run the following types of jobs at the operating system level:

- [Any Job](#)
- [OS Job](#)
- [SQL Job](#)
- [XML Job](#)

Be sure that you are in the correct directory so that KETL can find the job to run. Otherwise, specify the complete path of the job when you run the command.

**Important**

Confirm that you are at the operating system when you run these commands. Otherwise, if you try to use these commands in the KETL Console Manager, you receive error messages because it does not recognize the commands.

Any Job

To run any job, enter the following command:

```
ketl_runjob <job name>
```

This command works for all job types (OS, SQL, and XML).

The following example shows that MyJob.xml completed successfully.

```
[ketl@localhost xml]$ ketl_runjob MyJob.xml
[INFO]Thu May 17 07:37:55 EDT 2007 - [Thread[main,5,main]] Executing file /home/
ketl/ketl_dir/KETLBuild/examples/xml/MyJob.xml
[INFO]Thu May 17 07:38:04 EDT 2007 - [Thread[main,5,main]] KETL Version 2.1.12 P
roduction, ©2007 Kinetic Networks Inc.
[INFO]Thu May 17 07:38:04 EDT 2007 - [Thread[main,5,main]] Cache engine com.kni.
etl.ketl.lookup.SleepycatIndexedMap
[INFO]Thu May 17 07:38:05 EDT 2007 - [Thread[main,5,main]] Plugin com.kni.etl.ke
tl.transformation.geopip.GeopIPTransformation enabled.
[INFO]Thu May 17 07:38:54 EDT 2007 - [Thread[main,5,main]] Total execution time:
49.367 seconds
[INFO]Thu May 17 07:38:54 EDT 2007 - [Thread[main,5,main]] Job complete.
[ketl@localhost xml]$ _
```

OS Job

To run an OS job, enter the following command:

```
ketl_osrunjob <job name>
```

The following example shows that MyJob.xml completed successfully.

```
[ketl@localhost xml]$ ketl_osrunjob MyJob.xml
[INFO]Thu May 17 07:53:15 EDT 2007 - [Thread[main,5,main]] KETL Version 2.1.12 P
roduction, ©2007 Kinetic Networks Inc.
[INFO]Thu May 17 07:53:16 EDT 2007 - [Thread[main,5,main]] Cache engine com.kni.
etl.ketl.lookup.SleepycatIndexedMap
[INFO]Thu May 17 07:53:16 EDT 2007 - [Thread[main,5,main]] Plugin com.kni.etl.ke
tl.transformation.geopip.GeopIPTransformation enabled.
[INFO]Thu May 17 07:54:05 EDT 2007 - [Thread[main,5,main]] Total execution time:
49.302 seconds
[INFO]Thu May 17 07:54:05 EDT 2007 - [Thread[main,5,main]] Job complete.
Defaulting cache prefix to Console
[ketl@localhost xml]$ _
```

SQL Job

To run a SQL job, enter the following command:

```
ketl_sqlrunjob <job name>
```

XML Job

To run an xml job, enter the following command:

```
ketl_xmlrunjob <job name>
```

The following example shows that FileWriter.xml successfully completed:

```
[INFO]Thu May 17 08:03:45 EDT 2007 - [Thread[main,5,main]] - Starting threads
[INFO]Thu May 17 08:03:45 EDT 2007 - [Thread[main,5,main]] Threads initialized
[INFO]Thu May 17 08:03:45 EDT 2007 - [FileWrite(0)-Status(0)] Reading file exampl
es/data/CSV.txt
[INFO]Thu May 17 08:03:45 EDT 2007 - [FileWrite(0)-FileWrite] Final Throughput S
tatistics(Records Per Second)
    Overall Read: 8
    Average Read: 8
    Total Records Read: 8
    Overall Write: 8
    Average Write: 8
    Total Records Written: 8
    Thread Statistics
    -----
    Status, Type:com.kni.etl.ketl.reader.NIOFileReader [1 of 1]: 8, errors:
0, timing: N/A
    Target, Type:com.kni.etl.ketl.writer.FileWriter [1 of 1]: 8, errors: 0,
timing: N/A
[INFO]Thu May 17 08:03:45 EDT 2007 - [Thread[main,5,main]] Job details logged to
log/FileWrite.0.joblog
[INFO]Thu May 17 08:03:45 EDT 2007 - [Thread[main,5,main]] Total execution time:
1.141 seconds
[INFO]Thu May 17 08:03:45 EDT 2007 - [Thread[main,5,main]] Job complete.
Defaulting cache prefix to Console
[ketl@localhost xml]$ _
```

Enter the KETL Console Manager

While you can run jobs at the operating system, you can also use the KETL Console Manager. The KETL Console is a command line console that allows for basic administration of the server and metadata. The console communicates with the server via the metadata repository using JDBC.

To start the KETL Console, enter the following command:

```
ketl_ctl
```

```
[ketl@localhost xml]$ ketl_ctl
[INFO]Tue May 15 22:14:41 EDT 2007 - [Thread[KETL Console,5,main]] KETL Version
2.1.12 Production, ©2007 Kinetic Networks Inc.
[INFO]Tue May 15 22:14:42 EDT 2007 - [Thread[KETL Console,5,main]] Cache engine
com.kni.etl.ketl.lookup.SleepycatIndexedMap
[INFO]Tue May 15 22:14:42 EDT 2007 - [Thread[KETL Console,5,main]] Plugin com.kn
i.etl.ketl.transformation.geoip.GeoIPTransformation enabled.
KETL Console
->_
```

Note

The console does not have to be run on the same machine running the KETL server.

KETL Console Manager

The KETL Console Manager allows you to manage the KETL server as well as jobs. This section provides sample screenshots for reference only.

This chapter assumes that you have already started the KETL Console Manager. If not, refer to [Enter the KETL Console Manager](#) for more information.

KETL Server

By using the KETL Console Manager, you can complete the following KETL server operations:

- [Connect to the Metadata Database](#)
- [Check Server Status](#)
- [Pause the Server](#)
- [Resume the Server](#)
- [Shut Down the Server](#)
- [Restart the Server](#)
- [Display Project List](#)
- [Display Server List](#)
- [Display Available Commands](#)
- [Exit the KETL Console Manager](#)

Connect to the Metadata Database

KETL can connect to any database server defined in the KETLservers.xml file. If you do not specify a server name, KETL automatically connects to the default server defined in this file. To connect to the default server, enter the following command:

```
-> connect
```

Otherwise, to connect to a different server, enter the following command:

```
-> connect <server name><user name>
```

For example, to connect to the prod01 server as the ketl user:

```
-> connect prod01 ketl
```

The following example shows connection to the default server:

```
[ketl@localhost xml]$ ketl_ctl
[INFO]Wed May 16 07:11:18 EDT 2007 - [Thread[KETL Console,5,main]] KETL Version
2.1.12 Production, ©2007 Kinetic Networks Inc.
[INFO]Wed May 16 07:11:18 EDT 2007 - [Thread[KETL Console,5,main]] Cache engine
com.kni.etl.ketl.lookup.SleepycatIndexedMap
[INFO]Wed May 16 07:11:19 EDT 2007 - [Thread[KETL Console,5,main]] Plugin com.kn
i.etl.ketl.transformation.geoip.GeoIPTransformation enabled.
KETL Console

->connect
Connected to localhostPG
->_
```

Check Server Status

To check the server status at any point, enter the following command:

```
-> status
```

The following example shows the current server status of localhost.

```
->status
KETL Cluster Status
  Registered Servers: 1
  Alive Servers      : 1
  Pending Jobs

  Server   : localhost
  Status   : Active
  Start Time: 2007-05-15 21:43:34.351
  Last Ping : 2007-05-15 21:54:17.189522
  Executors (Stats)
    SQL: (Total: 2)
    KETL: (Total: 2)
    XMLSESSIONIZER: (Total: 1)
    OSJOB: (Total: 2)
```

Pause the Server

To pause the KETL server, enter the following command:

```
-> pause <server ID>
```

The server does not pause any jobs currently running and does not execute any new jobs. After pausing the server, you can check the status for confirmation.

The following example shows that the localhost server has been paused:

```
->pause 0
[INFO]Tue May 15 21:52:17 EDT 2007 - [Console] Request to pause server made
ID: 0, Name: localhost, Status: Paused, Last Ping: 2007-05-15 21:52:16.246194

->status
KETL Cluster Status
  Registered Servers: 1
  Alive Servers      : 0
  Pending Jobs

  Server   : localhost
  Status   : Paused
  Start Time: 2007-05-15 21:43:34.351
  Last Ping : 2007-05-15 21:52:16.246194
  Executors (Stats)
    SQL: (Total: 2)
    KETL: (Total: 2)
    XMLSESSIONIZER: (Total: 1)
    OSJOB: (Total: 2)
```

Resume the Server

To resume the server after pausing it, enter the following command:

```
-> resume <server ID>
```

The following example shows that the server resumed without problem:

```
->resume 0
[INFO]Tue May 15 21:53:34 EDT 2007 - [Console] Request to resume server made
ID: 0, Name: localhost, Status: Active, Last Ping: 2007-05-15 21:53:28.732073

->status
KETL Cluster Status
  Registered Servers: 1
  Alive Servers      : 1
  Pending Jobs

  Server   : localhost
  Status   : Active
  Start Time: 2007-05-15 21:43:34.351
  Last Ping : 2007-05-15 21:54:17.189522
  Executors (Stats)
    SQL: (Total: 2)
    KETL: (Total: 2)
    XMLSESSIONIZER: (Total: 1)
    OSJOB: (Total: 2)

->_
```

Shut Down the Server

To shut down the KETL server, enter the following command at the KETL console manager:

```
-> shutdown <server ID>
```

Each server has a unique server ID and must be specified upon shutdown. For example, to shut down the server associated with server ID 0, enter the following command:

```
-> shutdown 0
```

The following example shows the shutdown of server 0:

```
->shutdown 0
[INFO]Tue May 15 21:57:54 EDT 2007 - [Console] Request to shutdown server made,
waiting for confirmation
..
[INFO]Tue May 15 21:57:58 EDT 2007 - [Console] Server shutdown
->_
```

Optionally, you can also specify an immediate or normal shutdown by using the following syntax:

```
-> shutdown <server ID>{immediate|normal}
```

An immediate shutdown leaves the `job_log` table in a state that requires the job status to be modified. A normal shutdown allows current jobs to finish before the server shuts down. If you do not specify an immediate or normal shutdown, KETL shuts down normally.

Note

If the server does not have any executors defined, the server shuts down automatically. Before doing so, the server registers itself in the metadata and reports an error in the log files.

Restart the Server

To restart the server, enter the following command:

```
-> restart {immediate|normal}
```

The following example shows that confirmation is required before restarting the server:

```
->restart
[UNKNOWN]Wed May 16 07:17:45 EDT 2007 - [KETL Console] ERROR: Server not registered
[INFO]Wed May 16 07:17:45 EDT 2007 - [Console] WARNING: Server is registered and current status is alive
If a server has crashed then forcing startup might be ok
Force startup Y/N: y
->_
```

After you confirm that you want to force startup, you can check server status. The following example shows that the server started without problem:

```
->restart
[UNKNOWN]Wed May 16 07:17:45 EDT 2007 - [KETL Console] ERROR: Server not registered
[INFO]Wed May 16 07:17:45 EDT 2007 - [Console] WARNING: Server is registered and current status is alive
If a server has crashed then forcing startup might be ok
Force startup Y/N: y

->status
KETL Cluster Status
Registered Servers: 1
Alive Servers      : 1
Pending Jobs

Server   : localhost
Status   : Active
Start Time: 2007-05-16 07:01:22.076
Last Ping : 2007-05-16 07:18:33.814672
Executors (Stats)
  SQL: (Total: 2)
  KETL: (Total: 2)
  XMLSESSIONIZER: (Total: 1)
  OSJOB: (Total: 2)
->_
```

Display Project List

A project is a group of jobs. Each project has a unique project ID. To display the current list of projects, enter the following command:

```
-> project list
```

The following example displays a short project list, with just one entry:

```
->project list
Project(s)
ID      Description
--      -
1       Leah
->_
```

Display Server List

To display the current list of servers, enter the following command:

```
-> server list
```

The following example displays just one server in the list, along with the associated details:

```
->server list
ID: 0, Name: localhost, Status: Active, Last Ping: 2007-05-16 07:28:00.297279
-> _
```

Display Available Commands

To display the available commands in the KETL Console, enter the following command:

```
-> help
```

The following example shows all available commands and syntax:

```
->help
KETL Console allows remote administration of a KETL
cluster, the following commands are available.

      SHUTDOWN <SERVERID> {IMMEDIATE|NORMAL}
      STARTUP
      STATUS {CLUSTER|JOBS}
      JOB <NAME> {DEFINITION|DELETE|KILL|XMLDEFINITION|RESTART|SKIP|EXPORT <FI
LENAME>|IMPORT <FILENAME>|DEPENDENCIES|EXECUTE <PROJECTID> {MULTI} {IGNOREDEPEND
ENCIES}}
      RESTART {IMMEDIATE|NORMAL}
      QUIT
      CONNECT <SERVER|LOCALHOST> <USERNAME>
      HELP
      PARAMETERLIST <NAME> <EXPORT|IMPORT|DEFINITION> {FILENAME}
      PAUSE <SERVERID>
      RESUME <SERVERID>
      PROJECT LIST
      SERVER LIST
      RUN {LIST|RESET|<FILENAME>|LOADID <VALUE>}
      / {<REPEAT>} {<SECONDS BETWEEN REPEAT>}
```

Exit the KETL Console Manager

To exit the KETL Console, enter the following command:

```
-> quit
```

Upon exit, you return to your operating system environment.

Job Control

This section describes the following operations for jobs and parameter lists:

- [Import a Job Definition](#)
- [Export a Job Definition](#)
- [Display a Job Definition](#)
- [Import a Parameter List](#)
- [Export a Parameter List](#)
- [Start a Job](#)
- [Submit a Job](#)
- [Pause a Job](#)
- [Resume a Job](#)
- [Cancel a Job](#)
- [Restart a Job](#)
- [Skip a Job](#)
- [Delete a Job](#)
- [Check Status of a Job](#)

This section assumes that you have already started the KETL Console Manager.

Import a Job Definition

All job definitions must be imported before trying to run any commands on them. For example, if you try to execute a job without importing its definition, KETL returns an error because it does not recognize the job.

To import a job definition from an XML file, enter the following command:

```
-> job <job name> import <filename>
```

The following example imports the job definition from MyJob.xml:

```
->job MyJob import examples/xml/MyJob.xml
[INFO]Thu May 17 06:17:58 EDT 2007 - [Console] Reading file examples/xml/MyJob.xml
[INFO]Thu May 17 06:17:58 EDT 2007 - [Thread[KETL Console,5,main]] Creating new project Leah
MyJob
[INFO]Thu May 17 06:17:59 EDT 2007 - [Thread[KETL Console,5,main]] Creating job MyJob
Done
->_
```

If you make any changes to the file, you must re-import the job definition. Otherwise, KETL does not recognize any new changes.

The following example shows that the update was successful:

```
->job MyJob import examples/xml/MyJob.xml
[INFO]Thu May 17 07:05:53 EDT 2007 - [Console] Reading file examples/xml/MyJob.xml
MyJob
[UNKNOWN]Thu May 17 07:05:54 EDT 2007 - [KETL Console] Updating job MyJob
Done
->_
```

Export a Job Definition

Before deleting any job, you can export the job definition and save it for later use.

To export a job definition to an XML file, enter the following command:

```
-> job <job name> export <filename>
```

The following example shows that MyJob was exported to NewJob:

```
->job MyJob export NewJob.xml
MyJob
Export Jobs:
  MyJob
Done.
->_
```

Display a Job Definition

You can display job definitions in SQL script or XML. Be sure that the job has been imported into the metadata before you display any definitions. Otherwise KETL does not recognize the job and cannot display the definition.

SQL Script

To display a job definition in SQL script, enter the following command:

```
-> job <job name> definition
```

XML

To display a job definition in XML, enter the following command:

```
-> job <job name> xmldefinition
```

The following example shows the xml definition for MyJob:

```
->job MyJob xmldefinition
MyJob
<?xml version="1.0"?>
<ETL VERSION="1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<!-- Job: MyJob-->
<JOB DESCRIPTION="Run for awhile" DISABLE_ALERTING="N" ID="MyJob" NAME="MyJob" PROJECT="Leah" RETRY_ATTEMPTS="0" SECONDS_BEFORE_RETRY="10" TYPE="OSJOB">
<OSJOB>ping 127.0.0.1 -c 50 -W 1000</OSJOB>
</JOB>
</ETL>
->
```

Import a Parameter List

To import a parameter list from an XML file, enter the following command:

```
-> parameterlist <name> import <filename>
```

Export a Parameter List

To export a parameter list to an XML file, enter the following command:

```
-> parameterlist <name> export <filename>
```

Start a Job

If you are testing scripts, you may want to run a job individually, without worrying about job dependencies. You may also want to run a job without starting the KETL server. In both cases, you can run a single job in the current session foreground. To do so, enter the following command:

```
-> run <job name>
```

Be sure to enter the complete path of the job. For example:

```
-> run examples/xml/MyJob.xml
```

The following example shows that MyJob completed successfully:

```
->run examples/xml/MyJob.xml
[INFO]Thu May 17 07:09:00 EDT 2007 - [Console] Executing file examples/xml/MyJob.xml
[INFO]Thu May 17 07:09:49 EDT 2007 - [Thread[KETL Console,5,main]] Total execution time: 49.316 seconds
[INFO]Thu May 17 07:09:49 EDT 2007 - [Thread[KETL Console,5,main]] Job complete.
->_
```

Submit a Job

After you start the KETL server, you can submit a job to the work queue. By default, the server executes the job and all dependent jobs, unless configured not to do so.

To submit a job to the server, enter the following command:

```
-> job <job name> execute <project ID>
```

Refer to [Display Project List](#), if you need to look up the project ID of your job.

The following example shows the project ID lookup and job submission:

```
->project list
Project(s)
ID      Description
--      -
1       Leah

->job MyJob execute 1
MyJob
MyJob
Job submitted to server for direct execution.
```

Pause a Job

To pause a job during execution, enter the following command:

```
-> job <job name> pause
```

The KETL server does not pause any jobs already running. The current jobs finish running and all other jobs are put on hold.

Note

If you require a more immediate pause, then you can shutdown the server. To shut down the server, refer to the [Shut Down the Server](#) section.

Resume a Job

To resume a job after it has been paused, enter the following command:

```
-> job <job name> resume
```

Cancel a Job

To cancel a job, enter the following command:

```
-> job <job name> kill
```



Important

Canceling a job during execution may leave the tables in an intermediate state and should be avoided when possible.

Restart a Job

To restart a job, enter the following command:

```
-> job <job name> restart
```

The server picks up the job as soon as it has an available executor to run the job. Any dependent jobs run as soon as the job has successfully completed.

The following example shows that MyJob has been restarted, even though it may have already been executing:

```
->job MyJob restart
MyJob
Job MyJob is marked as executing, only do this if you are recovering from a server that has been killed! Y(Yes), N(No): y
Job: MyJob restarted
->_
```

Skip a Job

To skip a job, enter the following command:

```
-> job <job name> skip
```

The server sets the job status to “successful” and does not run. The following example shows a skipped job:

```
->job MyJob skip
MyJob
Job: MyJob skipped, it will appear as successful
->_
```

Delete a Job

Before deleting any job, be sure to export the job definition first. Then you have the option of re-importing the job definition for later use.

To delete a job from the metadata, enter the following command:

```
-> job <job name> delete
```

KETL requires confirmation before deleting the job. The following example shows the different options during the confirmation step:

```
->job MyJob delete
MyJob
WARNING: Deleting jobs can result in lost dependencies
It is recommended that you export job definition first!
Delete job MyJob Y(Yes), N(No), S(Skip all other jobs), A(Yes for All): _
```

The following example shows that MyJob has been deleted successfully:

```
->job MyJob delete
MyJob
WARNING: Deleting jobs can result in lost dependencies
It is recommended that you export job definition first!
Delete job MyJob Y(Yes), N(No), S(Skip all other jobs), A(Yes for All): y
[INFO]Thu May 17 13:07:11 EDT 2007 - [Console] Deleted job: MyJob
-> _
```

Check Status of a Job

To check status of a job, enter the following command:

```
-> status jobs
```

The resulting screen displays jobs that fall into the following categories:

Job Category	Description
Executing	Jobs currently assigned to executors and running.
Failed	Jobs that returned an error code during execution.
Just Failed	Jobs that returned an error code during execution, and must be restarted before pending dependent jobs can begin.
Paused	Jobs assigned to executors and not running.
Ready to Run	Jobs queued for the next available executors.

The following example shows that MyJob has not been assigned to an executor but is ready to run:

```
->status jobs
Executing
-----

Failed
-----

Just Failed
-----

Paused
-----

Ready To Run
-----
MyJob - OSJOB - 2007-05-17 13:22:10.429755 - - Not assigned - Ready to run

->
```

The following example shows that MyJob is currently executing:

```
->status jobs
Executing
-----
MyJob - OSJOB - 2007-05-17 07:22:36.736158 - - localhost - Executing

Failed
-----

Just Failed
-----

Paused
-----

Ready To Run
-----
```

Quick Reference

This section provides tables for quick reference when working with the KETL server and managing jobs or parameter lists.

KETL Server

Command Description	Syntax
Connect to the metadata database.	connect <server name><user name>
Check server status.	status
Pause the server.	pause <server ID>
Shut down the server.	shutdown <server ID>
Restart the server.	pause <server ID>
Resume the server.	resume <server ID>
Display project list.	project list
Display server list.	server list
Display available commands.	help
Exit the KETL Console Manager.	quit

Job Management

Command Description	Syntax
Import a job definition from an XML file.	job <job name> import <filename>
Export a job definition to an XML file.	job <job name> export <filename>
Display job definition in SQL.	job <job name> definition
Display job definition in XML.	job <job name> xmldefinition
Start a job in the foreground session.	run <job name>
Submit a job to the work queue.	job <job name> execute
Pause a job.	job <job name> pause
Resume a job.	job <job name> resume
Cancel a job.	job <job name> kill
Restart job that is not already running.	job <job name> restart
Skip a job.	job <job name> skip
Delete job.	job <job name> delete
Check status of a job.	status jobs

Parameter List Management

Command Description	Syntax
Import a parameter list from an XML file.	parameterlist <name> import <filename>
Export a parameter list to an XML file.	parameterlist <name> export <filename>

Performance Considerations

This section describes the following performance considerations:

- [Hardware](#)
- [Database](#)
- [Parallel Jobs](#)
- [KETL Clusters](#)
- [Logging Level](#)
- [Shared Machines](#)

Hardware

The more memory and CPUs you can allocate to KETL, the better the performance.

Database

Some jobs are database intensive and may require numerous reads on files or assigning of surrogate keys. Just running SQL or shell scripts, for example, results in numerous hits to the database. If your database has a limited sort size, then this kind of work may overtax the database and affect performance.

Parallel Jobs

Each job type allows for a defined number of jobs to be run in parallel. The KETL server, for example, may be set to run 10 SQL jobs at once, while a KETL job may only run four jobs at one time.

By default, the `server_executor` table sets the number of threads (executors) for each job type (SQL, OS, KETL, Sessionizer). If needed, you can update this table with new values. Be sure to test your system to confirm which values result in the best

performance. If a value is set too high, efficiency may be affected and slow down the entire system.

KETL Clusters

You can create clusters of KETL servers to improve on load balancing. More servers do affect performance, assuming that you are constrained by the server. For example, if you have one server running at 50%, then adding another server at 50% does not improve performance. If each KETL server is configured for optimum performance, then the cluster as a whole improves performance.

Logging Level

KETL allows you to determine the level of logging. If set to verbose (the most detailed level of logging), then system performance may be affected, but only slightly.

Shared Machines

If the KETL server and the metadata database reside on the same machine, system performance is not affected.

Backup and Recovery

KETL does not have specific tools to support backup and recovery of data. As such, KNI recommends that you follow standard best practice and backup your data on a regular basis.

Backup

KETL does not back itself up. If your system crashes, you lose the history and schedule of jobs, all information related to email notifications, and any customizations to executors.



Important

Be sure to make a secure backup of the system.xml file on a regular basis. Any external job files and metadata should also be backed up.

Recovery

If your system crashes, then you would need to recreate the database schema and re-import jobs into the metadata. All customized database settings would also need to be reset.

Configuration Parameters

You can update configuration parameters for the KETL server or for your metadata database.

KETL Server

All configuration files for KETL are located in the following directory:

`$KETLDIR/conf`

You can update any of the configuration parameters located in these files.

Metadata Database

For any fine tuning of the metadata repository, you need to update the parameters through the appropriate database tables. For example, you can update the number of job executors to be run or set the types of email notifications.

Troubleshooting

Almost any error that causes a job to fail may freeze the entire system, once all jobs are waiting on it (either directly or indirectly). To resolve this issue, you need to find the job that hit the error state. After doing so, you can fix the problem and restart the job. Otherwise, if you cannot fix the problem, then you can choose to skip the job.

To locate the failed job, you can use the following methods:

- Check for a failed job in job queue.
- Set the job to MULTI.
- Replace the transform step.
- Check the log file of the job.

Job Instance

When KETL runs a series of interdependent jobs, by default, it allows only one instance of that job. If it fails, the scheduler throws an exception because the job does not run twice. It is set to run just once.

In this situation, check the queue for a failed job.

Job Dependencies

If KETL runs 10 jobs, all in a string of dependencies, and the last one fails, then all 10 sit in the job queue. Even if the first nine are successful, the last job (which failed) leaves the entire job load in the queue. In other words, the entire load cannot run because the last job failed.

In this situation, you can set the job parameter to MULTI to create a multi-instance. In the KETL Console Manager, enter the following command:

```
-> job <job name> execute <project ID> multi
```

This command allows the other jobs in the load to run, even if a job fails within the load.

Job Output

Instead of writing to the database, you can alter the job to write the output to the console (temporarily) or even a null writer. You are essentially replacing the transform step. By doing so, you effectively eliminate jobs that do not cause an error.

To use this method, you replace the transform step and write the output to the console. If you do not hit an error with this job, then you can revert this job to its original state, and replace the transform step in the next job. You can systematically test each job, view the results directly on the screen, and move on to the next job until an error displays on the screen. When it does, you know the exact job that caused the error.

Log Files

Each job creates a log file and assigns a special ID (dmloadid), with the ID being unique to a job being run. This unique ID gets assigned each time it runs, unless it is being restarted.

The log file uses the following naming convention:

`jobID.dmloadID.joblog`

If you know which job is causing errors, you can check this log file. Be sure to check the stack trace.