

KETL Open Source Training

Kinetic Networks

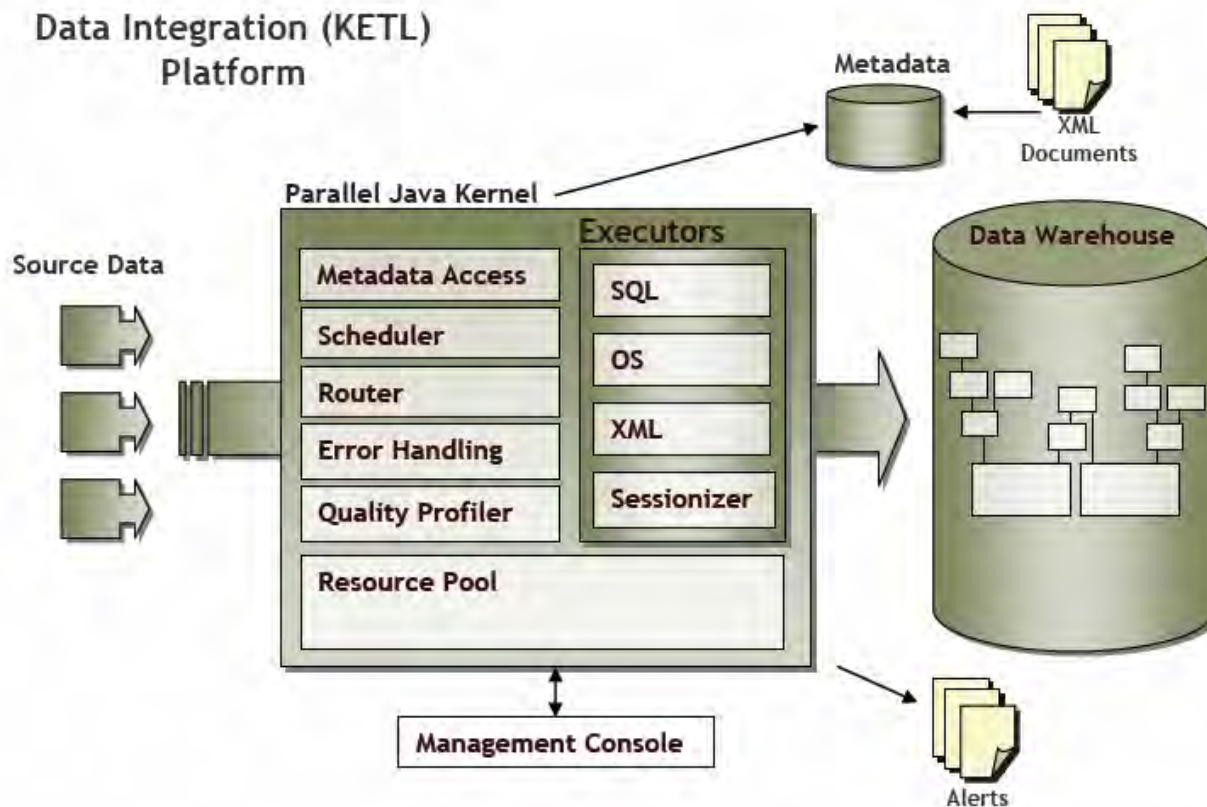
Last updated: June 29, 2004



Agenda

- KETL
 - Architecture
 - Java Kernel
 - Metadata
 - Executors
 - XML Documents
 - Example

KETL High Level Architecture



Executors

A KETL cluster consists of multiple servers, each server manages a configurable number of executors. Within the cluster work is passed to an available executor, which manages the execution and error handling of a job. New types of executors can easily be created but the four standard ones normally suffice.

SQL

- Executes Custom SQL
- Parameter Passing

KETL Log Sessionizer

- Intelligent Parsing of Dynamic URLs
- Highly Configurable Sessionization
- High Performance Bulk Loader
- Extendable Parallel Architecture (ex: Processes 10 million Hits in 160 seconds, hardware permitting)

Operating System

- Executes OS Commands
- Parameter Passing

XML

- Insulates Developer from SQL Coding
- Ability to define complex transformations
- Support for complex data sources
- Scalable expandable parallel architecture

All Jobs

Basic XML Layout

All jobs have a <JOB> tag

Jobs that depend on other jobs to complete Successfully use the <DEPENDS_ON> tag

Job that wait for other jobs to complete use the <WAITS_ON> tag

Relevant XML Tags and Attribute Values

<JOB> – Defined start and end of command

Type – Tells the engine what type of job it is

ID – Unique job name

Name – Short job name

Description – Detailed description

Project – Project that the job should belong to

Parameter_List – Default parameter list for job

Retry_Attempts – Allows the job to retry X times before failing

Seconds_Before_Retry – Seconds between retry attempts

XML Example

```
<JOB NAME="TRANS_IPS" ID="TRANSIPS" PROJECT="Demo Project" TYPE="OSJOB" RETRY_ATTEMPTS="0"
  SECONDS_BEFORE_RETRY="0" PARAMETER_LIST="tmpIPAddress">
  <DEPENDS_ON>WRITE_TRANS_IPS</DEPENDS_ON>
  <WAITS_ON>DEL_TRANS_IPS</WAITS_ON>
  !--- job details, following slides ---!
</JOB>
```

OS Job

Purpose

Executes a job on the machine running the KETL engine

Relevant XML Tags and Attribute Values

<OSJOB>—Defined start and end of command

Type = “OSJOB”—Tells the engine this is an OS Job

XML Example

```
<JOB ID="TRANSIPS"NAME="TRANS_IPS" PROJECT="Demo Project" TYPE="OSJOB"  
  PARAMETER_LIST="tmpIPAddress">  
<DEPENDS_ON>WRITE_TRANS_IPS</DEPENDS_ON>  
<OSJOB>  
/u02/DI/dnstran -translate -normal -exchange /u02/DI/tmpIPAddress.tmp  
</OSJOB>  
</JOB>
```

SQL Job

Purpose

Executes SQL on the database specified in the parameters

Relevant XML Tags and Attribute Values

<SQL>–Defines a collection of statements or an individual statement if no statements are defined

<STATEMENT>–The SQL statement to be executed

Type = “SQL”–Tells the engine this is a SQL Job

Parameter_List–Which database to run SQL against

Autocommit–Enables or disables explicit database transactions

XML Example

```
<JOB ID="TRUNCDEMO12" NAME="TRUNCATE_DEMO_1_2" PROJECT="Demo Project"
  TYPE="SQL" PARAMETER_LIST="stageDB">
  <SQL>
    <STATEMENT AUTOCOMMIT="FALSE">
      TRUNCATE TABLE stage.demo1
    </STATEMENT>
    <STATEMENT AUTOCOMMIT="FALSE">
      TRUNCATE TABLE stage.demo2
    </STATEMENT>
  </SQL>
</JOB>
```

Parameter Lists

Purpose

Allows the developer to override and manipulate parameters defined in the metadata.

Relevant XML Tags and Attribute Values

<PARAMETER_LIST>—Root of parameter list

<PARAMETER>—Parameter within the parameter list and associated value

Name = "SQL"—Name of parameter or parameter list

Parameter_List= "SubParSrc"—Allows the parameter list to contain parameters from other parameter lists, value can be a wildcard. Child values override parent values. Such that a value for parameter named filepath in the root will be overwritten by values in the child and if a parameter list has multiple children a value may occur more than once.

XML Example

```
<ETL>
  <PARAMETER_LIST NAME="param1">
    <PARAMETER NAME="PATH" PARAMETER_LIST="SubParDest"/>
  </PARAMETER_LIST>
  <PARAMETER_LIST NAME="SubParDest">
    <PARAMETER NAME="FILEPATH">c:\testing\weblogs\copy1</PARAMETER>
  </PARAMETER_LIST>
  <PARAMETER_LIST NAME="param2">
    <PARAMETER NAME="PATH" PARAMETER_LIST="SubParSrc*"/>
  </PARAMETER_LIST>
  <PARAMETER_LIST NAME="SubParSrc1">
    <PARAMETER NAME="SEARCHPATH">c:\testing\weblogs1\web*</PARAMETER>
  </PARAMETER_LIST>
  <PARAMETER_LIST NAME="SubParSrc2">
    <PARAMETER NAME="SEARCHPATH">c:\testing\weblogs2\web*</PARAMETER>
  </PARAMETER_LIST>
</ETL>
```

Sessionizer Job

Purpose

Parses a weblog or similar file and organizes hits into sessions and pageviews. Can be used for more than web logs if need be.

Relevant High Level XML Tags

<SESSIONIZER>—Root of the sessionizer job, contains global session parameters

<IDENTIFIERS>—Items within a weblog that can be used to identify a session

<LOG_FILES>—Weblog file structure

<DESTINATION>—Destination table structure

<PAGE_DEFINITIONS>—Characteristics that are use to identify pages within hits

<PAGE_PARAMETER_SETS>—Page parameters that should be kept

XML Example

```
<JOB NAME="SESS_WEBLOG" TYPE="XMLSESSIONIZER" PROJECT="Demo Project">
<SESSIONIZER WEBSERVERTYPE="APACHE" TIMEOUT="1800"
SKIPINSERTS="false" PEAKSESSIONSANHOUR="1000"
BATCHCOMMITSIZE="5000">
<LOG_FILES/>
<IDENTIFIERS/>
<DESTINATION/>
<PAGE_DEFINITIONS/>
<PAGE_PARAMETER_SETS/>
</SESSIONIZER>
</JOB>
```

Sessionizer Job – Sessionizer Tag

Purpose

Root of the sessionizer job, contains global session parameters.

Relevant XML Tags and Attributes

WEBSERVERTYPE – Apache or Netscape, Apache works for most web servers

TIMEOUT – Default session timeout, in seconds, normally 30 minutes

SKIPINSERTS – Used for testing, stops the sessionizer from inserting data into the db

PEAKSESSIONSANHOUR – Peak number of sessions an hour

BATCHCOMMITSIZE – Default number of records to insert at once

XML Example

```
<SESSIONIZER WEBSERVERTYPE="APACHE"  
    TIMEOUT="1800"  
    SKIPINSERTS="false"  
    PEAKSESSIONSANHOUR="1000"  
    BATCHCOMMITSIZE="5000"/>
```

Sessionizer Job –Identifiers Tag

Purpose

Items within a weblog that can be used to identify a session.

Relevant XML Tags and Attributes

<IDENTIFIER_SET>–A combination of identifiers that define session

<IDENTIFIER>–A session identifier, within an identifier set

PRIORITY–Allows an identifier to have precedence over another

TYPE–Identifies the type of identifier, MAIN (e.g.session cookie), PERSISTANT (e.g.multip-session cookie), IP_BROWSER (ipaddress browser combination)

OBJECTTYPE–Object in log file to pull identifier from

VARIABLENAME–Name of variable if needed

ENABLEFALLBACK–Allows sessionizer to revert to this cookie once a better matchis found.

EXPIREWHENBETTERMATCH –Disable any type of fallback

TIMEOUT–How long this identifier can be used to match a session

XML Example

```
<IDENTIFIERS>
  <IDENTIFIER_SET TYPE="MAIN" TIMEOUT="1800" PRIORITY="1">
    <IDENTIFIER OBJECTTYPE="IN_COOKIE" VARIABLENAME="JSESSIONID"/>
  </IDENTIFIER_SET>
  <IDENTIFIER_SET TYPE="MAIN" TIMEOUT="1800" PRIORITY="2">
    <IDENTIFIER OBJECTTYPE="URL_REQUEST" VARIABLENAME="JSESSIONID"/>
  </IDENTIFIER_SET>
  <IDENTIFIER_SET TYPE="PERSISTANT" TIMEOUT="1800" PRIORITY="3" ENABLEFALLBACK="TRUE">
    <IDENTIFIER OBJECTTYPE="IN_COOKIE" VARIABLENAME="DIUNIQID"/>
  </IDENTIFIER_SET>
  <IDENTIFIER_SET TYPE="IP_BROWSER" TIMEOUT="30" PRIORITY="4" EXPIREWHENBETTERMATCH="TRUE">
    <IDENTIFIER OBJECTTYPE="IP_ADDRESS"/>
  <IDENTIFIER OBJECTTYPE="USER_AGENT"/>
  </IDENTIFIER_SET>
</IDENTIFIERS>
```

Sessionizer Job –Destination Tag

Purpose

Destination table structure for sessions and pageviews.

Relevant XML Tags and Attributes

<SESSION>–Sessionable root tag

<SESSION_IN>–Session field tag

<HIT>–Hittable root tag

<HIT_IN>–Hit table field tag

NAME–Field name in destination table

SOURCE–Sourceobject, e.g. BYTES_SENT

TABLENAME–Table to insert into

PARAMETER_LIST–Database to insert into

HINT–Database optimization

CLASS –High performance bulk loading class, dependent on the destination database

XML Example

```
<SESSION TABLENAME="TMP_SESSION"
    CLASS="com.kni.etl.sessionizer.PGCopySessionDatabaseWriter"
    PARAMETER_LIST="destDB" HINT="/*+ APPEND */">
  <SESSION_IN NAME="TEMP_SESSION_ID" SOURCE="TEMP_SESSION_ID"/>
  <SESSION_IN NAME="BROWSER" SOURCE="BROWSER"/>
</SESSION>
<HIT TABLENAME="TMP_HIT" PAGESONLY="FALSE" PARAMETER_LIST="destDB">
  <HIT_IN NAME="ACTIVITY_DT" SOURCE="ACTIVITY_DATE_TIME"/>
  <HIT_IN NAME="BYTES_SENT" SOURCE="BYTES_SENT"/>
</HIT>
```

Sessionizer Job –Page Definitions Tag

Purpose

Series of searches used to identify a page.

Relevant XML Tags and Attributes

<PAGE_DEFINITION> – Page definition root tag

PRIORITY – Order in which searches will be applied to a request

DIRECTORY – Directorysearch string

HOSTNAME – Hostnamesearch string

PROTOCOL – Protocolsearch string

TEMPLATE – Page search string

STATUSCODES – Comma separated list of valid status codes

VALID – Used to negate the search

PAGE_PARAMETER_SET – Parameters associated with page

ID – ID to be put in database when found

XML Example

```
<PAGE_DEFINITIONS>
<PAGE_DEFINITION PRIORITY="1" DIRECTORY="" HOSTNAME="" PROTOCOL="" TEMPLATE="home"
  PAGE_PARAMETER_SET="Default" ID="1"/>
<PAGE_DEFINITION PRIORITY="2" DIRECTORY="" HOSTNAME="" PROTOCOL="" TEMPLATE="*.portal"
  PAGE_PARAMETER_SET="Default" ID="2"/>
<PAGE_DEFINITION PRIORITY="3" DIRECTORY="/home/" HOSTNAME="" PROTOCOL="" TEMPLATE=""
  PAGE_PARAMETER_SET="Default" ID="3"/>
<PAGE_DEFINITION PRIORITY="4" DIRECTORY="/" HOSTNAME="" PROTOCOL="" TEMPLATE=""
  PAGE_PARAMETER_SET="Default" ID="4"/>
</PAGE_DEFINITIONS>
```

Sessionizer Job –Page Parameter Sets Tag

Purpose

Defines how a pages parameters are to be cleansed.

Relevant XML Tags and Attributes

<PAGE_PARAMETER_SET> – Root tag for a set of parameters that can be associated with a page

<PAGE_PARAMETER_NAME> – Page parameter tag

NAME– Nameof parameter

REMOVE_VALUE – Keep or remove parameter value

REQUIRED – Requiredto match page

SEPERATOR – Name, value pair separator

REMOVE_PARAMETER – Remove parameter name

XML Example

```
<PAGE_PARAMETER_SETS>  
<PAGE_PARAMETER_SET NAME="Default">  
<PAGE_PARAMETER NAME="_pageLabel" REMOVE_VALUE="FALSE" REQUIRED="FALSE" SEPERATOR="=" REMOVE_PARAMETER="FALSE"/>  
</PAGE_PARAMETER_SET>  
</PAGE_PARAMETER_SETS>
```

Sessionizer Job – Log File Tag

Purpose

Defines the structure of the log file, based on tag definition of normal KETL file reader. Multiple files can be read at once in parallel, allowing support for clusters of web servers. The parameter list referenced must contain one or more SEARCHPATH parameter used to locate the weblogs.

Relevant XML Tags and Attributes

<LOG_FILE>–Log file root tag

<OUT>–Field tag root

NAME–Name of parameter

DATATYPE–Datatype of field

OBJECTTYPE–Object type of field, used by the identifier tags e.g REQUEST_STRING

QUOTESTART–Quote start string

QUOTEEND–Quote end string

PARAMETER_LIST–Parameter defining search path of files

DELIMITER–Default field delimiter

SKIPLINES–Number of initial lines to skip

FORMATSTRING–Format used to parse date time values

READORDERSEQUENCE–Defines order which rows from multiple source files will be read

READORDER–Direction of read

XML Example

```
<LOG_FILES>
  <LOG_FILE NAME="WeblogApache" PARAMETER_LIST="mylogfile" DELIMITER=" " SKIPLINES="0">
    <OUT NAME="COMMON_LOG_DATETIME" DATATYPE="DATE" FORMATSTRING="dd/MMM/yyyy:HH:mm:ssZ"
      QUOTESTART="[" QUOTEEND="]" READORDERSEQUENCE="1" R
      EADORDER="ASC" OBJECTTYPE="HIT_DATE_TIME"/>
    <OUT NAME="REFERRER" DATATYPE="STRING" OBJECTTYPE="REFERRER_URL"
      QUOTESTART="&quot;" QUOTEEND="&quot;"/>
    <OUT NAME="USER_AGENT" DATATYPE="STRING" OBJECTTYPE="USER_AGENT"
      QUOTESTART="&quot;" QUOTEEND="&quot;"/>
    <OUT NAME="REQUEST_METHOD" DATATYPE="STRING" OBJECTTYPE="REQUEST_METHOD"/>
    <OUT NAME="URL_REQUEST_NO_QUERY_STRING" DATATYPE="STRING" OBJECTTYPE="REQUEST_STRING"/>
    <OUT NAME="QUERY_STRING" DATATYPE="STRING" OBJECTTYPE="QUERY_STRING"/>
  </LOG_FILE>
</LOG_FILES>
```

Sessionizer Job –Guidelines

- Log files must be sorted by time – Slight discrepancies are handled but appending one log onto another will not work
- Sessionizer writes to the database at an extremely fast rate, archive redo logs or similar will throttle the process
- SMP Machines will improve performance
- A good understanding of what defines a page on your website is essential, examining the referrer URLs in your weblogis quick way to understand what makes up a valid page
- Only using IP Browser session identification is prone to error
- Setting IP Browser session identification to a long timeout will merge dial-up users
- Web users purge cookies, a cookie that regenerates based on logging in will improve data accuracy
- The easy way to test your log file definition is to use a KETL job
- Don't get carried away with page definitions, to many will slow things down

KETL Job

- Parameter Lists
- Worker Classes
 - File Classes –File manipulation
 - Database Classes –Database interaction
 - Inline Classes –Complex or optimized transformations
 - Other –Custom classes, special purpose classes
- QA Monitors

KETL Job –File Worker Classes

- File Classes
 - FileReader, NIOFileReader –Used to read files
 - NIOFileWriter, FileWriter –Used to write files
 - FTPFileReader –Used to read files remotely via ftp
 - FTPFileFetcher –Used to ftp files to a local machine
 - XMLReader –Used to read XML files, using an XSL transformation

KETL Job –Database Worker Classes

- Database Classes
 - JDBCReader –Executes one or more select statement
 - JDBCWriter –Batch data upserts into destination tables
 - JDBCIncrementor –Manages sequences for surrogate key generation
 - JDBCNonBatchedWriter –Non-Batched database writer, enabling support for raw and long data types
 - LookupTransformation –Batch database lookup

KETL Job –Inline Worker Classes

- Some Inline Classes
 - DeflateTransformation –Used to compress data –uses any compression algorithms available in the Java SDK
 - ConditionalStep –Applies conditional logic to an ETL job
 - ConsoleWriter –Outputs data to the console, mainly used for testing
 - StringTransformation –String transformations, extensible using java
 - DNSTransform –Parallel name resolver

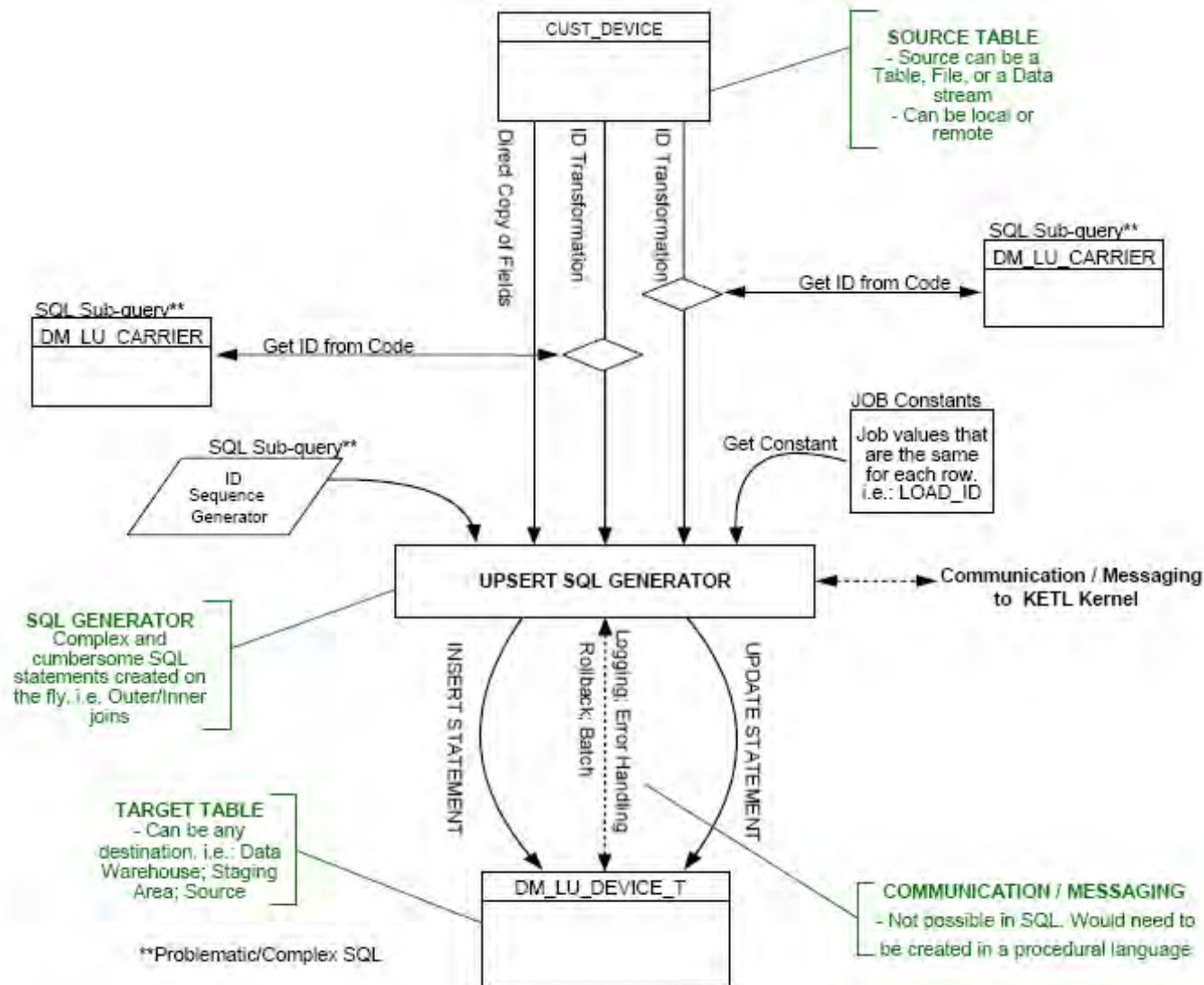
KETL Job –Other Worker Classes

- Other Classes
 - Building new classes to add additional functionality is relatively easy. All that is needed is a basic knowledge of java and batch processing. Parallel execution and job management is handled for you.

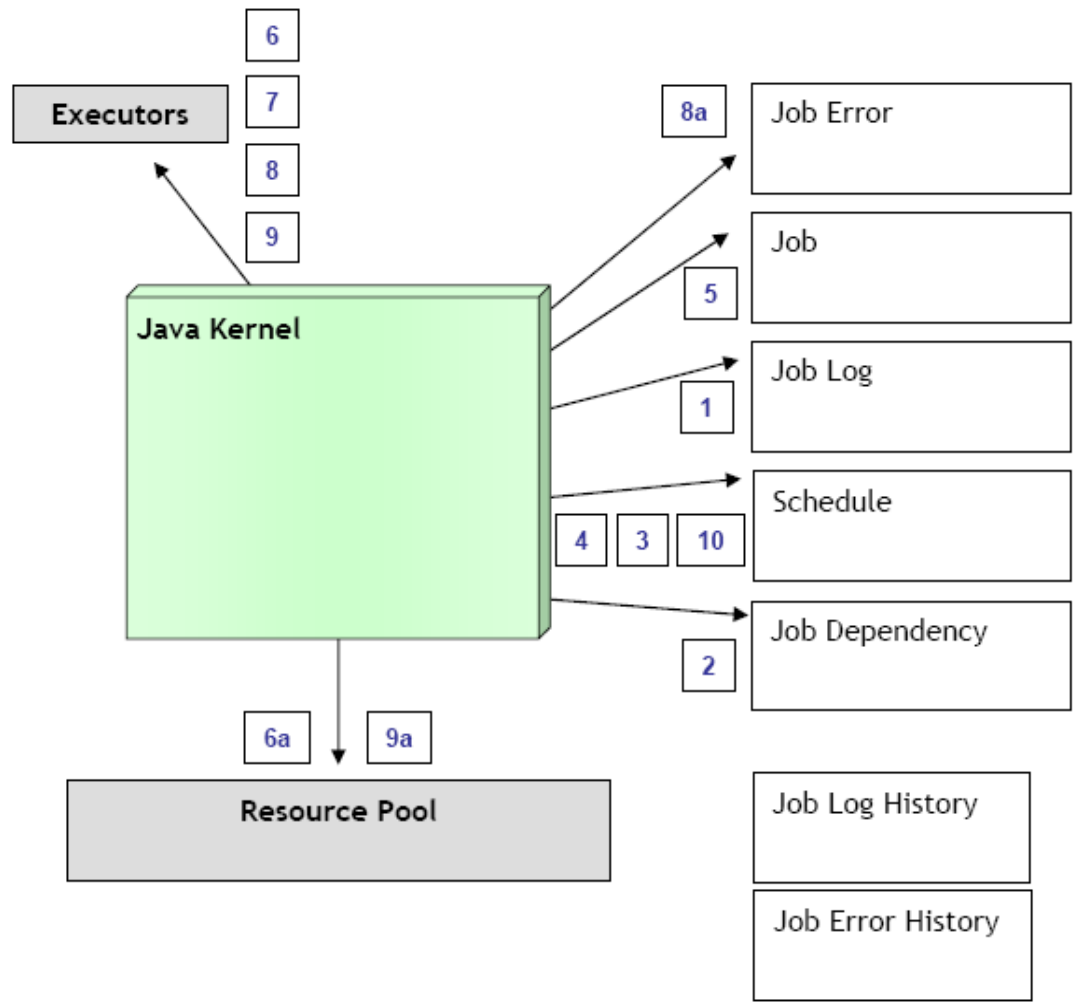
KETL Job Examples

- Read a weblog and output to screen
 - FileToScreen.xml
- Read a weblog and copy to file
 - FileToFile.xml
 - FileToFileX2.xml
- Read from DB and insert into File
 - DBToFile.xml
- Read a file and insert into DB with id
 - FileToDBWithKey.xml
- Read a file and upsertwith surrogate key and id
 - FileToDBWithMultiUpserts.xml
- Read a file, transform elements and write to a file
 - FileToFileModifyingString.xml

KETL Job Example



Metadata Kernel Interaction Flow



1. Poll for Loads
2. Fetch Jobs for a Load
3. Write and Update Job Information
4. Poll for Jobs
5. Fetch Job Details
6. Supply Job
- 6a. Request Resources
7. Poll for Job Status
8. Retrieve Messages
- 8a. Write and Send Notifications
9. Close Out Job
- 9a. Return Resources
10. Poll for Jobs