THREEGLAV

Threeglav.com

StreamHorizon - User Guide

Version 3.0.2

STREAM HORIZON

DataProcessingPlatform

TABLE OF CONTENTS

INTRODUCTION

GETTING STARTED

INSTALLATION REQUIREMENTS

Installation of StreamHorizon platform is simple and easy. Download installation archive and uncompress it to your hard drive in directory of your choice (we will refer to this directory as $ENGINE_HOME from now on).

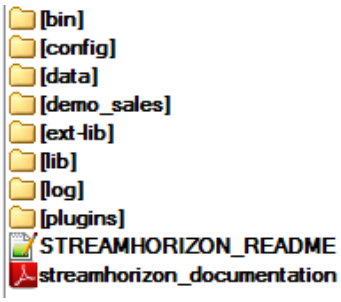
There are very few requirements that need to be met in order to run StreamHorizon.

1. Mainstream operating system (Linux, Windows, Solaris)
2. JDK 1.7+ (we recommend Oracle HotSpot)
3. Database that supports JDBC (most likely data produced by StreamHorizon will go into some kind of database)

Even if you intend to run multiple (clustered) StreamHorizon instances there is no need to install it more than once per physical machine.

INSTALLATION AND DIRECTORY STRUCTURE

After installing StreamHorizon platform to $ENGINE_HOME directory you will see following directory structure inside:



Directory name	Purpose
$ENGINE_HOME/bin/	All startup scripts should be placed here. There are few default scripts that can be used for starting single instance. If you are adding new startup scripts always place them in this directory.
$ENGINE_HOME/config/	All configuration files are here: • What engine should log and at what level • Here is where you copy your main engine configuration file(s) • Configuration for clustered caches used by engine
$ENGINE_HOME/data/	This is engine private directory used for housekeeping. You should not add, delete or modify anything inside.
$ENGINE_HOME/demo_sales/	Demo feature showing capabilities of engine (performance and functionality).
$ENGINE_HOME/ext-lib/	Additional dependencies needed by plugins.
$ENGINE_HOME/lib/	External dependencies needed by engine are placed here. You should not add, delete or modify anything inside this directory.
$ENGINE_HOME/log/	Engine log files will be found here.
$ENGINE_HOME/plugins/	Java plugins. Plain *.java files can be placed here and engine will compile and use them (when configured to do so).

STREAMHORIZON CONFIGURATION FILES

ENGINE CONFIGURATION FILE (ENGINE-CONFIG.XML)

The main configuration file for StreamHorizon engine is $ENGINE_HOME/config/engine-config.xml

This file describes to StreamHorizon engine where to find input data, how to process it and where to send output results.

LOGGING

Logging is configured in $ENGINE_HOME/config/logback.xml

Please refer to logback (http://logback.qos.ch/manual/configuration.html) documentation for further details.

CACHING

StreamHorizon uses clustered cache to avoid unnecessary trips to database. It is possible to choose between two clustered cache technologies: Infinispan and Hazelcast. By default Infinispan is used and this is recommended.

Configuration files for each of these cache technologies can be found in $ENGINE_HOME/config/

Please refer to appropriate online documentation to understand how to tune these libraries properly.

MAPPING DIMENSIONS

PROCESSING INPUT DATA (FEEDS)

StreamHorizon engine is configured to watch particular directory on disk (<sourceDirectory>) and process files found there. Different feeds can decide to watch for different files in the same directory and this is achieved by setting file name regular expression.

It is safe for multiple threads to watch same source directory and all threads will evenly partition work and this works out of the box. It is also possible to configure multiple StreamHorizon engine instances to watch same source directory, but this requires explicit configuration.

All ETL threads watch source directory, try to match them against their file name pattern and files that pass criteria are processed in a round-robin fashion by one of free threads.

Files are processed as configured in engine-config.xml file.

CREATING OUTPUT FILES

The result of processing input data files can be written directly to one of directories on disk (<bulkOutputDirectory>). Files written to disk are text files. When writing files to disk it is important to define extension for final output file and delimiter to be used when outputting data values.

All input data files are processed by ETL thread pool.

Here is code sample how to output results of processing directly to disk

```
…
<bulkLoadDefinition outputType="file">
        <bulkLoadOutputExtension>ext1</bulkLoadOutputExtension>
        <bulkLoadFileDelimiter>,</bulkLoadFileDelimiter>
…
```

All input files (if successfully processed) will be output to <bulkOutputDirectory> with extension ext1 and resolved values will be delimited using comma separator. Every line in input file will have corresponding line in output file.

Output files are created as temporary (with different extension) and atomically renamed only after all data has successfully been written. This prevents threads in DB pool to read half-written files.

LOADING OUTPUT DATA FILES INTO DATABASE

It is possible to instruct StreamHorizon to load data from output file into database. In this case DB threads will watch folder <bulkOutputDirectory> for files with given extension (<bulkLoadOutputExtension>) and will execute command specified in <bulkLoadInsert>. It is possible to specify SQL or shell command here.

For example, this can be used to load data to Oracle (external tables), to execute shell script to invoke MSSQL to load data or let MySQL execute LOAD DATA INFILE statement.

INSERTING OUTPUT DATA DIRECTLY INTO DATABASE

It is possible to configure StreamHorizon to insert output data into database using JDBC interface. This avoids generation of interim data files on disk and can help reducing IO. Processing logic and steps are exactly the same as if data is written to disk, the only difference is that loading happens in batches as data is being processed. Batch size can be configured.

ERROR HANDLING

TODO: Describe error handling onCompletion/onFailure/onSuccess

By default all warnings and errors are written to corresponding log files in $ENGINE_HOME/log/ directory. This happens in case when some of user-defined SQL statements are invalid, data format is not correctly described in engine-config.xml or similar cases.

CONFIGURATION PARAMETERS

Almost all parts of StreamHorizon can be configured by supplying configuration parameters. It is possible to turn off certain features (like remote commands, local caches) or tune them to behave differently (for example use Hazelcast instead of Infinispan for caching, use different read buffer size).

Configuration parameters can be specified in engine configuration file and can also be overridden by providing Java system parameters with the same name. This is useful when multiple instances of StreamHorizon want to perform processing in the same way but only some parts need to be different. In this case engine configuration is template for processing, containing default tuning, and every instance overrides some of parameters by specifying them in startup scripts.

CONTEXT ATTRIBUTES

Context attributes are very simple extension point of StreamHorizon engine. They can originate from StreamHorizon engine itself (implicity context attributes) and from Java plugins provided as extensions to the engine itself.

Context attributes can be used in engine configuration files and in Java plugins.

Some attributes are available only during execution of ETL threads, some only during execution of DB threads and some always.

Context attributes can be used in engine configuration file by reference – enclosed in ${ and }. Engine will recognize these and replace them with appropriate values during execution. For example ${ engineInstanceStartTimestamp} will always be replaced with time when that particular StreamHorizon engine was started (Unix timestamp). Java plugins get all the context attributes passed as one of parameters and they can change existing attributes or add new ones.

Implicitly provided context attributes are:

Attribute name	Description	Available in
engineInstanceStartTimestamp		ETL,DB
engineInstanceIdentifier		ETL,DB
bulkProcessingThreadID		

feedProcessingThreadID		ETL
feedInputFilePath		ETL
feedInputFileName		ETL
feedInputFileReceivedTimestamp		ETL
feedInputfileReceivedDateTime		ETL
feedInputFileProcessingStartedTimestamp		ETL
feedInputFileProcessingFinishedTimestamp		ETL
feedInputfileProcessingStartedDateTime		ETL
feedInputFileSize		ETL
feedBulkLoadOutputFilePath		ETL
bulkFilePath		ETL,DB
bulkFileAlreadySubmittedForLoading		DB
bulkFileName		ETL,DB
bulkFileReceivedForProcessingTimestamp		DB
bulkFileProcessingStartedTimestamp		DB
feedInputFileJdbcInsertStartedTimestamp		ETL (with JDBC output)
feedInputFileJdbcInsertFinishedTimestamp		ETL (with JDBC output)
bulkFileProcessingFinishedTimestamp		DB
bulkCompletionProcessingSuccessFailureFlag		DB
bulkCompletionProcessingErrorDescription		DB
feedCompletionProcessingSuccessFailureFlag		DB
feedCompletionNumberOfTotalRowsInFeed		DB
feedCompletionProcessingErrorDescription		DB

CREATING PLUGINS

StreamHorizon platform provides extension points where users can add their own processing logic written in SQL, Java or shell scripts. This enables to interact with StreamHorizon engine and customize processing steps and actions to be taken.

SQL AND SHELL COMMANDS

JAVA PLUGINS

CLUSTERING STREAMHORIZON

It is possible to run multiple StreamHorizon engine instances (multiple JVMs) and let them partition processing and data loading.

TODO: describe further

PERFORMANCE TUNING

TUNING OUTPUT TYPE

TUNING DATABASE

TUNING THREAD POOLS

StreamHorizon uses two different thread pools. First thread pool (ETL) is used for processing input feed files and the second one (DB) is used to load bulk files. In case when data is inserted into database using JDBC then ETL thread pool will execute that logic.

It is recommended to set number of threads to a number very close to number of cores your machine has. It is always recommended to thoroughly test your configuration.

STORAGE, READ AND WRITE BUFFERS

DIMENSION CACHES

There are two levels of caching used by StreamHorizon engine.

1. Per-thread dimension cache
2. Local dimension cache
3. Distributed dimension cache (Infinispan or Hazelcast)

Tuning these caches properly can reduce memory consumption and significantly speed-up execution.

TODO: explain tuning each cache here.

REMOTE COMMANDS

StreamHorizon engine accepts remote commands via HTTP. This feature can be turned off. Currently we support following commands:

FLUSHING DIMENSION CACHE REMOTE COMMAND

In case when dimension cache is stale (for example data was loaded into table bypassing StreamHorizon engine) it is important to inform engine to stop using old data. This can be done by executing

http://localhost:<remoting.server.port>/flushDimensionCache/?dimension=DIMENSION_NAME_AS_DEFINED_IN_CONFIG

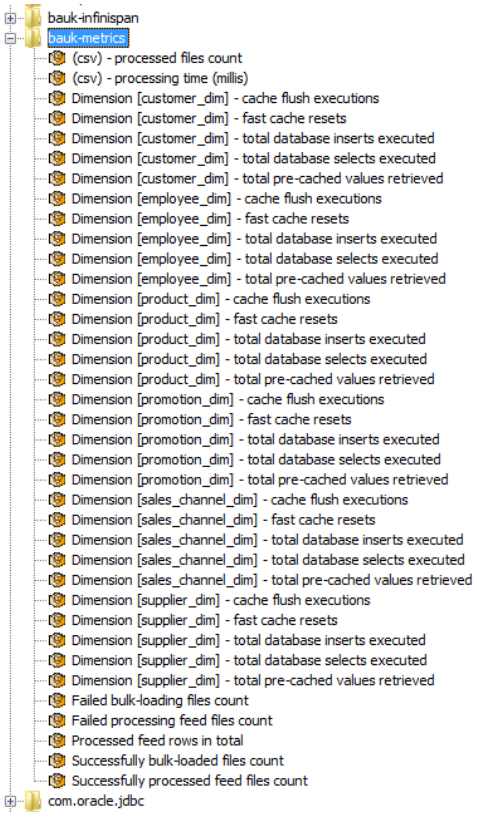
When this is executed engine will go through a set of steps:

1. Wait for all threads to finish their current work
2. Pause all threads so that they do not accept new work
3. Flush data cached for specified dimension
4. Notify all threads that they can continue processing data

Please note that after flushing dimension cache there is expected slow-down in execution because cache will be populated as needed and this might require execution of database queries.

MONITORING

StreamHorizon exposes internal processing metrics via JMX.



It is also possible to use extension points to collect and calculate metrics about performance of engine. See the sales demo provided with installation for more details (sh_metrics table).

MISCELLANEOUS

COMMON MISTAKES

PLANNED PLATFORM EXTENSIONS

There are many planned extension to StreamHorizon. Here are only few of them:

- Accepting streams and being able to process them (no need to read data from disk)
- Being able to write output file in format different than plain text (we plan to make this pluggable so that customers can plug-in their own format)
- More extension points