CONDUCTOR: MANAGING PROCESSING PIPELINES. Bradford Castalia, University of Arizona, Department of Planetary Sciences, 1541 E. University Blvd., Tucson, Arizona 85721-0063, Castalia@Arizona.edu.

Conductor is a Java application for managing queues of source files to be processed by sequences of procedures.

Processing Pipelines: Data production operations commonly involve a sequence of procedures that are routinely used to generate output data products from an input data source; this is a processing pipeline. More than one procedure sequence may be employed, typically depending on the kind of input data and/or the desired type of output, but any given sequence is usually sufficiently well defined that it can be automated into a non-interactive uber-procedure that encapsulates the individual procedures of the sequence. This often takes the form of a script that uses as input the file containing the data source and executes the sequence of procedures on the file, and/or intermediate data files, to generate one or more products containing the desired output data.

Managing Pipelines: When implementing a processing pipeline as a script it is easy to overlook the mundane tasks of output logging, checking the completion status of each procedure and gracefully handling failure conditions. Conductor does this automatically. Managing all the parameters that control a processing pipeline can be a challenge. Conductor uses a configuration file, supplemented by dynamically managed parameters, to provide parameter management. Parameters and database field values can be referenced by procedure definitions much like scripting language syntax. However, Conductor is not intended as a substitute for scripts. Conductor manages procedures. Any of the procedures in a Conductor pipeline may be scripts as well as binary executables. Conductor is intended to keep the processing pipeline manageable even when the procedures and processing environment become complex.

Database Driven: Conductor is designed to manage processing pipelines without requiring the pipeline implementer to write a script. Instead, the definition of the procedures is provided in a *Pipeline*_Procedures database table and the names of the source files to be processed are entered into a *Pipeline*_Sources database table. These two tables constitute the named *Pipeline*. Currently support is provided for access to MySQL database servers. Support for additional types of database servers may be added as needed. Access information to the database server is provided in the configuration file given to Conductor. If Conductor's connection to the database server is lost it will make repeated attempts to reconnect before giving up and reporting loss of database connectivity.

Pipeline_Sources. Each source record identifies the data source to be processed, its processing status, and the location of the log file containing a detailed report of all processing of the data source. Additional fields, beyond those required by Conductor, may be present. Source records are processed in the order they occur in the table, and new records may be added to the table at any time.

The only field value that must be user specified to identify a data source is the Source_Pathname that provides a pathname to a File. Conductor will confirm read access to each source file; files without read access will not be processed. The Source_Pathname need not be unique; sources may be repeatedly processed. A Source_Number field that contains a unique integer value, usually maintained by the database server as an auto-increment field, is required. In addition a Source_ID may be provided by the user; otherwise Conductor will set this to the filename portion of the Source_Pathname, with any extension removed.

Conductor will only process a source record that indicates unprocessed status. Conductor acquires a source record by setting the Conductor_ID field to the processing hostname in a way that guarantees only one Conductor will acquire each record, thus allowing any number of Conductor processes to be simultaneously working on a pipeline. Conductor maintains the Status field of the source record with a list of Status Indicators, one for each procedure that has been applied to the source, in procedure sequence order. The value indicates a procedure in progress or the final status of the procedure – success, failure or timeout. A source record with a Status field value when it is acquired will be reprocessed beginning with the next unrecorded procedure sequence, but only if the the last procedure was successful.

Conductor will always write a detailed report of all source record processing to the file listed in the Log_Pathname field. The user may specify the value of this field or let Conductor determine the filename based on the pipeline name, Source_ID and Source_Number values. The log file will be created in Conductor's current working directory unless a Log_Directory configuration parameter has been provided. If Conductor finds that the Log_Pathname field has a value then its report will be appended to

an existing file, which assures that source record reprocessing will be reported to the same log file. The log file contains details about the pipeline Conductor is processing, the host system in use, the source data identification, timestamps for the beginning and end of each procedure, and a copy of all normal and error listings. The report is marked in a way to facilitate automated data extraction.

Pipeline_Procedures. Each procedure record specifies the order in which procedures are to be applied to a data source, a primary command line, completion success conditions, and a branch command to be used on failure of the primary command. Additional fields, beyond those required by Conductor, may be present; in particular, a Description field will be included in the log reporting. Procedure definitions may be modified while Conductor is running.

The Sequence field determines the order in which procedures are run. The value of this field is a real number to enable new procedure records to be inserted in the table in any effective sequence location without requiring procedure record reordering or renumbering.

The Command_Line specifies the procedure to be executed. The command line specification may contain embedded references to be resolved by configuration parameter or database field values. The Conductor configuration file parameters are defined with simple Parameter Value Language syntax. These are supplemented with all the environment variables available to Conductor plus a set of parameters identifying the pipeline and its database. Conductor also maintains a set of dynamic parameters that identify the current source record, procedure record sequence number and the completion status value of the last procedure run. Both parameter and field references may be nested.

A procedure will only be allowed to run the amount of time specified by the Time_Limit value to avoid "hung" or "runaway" processes. The specification is reference resolved and also evaluated as a potential mathematical expression.

Once a procedure has completed its exit status is compared to the Success_Status value to determine if the procedure completed successfully. In addition to being reference resolved and evaluated as a potential mathematical expression, the Success_Status is also evaluated as a potential logical expression to determine the success condition of the procedure exit status. As an alternative to the Succcess_Status, a Success_Message (reference resolved) may be specified; it is used as a regular expression match condition on all normal and error output from the procedure. Conductor may be configured to use the conventional zero Success_Status or to always assume the procedure completed successfully.

If Conductor determines that the procedure completed successfully, then the next procedure is applied or, if the last procedure in the sequence completed, the next source record is acquired and the sequence begins again. However, if the procedure failed, or timed out, the On_Failure command line (reference resolved, of course) is run instead. This procedure is allowed to run to completion (without any time limit) before the next source record is acquired and the sequence begins again.

Distributed Processing: Conductor is designed for use in a distributed processing systems environment. Each pipeline may be processed by multiple Conductor processes simultaneously; each source record is guaranteed to be processed by only one Conductor regardless of the number of Conductors working on the pipeline. Typically each Conductor runs on a different processing engine to maximize throughput. Multiple Conductors may be run on the same host system with each processing a separate pipeline "segment". A segment is a pipeline in which the final procedure, or any branch (On_Failure) procedure, makes an entry in another pipeline considered to be "chained" to the first segment. Networks of pipeline segments may be constructed in this way.

Operating Modes: Conductor may be run in either monitor, batch or daemon mode. In monitor mode a graphical user interface is provided to control and monitor pipeline processing; the entire log report for each source record is displayed as it is being produced, along with other information about the procedures. In batch mode Conductor processes all unprocessed source records in the pipeline and then quits. In daemon mode Conductor runs in the background and continuously polls, at a configurable interval, for unprocessed source records. In this mode Conductor can be run unattended and it will automatically process new source records as they appear in the pipeline queue.

Lightweight and Open Source: Conductor is a single pure Java application. It is distributed from http://pirl.lpl.arizona.edu/software/Conductor as part of the PIRL Java Packages available as open source software that includes comprehensive javadoc for all APIs to facilitate reuse. Additional applications are provided to complement Conductor for adding source record entries to a pipeline and sending email notifications from a pipeline.