GNC DATA EXPORT UTILITY

Developed by General Networks Corporation MarkLogic Consultants
Developer: Robert Kennedy (rkennedy@gennet.com)
Contributors: Dennis Garlick (dgarlick@gennet.com) & Javier Lizarraga (jlizarraga@gennet.com)

GNC Data Export Utility

Overview:

The GNC Data Export Utility is a Java-based command-line application that allows easy importing and exporting of data when using MarkLogic NoSql Databases. The utility supports two types of data sources, relational databases (currently Oracle and MS SQL) and MarkLogic databases. Depending on the data source, the program leverages different configuration files that contain connection information, queries, filters, selected elements and file locations. Data is exported to a text-delimited file with a user-defined delimiter. Data pulled out of relational databases may then be imported into MarkLogic using the MarkLogic Content Pump (MLCP), while data pulled out of MarkLogic may be imported into other applications that are compatible with a text-delimited format such as Excel, R and Tableau.

This document will first discuss how to configure the export utility when the data source is a relational database, followed by some examples. Next it will discuss how to configure the export utility when the data source is MarkLogic, again followed with some examples.

Extracting Data from Relational Databases

There are at least two configuration files that need to be edited before running the program, namely settings.txt and query.txt.

The settings.txt file contains the various database connectivity settings, authentication details and date-format configuration. It is important to note that the utility supports password encryption, so that the password does not need to be stored in clear text. The file also contains performance settings that are described below:

1. Fetch Size: Number of rows fetched from the database and stored in memory at a time. This can be fine-tuned to improve performance.
2. Output Size: Number of exported rows to be stored in memory before writing to disk. Keeping this value low leads to greater performance.
3. Chunk Size: Number of rows per file, before creating a new chunk file. This is to avoid situations where one file is unmanageable in size, and allows you to control the number of files generated by the program.

The query.txt file contains the result-set target destination including filename, the SQL query to be executed and an optional count query used to estimate completion time. It's possible to have multiple rows configured, with each row in the text file corresponding to a separate query. Within each row, three pound symbols (###) are used to separate parameters.

1. Filename. This is the filename the exported data will be written to for this query. The file is placed by default in the output folder specified in the settings file. For exports where multiple chunks are created, each chunkfile number is appended to the filename, e.g. filename.1.dsv, filename.2.dsv.
2. Query. This is the sql query that will be executed against the data source. This field also optionally supports a file location instead of a query. When used, the file location refers to a text file that contains an SQL query. This is needed when the SQL query if formatted to extend over multiple rows.
3. Count Query (optional). In the event that you are executing very large queries, you can include the third optional query to estimate a completion time. This is used by the application solely to estimate how many rows there are and to estimate a completion time. If this field is not populated, then the program does not estimate a completion. Note that a count(*) on a complex query can sometimes take almost as long as the complex query itself. However, if it is known that the number of records to be returned is equal to the number of records in a much simpler query such as a single table, then this count(*) can be used to get a quick estimate of overall progress for the complex query.

Running the program:

Once the two configuration files have been edited, the program can be run. The program code is all contained within a single JAR file and can be run using a batch script or a direct command in the command console. Note that you can also use Windows Task Scheduler to automate the process. The program takes a single argument, the path to the settings.txt file you are using. See examples of the full command further in the documentation.

If the program is run from a Command Prompt, it will provide status information including the query being run and the estimated completion time of the query – assuming a count query was also included. In addition, pressing X will stop the program immediately and cause a graceful exit.

Logging:

Either a logging directory or path to a log file must be specified in the settings.txt file.
If a directory is specified, then each run of the program creates a new logfile that is stored in the directory specified. The logfile contains the timestamp when the export was run in its filename.
If a full path to a file is specified, then each run will append to this specific log file.
The log contains a reasonable amount of logging of the running of the export, including any errors and also including the times that each query took to export.

Encryption:

Password Encryption is done using Java Simplified Encryption (see http://www.jasypt.org/encrypting-configuration.html). Below are directions on how to encrypt the database-user password. You will use the following procedure to create an encrypted password that would be stored in the settings.txt configuration file.
You can download the Java Simplified Encryption package below:
http://www.jasypt.org/download.html
You will need to download the following zip file jasypt-1.9.2-dist.zip. Unzip the file, then go to the following bin directory:

C:\... \jasypt-1.9.2-dist\jasypt-1.9.2\bin

Run the following command.

```
C:\...\jasypt-1.9.2-dist\jasypt-1.9.2\bin>encrypt.bat input=mypassword password=secret
----ENVIRONMENT-----------------
Runtime: Oracle Corporation Java HotSpot(TM) 64-Bit Server VM 25.40-b25
----ARGUMENTS-------------------
input: mypassword
password: secret
----OUTPUT----------------------
FPK7vImLr7XKKLEmhTfgXnOWxHeiEzSs
```

The setting.txt file contains two configuration steps. Step one is selecting the Database Type and the flat file Delimiter. Step two is configuring the Database Type selected in Step one. Both Oracle and MS SQL contain an additional <DATABASE TYPE>_No_HEADER and <DATABASE_TYPE>_DATE_FORMAT setting. If <DATABASE_TYPE>_NO_HEADER is set to TRUE then the column name header will not be included. If <DATABASE_TYPE>_NO_HEADER is set to FALSE then the column name header will be included in the first row of data. You can configure the desired date format by setting the <DATABASE_TYPE>_DATE_FORMAT, for instance, <DATABASE_TYPE>_DATE_FORMAT=MM-DD-YYYY.

EXAMPLES:

Below are examples of how to use the configuration files. You'll notice that the settings.txt file has a reference to the query.txt configuration file. As explained above, the query.txt configuration file can contain the embedded SQL query to be executed, or it can be configured to reference a separate SQL query configuration file.
Please note that all of the examples below reference an encrypted password. In order for the Java Utility to decrypt an encrypted password, be sure to encapsulate the password with ENC(encrypt password).

See the Examples directory. The execute command is as follows:

- java -jar <full path>/GNC_DataExtract.jar <full path>/settings.txt

For example: java -jar "C:\Examples\GNC_DataExtract.jar" "C:\Examples\settings.txt"
The examples include configuration for Oracle (SID/Service Name) and MSSQL (Server Authentication/Windows Authentication).

Oracle (Only populate 1 leaving the other blank, either SID or Service Name accordingly). See settings_oracle.txt.

- ORACLE_HOST: localhost
- ORACLE_PORT: 1521
- ORACLE_SID: XE or ORACLE_SERVICE_NAME: tap75p.oradb.xyz.com
- ORACLE_USERNAME: marklogic_ro
- ORACLE_PASSWORDE: ENC(0s9ZqzKoK9oicsJS5iOpOh1kUpuZFcwe)

MSSQL (Server Authentication – Set MSSQL_WINDOWS_AUTH=FALSE) See settings_mssql_sa.txt.

- MSSQL_HOST: localhost
- MSSQL_PORT: 1433
- MSSQL_USERNAME: marklogic_ro
- MSSQL_PASSWORD ENC(0s9ZqzKoK9oicsJS5iOpOh1kUpuZFcwe)

MSSQL (Windows Authentication – Set MSSQL_WINDOWS_AUTH=TRUE) See settings_mssql_wa.txt.

- MSSQL_HOST: localhost
- MSSQL_PORT: 1433
- MSSQL_WINDOWS_AUTH: marklogic_ro
- MSSQL_DATABASE_NAME Test_Table

CONFIGURATION EXAMPLES FOR RELATIONAL DATABASE SOURCE SYSTEM:

CONNECTING TO ORACLE – USING SID OR SERVICE NAME & EMBEDDED SQL QUERY

1. Use following setting file: .../Examples/Embedded/settings_oracle.txt
2. Use following query file: .../Examples/Embedded/query.txt
   Execute Command:
   java -jar C:\Examples\GNC_DataExport.jar C:\Examples\Embedded\settings_oracle.txt

CONNECTING TO MS SQL – USING SERVER AUTHENTICATION & EMBEDDED SQL QUERY

1. Use following setting file: .../Examples/Embedded/settings_mssql_sa.txt
2. Use following query file: .../Examples/Embedded/query.txt
   Execute Command:
   java -jar C:\Examples\GNC_DataExport.jar C:\Examples\Embeded\settings_mssql_sa.txt

CONNECTING TO MS SQL – USING WINDOWS AUTHENTICATION & EMBEDDED SQL QUERY

1. Use following setting file: .../Examples/Embedded/settings_mssql_wa.txt
2. Use following query file: .../Examples/Embedded/query.txt
   Execute Command:
   java -jar C:\Examples\GNC_DataExport.jar C:\Examples\Embedded\settings_mssql_wa.txt

CONNECTING TO ORACLE – USING SID OR SERVICE NAME & REFERENCED SQL QUERY

1. Use following setting file: .../Examples/Referenced/settings_oracle.txt
2. Use following query file: .../Examples/Referenced/query.txt
3. Use following query file: .../Examples/Referenced/my_sql_query.txt
   Execute Command:
   java -jar C:\Examples\GNC_DataExport.jar C:\Examples\Referenced\settings_oracle.txt

CONNECTING TO MS SQL – USING SERVER AUTHENTICATION & REFERENCED SQL QUERY

1. Use following setting file: .../Examples/Referenced/settings_mssql_sa.txt
2. Use following query file: .../Examples/Referenced/query.txt
3. Use following query file: .../Examples/Referenced/my_sql_query.txt
   Execute Command:
   java -jar C:\Examples\GNC_DataExport.jar C:\Examples\Referenced\settings_mssql_sa.txt

CONNECTING TO MS SQL – USING WINDOWS AUTHENTICATION & REFERENCED SQL QUERY

1. Use following setting file: .../Examples/Referenced/settings_mssql_wa.txt
2. Use following query file: .../Examples/Referenced/query.txt
3. Use following query file: .../Examples/Referenced/my_sql_query.txt
   Execute Command:
   java -jar C:\Examples\GNC_DataExport.jar C:\Examples\Referenced\settings_mssql_wa.txt

Extracting Data from MarkLogic Databases

Pulling data out of MarkLogic into text-delimited flat files is useful as this is a common format that allows data contained within Marklogic to be loaded into other applications that do not have native MarkLogic connectors. Even when such connectors do exist, sometimes exporting to a text file first can be both faster and more reliable.

There are three configuration files that need to be edited before running the program.

1. settings.txt – This is the same settings.txt file used for extracting from relational sources, but now needs to be configured with settings for the MarkLogic database source. The configuration file consists of two main sections defining the extraction source (data type) and defining the data type source connection details. In this case, the database source would be MarkLogic. Again, the utility supports password encryption, so that the password does not need to be stored in clear text. Please refer to the section above to know how to encrypt a password.
2. elements.txt – This contains a list of the elements to extract from a MarkLogic database. Only one element should be specified per row. Each element name is used as a column header in the output file. You can choose to map the element name in the Marklogic document to a different column header if desired using ==. For example, if you wanted to pull an element called "Name" from Marklogic but have this element placed in a column called "FullName", you would use the syntax "Name==NewName". Finally, you can specify comments in this file by starting a line with #.
3. filters.txt (optional) - Constrains the documents returned from a MarkLogic database via xml syntax (see https://docs.marklogic.com/guide/search-dev/). This file needs to be present, but can be empty if no filters are to be applied to the data.

The Logging section and Encryption section discussed above in Relational Databases also applies to this section.

Here are some simple examples of how the filter.txt file can be leveraged to constraint the MarkLogic documents.

1. Filter based on the occurrence of an element and value in the document:
```
<category>CARS</category>
```
2. Filter based on the occurrence of two elements and values in the document:
```
<category>CARS</category>
<style>SEDAN</style>
```
3. Filter based on the occurrence of one or another element in the document:
```
<q:or>
<Date>2015-12-26</Date>
<Date>2015-12-27</Date>
</q:or>
```
4. Filter based on an element value falling within a range (need to use filtered if the element is not indexed):
```
<Date><q:gt>2014-01-01T00:00:00</q:gt></Date>
<Date><q:lt>2014-01-31T00:00:00</q:lt></Date>
<q:filtered>true</q:filtered>
```

The MarkLogic section also contains file path location configuration for the following:

1. MARKLOGIC_OUTPUT_PATH: Full path to where the output file will be written.
2. MARKLOGIC_LOGS_DIR: Directory or full path to where the log file(s) will be written.
3. MARKLOGIC_ELEMENTS_FILE_PATH: Full path to the elements.txt file.
4. MARKLOGIC_FILTERS_FILE_PATH: Full path to the filters.txt file.

There are also the following additional settings within the MarkLogic section:

1. MARKLOGIC_PAGE_SIZE: Number of exported rows to be stored in memory before writing to disk. Fine tuning this value should lead to greater performance.
2. MARKLOGIC_ROOT_NODE: Used to help flatten a nested XML document. This functionality currently supports simple nesting but can be enhanced to support more complicated XML structures. For example given the following XML document structure.

```
<Checklist>
    <Name>My Checklist</Name>
    <Date>02/02/2016</Name>
    <Question>
        <Text>First Question</Text>
        <Answer>First Answer</Text>
    </Question>
    <Question>
        <Text>Second Question</Text>
        <Answer>Second Answer</Answer>
    </Question>
</Checklist>
```

Assume the elements.txt contains all the elements, Name, Date, Question, Text and Answer. If you set MARKLOGIC_ROOT_NODE=Question then the generated output file format would be:

```
Name|Date|Text|Answer
My Checklist|02/02/2016|First Question|First Answer
My Checklist|02/02/2016|Second Question|Second Answer
```

Below are some examples based on the following MarkLogic information:

MarkLogic - See settings_marklogic.txt.

- MARKLOGIC_HOST: localhost
- MARKLOGIC_PORT: 8001
- MARKLOGIC_USERNAME: marklogic_ro
- MARKLOGIC_DB_NAME: Appointments

CONFIGURATION EXAMPLES FOR MARKLOGIC DATABASE SOURCE SYSTEM:

CONNECTING TO MARKLOGIC – NO FILTERING

1. Use following setting file: .../Examples/MarkLogic/settings.txt
2. Use following elements file: .../Examples/MarkLogic/elements.txt
3. Use following filters file: .../Examples/MarkLogic/filters_empty.txt
   Execute Command:
   java -jar C:\Examples\GNC_DataExport.jar C:\Examples\MarkLogic\settings.txt

CONNECTING TO ORACLE – USING FILTERING OPTION

1. Use following setting file: .../Examples/MarkLogic/settings.txt
2. Use following elements file: .../Examples/MarkLogic/elements.txt
3. Use following filters file: .../Examples/MarkLogic/filters.txt
   Execute Command:
   java -jar C:\Examples\GNC_DataExport.jar C:\Examples\MarkLogic\settings_filters.txt