JobStarter

Contents

[Description 3](#_Toc534708264)

[Modules 3](#_Toc534708265)

[JobStarter 3](#_Toc534708266)

[Description 3](#_Toc534708267)

[Settings 3](#_Toc534708268)

[Segments 4](#_Toc534708269)

[<app-config> 4](#_Toc534708270)

[<bulk-size> 4](#_Toc534708271)

[<connection-timeout> 4](#_Toc534708272)

[<directories> 4](#_Toc534708273)

[<mail> 5](#_Toc534708274)

[<db-types> 5](#_Toc534708275)

[<connection-string> 6](#_Toc534708276)

[<ldap-server> 6](#_Toc534708277)

[Commands 7](#_Toc534708278)

[Parameters 9](#_Toc534708279)

[CallExtractSql 13](#_Toc534708280)

[CallExtractTable 13](#_Toc534708281)

[CallSqlLoader 13](#_Toc534708282)

[CallStoredProcedure 13](#_Toc534708283)

# Description

JobStarter is a command window based ETL tool created for extracting information from different sources based on:

1. Queries (SQL) contained in a file.
2. Metadata. The client used for the connection must have read from data dictionary privileges.

This tool support multiples databases connection, such as: Oracle, DB2, Teradata, SQL Server, MS Access. In the future, more databases support can be added, using the interface “IDBConnection”.

One benefit offered by this tool is the ability of encrypt the password used by the Database connection, adding another level of security. Connections and password modifications can be done by the respective interface passing the necessaries parameters, we will cover this in further session.

This applications use xml configuration files adding a high level of customization. If you need to pass a new parameter to a module, just defined it in the “Commands.xml” file (covered later on this document).

# Modules

## JobStarter

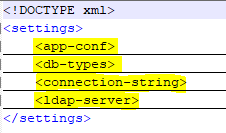
### Description

This is the Main Module containing the executable logic. Parameters are passed to this module and desired logic will be executed.

### Settings

The setting file for this module can be found under “**Settings**” folder (Application Path -> settings -> settings.xml). The settings file is in XML format and contain 4 sections or segments:

Figure



#### Segments

##### <app-config>

Figure



###### <bulk-size>

Default value 50,000 (records)

This parameter defined the number of records pulled from the database on the RecordSet at a time. The value is a combination of network speed, database type, volume of information extracting. We found that 50K is the best number, and we get the best runtime/performance.

###### <connection-timeout>

Default value 900,000 (milliseconds)

This parameter is useful when you are running a query for long period of time. Any extracting operation should finalize before this timeout, otherwise you will get a connection timeout and triggering an error reporting process.

###### <directories>

Here is defined the directories used by the application. [app\_directory] is a reserved word and has to be in-between “[]”, this indicated to the app to include in the path the working directory where the JobStarter.jar app is running:

1. Outpu-dir

Where plain files generated from the source extraction will be placed.

1. Log-dir

Each run generate a LOG file, here you can define the path for these files

1. Archive-dir

After loading a file into the database, you have the option to specify if archiving those files is desired. Here you can define the path for the archiving.

1. Input-dir

The input folder is used by the LOAD process for loading plain files into the database.

1. Work-dir

Once the LOAD process identify a file or files to be loaded, these are copied into the WORK directory (path provided here) + and folder named after the Process ID.

1. Lib-dir

Where all modules created for this application are copied. All JARs file not including database connection drivers, these are copied into the DRIVERS folder.

1. Script-dir

All scripts created for extraction need to be placed in this folder. The application will look for any xxx.sql file matching the parameter used.

1. Ctl-dir

90% of the time CTLs are built on runtime by CallSqlLoader.jar module, using destination metadata. But if needed it, custom CTL file can be created and has to be place in this folder.

###### <mail>

1. Mail-param

All parameter related to the SMTP server

1. Mail-list

List of all receipting mails

##### <db-types>

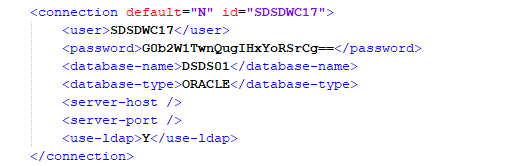
Here is defined all supported databases:

1. Class: name of the class used
2. Driver: the driver published by the fabricant
3. Id: used later on the connection-string configuration

##### <connection-string>

Here is configured necessary parameters to connect to a desired database. Any modification or new connection has to be done through the interface calling the respective parameter.

Figure



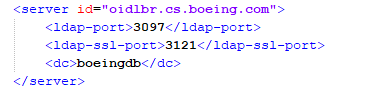
1. ID: the unique identifier for the connection. Used by jobs
2. Default (Y/N): this property indicate if the connection is default and used if no connection id is specified in a job call.
3. User: database user used by the connection
4. Password: encrypted password used by the connection
5. Database-name: The name of the database
6. Database-type: Id specified and created in previous section (<db-types>)
7. Use-ldap (Y/N): this parameter is valid for Oracle database connection. In case of Boeing, an LDAP server is used for user authentication.
8. Server-host: if no LDAP server authentication is used, the you might have to indicate the server address
9. Server-port: same as prior point, is no LDAP server is used, server-host and port must be specified.

##### <ldap-server>

Boeing use LDAP server for database user authentication. This information was obtained from TOAD connection tool. For any other information related contact:

Moeller (US), Kenneth M [Kenneth.M.Moeller@boeing.com](mailto:Kenneth.M.Moeller@boeing.com)

Figure



1. Id: LDAP host server name
2. Ldap-port: none ssl port
3. Ldap-ssl-port: SSL port
4. Dc: the domain controlled

### Commands

| **ID** | **Parent** | **Description** | **Name\*** | **Type\*** | **Class\*** | **Value\*** | **Jar\*** | **Required\*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| -r |  | Parameter indicating the RUN action | OBJECT | Call | CallerImpl | next | N/A | N/A |
| r | -r | Run stream job configured in the database. | OBJECT | LIB | Not Implemented |  |  |  |
| s | -r | Run Stored Procedure using a Jar module in LIB directory  E.g.  -rs <procedureName1,procedureName2,…> --c connectionName JOBNAME=’Name given to the job’ | OBJECT | LIB | com.boeing.jobstarter.services.CallStoredProcedure | next | CallStoredProcedure.jar | N/A |
| f | -r | Run anonymous block contained in a file. | OBJECT | LIB | Not Implemented |  |  |  |
| l | -r | Run load module (SQL Loader) for loading files into oracle database | OBJECT | LIB | com.boeing.jobstarter.services.CallSqlLoader | none | CallSqlLoader.jar | JOBNAME,  CONNECTION |
| x | -r | Extract |  |  |  |  |  |  |
| t | x | Extract and generate plain file, from sources based on table name and metadata on the source.  e.g.  -rxt <tablename1,tableName2,…> --c connectionName --sd schemaDestName –JOBNAME=’Name given to the job’ | OBJECT | LIB | com.boeing.jobstarter.services.CallExtractTable | next | CallExtractTable.jar | JOBNAME,  CONNECTION, SCHEMADEST |
| w | x | Extract from Web Service | OBJECT | LIB | Not Implemented |  |  |  |
| q | x | Extract and generate plain file, based on a file containing a SQL command  e.g.  -rxt SQLFIleName.sql --c connectionName --sd schemaDestName --JOBNAME=’Name given to the job’ | OBJECT | LIB | com.boeing.jobstarter.services.CallExtractSql | next | CallExtractSql.jar | JOBNAME,  CONNECTION, SCHEMADEST |
| c | -r | Connection Configuration module |  |  |  |  |  |  |
| l | c | List all connections | LIST | call | CallConnectionConfig | none | N/A | N/A |
| a | c | Add new connection | ADD | Call | CallConnectionConfig | none | N/A | N/A |
| d | c | Delete a connection | DELETE | Call | CallConnectionConfig | none | N/A | N/A |
| m | c | Modify a connection | MODIFY | Call | CallConnectionConfig | none | N/A | N/A |
| t | c | Test a connection | TEST | Call | CallConnectionConfig | none | N/A | N/A |
| d | -r | Create a table on the destination schema base on a table name passed as parameter on the source database  e.g.  -rd tableName1, tableName2… --c connectionName --sd schemaDestName | OBJECT | Call | CallDDL | next | N/A | N/A |

**ID**: Parameter code. Can be one letter or multiple base on desire logic to execute.

E.g. Let say we want to execute the SQLLoader module. We need to send the respective parameters

-r => Run

l => Load

The complete parameter command should looks like

-rl --p <file patter1>=<destination table1>,<file patter2>=<destination table2>,… --c <connection id> --REFRESH=<TRUNCATE|DELETE> (if DELETE was chosen, then DELETEBY parameter must be specify) DELETEBY=”column1=value1 and column2=value2…“

**Name**: Command name

**Type**: specify if the desired action is a call to a core logic on JobStarter module or a call to a custom jar (module) added as enhancement

Call: will call a class or CORE logic in the JobStarter application

E.g. the Connection configuration is part of the Core and the way to invoke this functionality is declaring the action as Call because it, implements the ICall interface.

LIB: Will invoke an external JAR included in the LIB folder. This functionality will generate an instance of the invoked class and execute it. Any LIB class to be call, has to extend from Callable Abstract class.

**Class**: the class name to be invoke. In the case of LIB should be included the entire classPath and for Call type only the class name

**Value**: If there is a value to be read, indicate in which position the value is coming. To any given value, containing spaces, it should be enclosed by “…”

None: no value to read

Next: read from next position parameters.

Read: read the parameter from the same position after the expected “=”

E.g. --PARAM=value or --PARAM=”value with spaces”

**Jar**: for Command type LIB, jar file name must be specified.

**Required**: indicate which parameters are required by the command.

### Parameters

| **ID** | **Description** | **Name** | **Value** | **Other Values** |
| --- | --- | --- | --- | --- |
| --c | Primary connection used for the apps. | CONNECTION | Next |  |
| --p | Parameter or list of parameters (split by comma) | PARAMETER | Next |  |
| --sd | Schema destination. Where the extracted files will be loaded. This parameter is fundamental because it is included in the generated file name. Then the load process will look for this name in the file name pattern and load the info in the respective schema. | SCHEMADEST | Next |  |
| --NOHEADER | Indicate in the extract phase, if the file will include the header or not (By default header is generated). In the Load phase will indicate if the file contain a header and needs to be skipped. | NOHEADER | none |  |
| --SKIP | This parameter is used to indicate to the SQLLoader how many lines are desired to be skipped. By default the skip value depend of the NOHEADER parameter, if was indicated as header is included, then the skip value going to be 1, other way will be 0 | SKIP | Read |  |
| --ZIP | Indicate if the file will be zipped after the generation. By default no zip functionality | ZIP | none |  |
| --ARCHIVE | Indicate if after the generation or load, the archive functionality is desired. This process will copy and zip the file/work directory, into the archive folder | ARCHIVE | None |  |
| --NODELETE | This parameter indicate if after the generation, load or any process, the resulting files will be deleted. If this parameter is no indicated, the default behavior is to delete all | NODELETE | None |  |
| --COPY | Indicate if the generated file will be copy over any given path | COPY | Read |  |
| --MOVE | Indicate is the generated file will be moved to different path | MOVE | Read |  |
| --REPORTBY | Indicate how do you want to report Failure/Success | REPORTBY | read | MAIL,COPY,DATABASE |
| --REPORT | Indicate if you want to report any failure, success or both when job ran | REPORT | read | SUCCESS or ERROR or BOTH |
| --JOBNAME | The job name for reporting purpose | JOBNAME | read |  |
| --LOADLAST | This parameter indicate when multiples files from the same table were generated, if the application can only load the latest version. | LOADLAST | None |  |
| --USEHEADER | Indicate if the header contained in the file can be used as column name for the dynamic CTL construction. Instate of using the METADATA or custom CTL file | USEHEADER | None |  |
| --DATEFORMAT | This is used by the load process to indicate how the date is formatted | DATEFORMAT | Read |  |
| --REFRESH | Indicate how the destination table will be refreshed before loading the new data. Possible values are TRUNCATE or DELETE.  If truncate is specified, then the entire table will be erased  If delete is specified, then the table will be delete before load base on the conditions specified in the parameter DELETEBY | REFRESH | Read | TRUNCATE or DELETE |
| --DELETEBY | Provide the conditions under a table will be deleted before loading new values. Condition must be provided, same as any WHERE clause but without the word WHERE.  E.g.  --DELETEBY=”column1=value1 and column2=value2 …” | DELETEBY | Read |  |
| --ADDLOADDATE | In the load process, include the sysdate to specified column  E.g.  --ADDLOADDATE=columDate1 | ADDLOADDATE | Read |  |
| --CTL | I customized CTL file is required for the load process, this must be specified through this parameter. Notice that the CTL file must exists in the CTL folder | CTL | Read |  |

**Value**: If there is a value to be read, indicate in which position the value is coming. To any given value, containing spaces, it should be enclosed by “…”

None: no value to read

Next: read from next position parameters.

Read: read the parameter from the same position after the expected “=”

E.g. --PARAM=value or --PARAM=”value with spaces”

## CallExtractSql

## CallExtractTable

## CallSqlLoader

## CallStoredProcedure