DV8 2.0 Build User Guide

Table of contents

DV8 Console 3
Overview 3
Converter Commands 3
Basic Commands (4) 3
Cytoscape Command (1) 5
Minos Commands (2) 6
History Commands 7
svn Commands (3) 7
Git Commands (3) 10
Office Command (1) 12
Analysis Commands 13
Arch Issue Command (1) 13
Metrics Commands (3) 14
Debt Commands (2) 15
Arch Root Command (1) 16
Arch Report Command (1) 17
Matrix Commands (3) 17
Clustering Commands (2) 19

DV8 Console

Overview

Overview

DV8 console is a suite of command line interface (CLI) tools including the following major categories:

(1) Converter commands: Converting software source code dependency data to DV8 format data, e.g. DSM (`.dv8-dsm`) or Clustering (`.dv8-clsx`) files, or exporting a DV8 DSM or clustering into a DV8 standard JSON/XML format or into a spreadsheet. Valid input data can be in DV8 standard JSON/XML format, Cytoscape, Minos, History, and Target issue list. More details can be found in the following sections for these commands.

(2) Analysis commands: Analyzing software architectures, modeled as DSMs, using different analysis tools provided by DV8, including architecture metrics, anti-pattern detection, root detection, and debt calculation, etc. DV8 also provides an arch report command to help reduce the labor of running each of the specific commands individually. Using this command, you only need to prepare a configurable properties file as an input parameter.

(3) Matrix commands: Creating new DSMs by either merging multiple DSM files into one (such as merging a structural DSM with a history DSM), or splitting one DSM into several smaller DSMs.

(4) Clustering commands: Clustering a set of source files into either a package structure, or a design rule hierarchy structure.

To list all the commands, you can run the following command in a terminal window:

```
>dv8-console
```

To understand how to use a command, you can enter one of the following 3 commands:

```
>dv8-console help <command name>
```

or

```
>dv8-console <command name> -h
```

or

```
>dv8-console <command name> --help
```

Converter Commands

Basic Commands (4)

1. core:convert-cluster

Convert an input file (JSON/XML) to a clustering file (`*.dv8-clsx`).

Usage

```
dv8-console core:convert-cluster [-h] [-outputFile <OUTPUTFILE>] [-xml]
INPUT_FILE
```

Input

A Clustering file in JSON or XML format.

Options

(1) outputFile: The output file to create (`*.dv8-clsx`).

(2) xml: Convert from XML (instead of JSON) format. Without this option, a JSON file is expected.

Example

```
>dv8-console core:convert-cluster -outputFile clustering.dv8-clsx
clustering.json
```

2. core:convert-matrix

Convert an input file (JSON/XML) to a dependency matrix file (`*.dv8-dsm`).

Usage

```
dv8-console core:convert-matrix [-dependPath <MAPPING_FILE>] [-h] [-outputFile
<OUTPUTFILE>] [-xml] INPUT_FILE
```

Input

A DSM file in JSON or XML format.

Options

(1) dependPath: Dependency type mapping file location for customized dependency types.

(2) outputFile: The output file to create ((`*.dv8-clsx`)).

(3) xml: Convert from XML (instead of JSON) format. Without this option, a JSON file is expected.

Example

```
>dv8-console core:convert-matrix -outputFile structure-dsm.dv8-dsm structure-
dsm.json
```

3. export-cluster, core:export-cluster

Export a clustering as a separate XML or JSON file.

Usage

```
dv8-console core:export-cluster [-h] [-outputFile <OUTPUTFILE>] [-xml]
INPUT_FILE
```

Input

A Clustering (`*.dv8-clsx`) file generated by DV8.

Options

(1) outputFile: The output file to create.

(2) xml: Export in XML (instead of JSON) format. Without this option, a JSON file is expected.

Example

```
>dv8-console core:export-cluster -outputFile clustering.json clustering.dv8-
clsx
```

4. export-matrix, core:export-matrix

Export a dependency matrix as a separate XML or JSON file.

Usage

```
dv8-console core:export-matrix [-h] [-outputFile <OUTPUTFILE>] [-xml]
INPUT_FILE
```

Input

A DSM (`*.dv8-dsm`) file generated by DV8.

Options

(1) outputFile: The output file to create.

(2) xml: Export in XML (instead of JSON) format. Without this option, a JSON file is expected.

Example

```
>dv8-console core:export-matrix -outputFile structure-dsm.json structure-
dsm.dv8-dsm
```

Cytoscape Command (1)

1. cytoscape:convert-matrix

Convert a Cytoscape dependency file to a dependency matrix file (`*.dv8-dsm`).

Usage

```
dv8-console cytoscape:convert-matrix [-dependPath <MAPPING_FILE>] [-h] [-
outputFile <OUTPUTFILE>] INPUT_FILE
```

Input

A dependency file in Cytoscape format. You can use Understand as preprocessor to export the Cytoscape file from analyzed source code.

Options

(1) dependPath: Dependency type mapping file location for customized dependency types. This is optional as DV8 already provides a default dependency mapping covering the basic (most common) dependency mappings. If a missing dependency type is found during processing, you must provide a customized dependency mapping file; a sample of this file can be found in "*dv8-console\samples\dv8-dependency.mapping*".

(2) outputFile: The output file to create (`*.dv8-dsm`).

(3) Dextended.config: To match file names in different data sources, DV8 provides a preprocessor to fix the inconsistencies between the input files. These inconsistencies derive from the different prefixes of a file path in the dependency file and the revision history file. You may have to configure the prefix string to be removed in an extended configuration file to fix this common problem. This option is widely used in the converter commands of DV8. A sample of an extended configuration file can be found in "*dv8-console\samples\dv8-extended-config.xml*" (download). Specifically, you need to edit the `<value></value>` block with the prefix strings to be excluded. You can exclude multiple prefix strings using multiple `<value> </value>` blocks.

For example, if a file path in the dependency file is like this

"*C:\pdfbox\pdfbox\src\main\java\org\apache\pdfbox\pdmodel\PageMode.java*", while in the revision history file the same file path would be relative to the project root, e.g. "*pdfbox\src\main\java\org\apache\pdfbox\pdmodel\PageMode.java*", then you need to replace the `<value> </value>` block (line 6) in *[dv8-extended-config.xml]()* with "*C:\pdfbox\*" so that DV8 will automatically ignore it in the file path. For this option, the path should be enclosed in quotation marks (this is required in some terminals, like Windows PowerShell). See the section below for an example.

Example

```
>dv8-console cytoscape:convert-matrix -outputFile pdfbox_structure.dv8-dsm -
dependPath dv8-dependency.mapping "-Dextended.config=dv8-extended-config.xml"
cytoscape.xml
```

Minos Commands (2)

1. minos:convert-matrix

Convert a Minos-formatted DSM file to a dependency matrix file (`*.dv8-dsm`).

Usage

```
dv8-console minos:convert-matrix [-dependPath <MAPPING_FILE>] [-h] [-outputFile
<OUTPUTFILE>] DSM_FILE
```

Input

A dependency matrix file in Minos format; this file format is used by Titan.

Options

(1) dependPath: Dependency type mapping file location for customized dependency types. This is optional as DV8 already provides a default dependency mapping covering basic dependency mappings. If a missing dependency type is found during processing, you must provide a customized dependency mapping file, a sample of this file can be found in "*dv8-console\samples\dv8-dependency.mapping*".

(2) outputFile: The output file to create (`*.dv8-dsm`).

(3) Dextended.config: To match file names in different data sources, DV8 provides a preprocessor to fix the inconsistencies between the input files. These inconsistencies derive from the different prefixes of a file path in the dependency file and the revision history file. You may have to configure the prefix string to be removed in an extended configuration file to fix this common problem. This option is widely used in the converter commands of DV8. A sample of an extended configuration file can be found in "*dv8-console\samples\dv8-extended-config.xml*" (download). Specifically, you need to edit the `<value> </value>` block with the prefix strings to be excluded. You can exclude multiple prefix strings using multiple `<value> </value>` blocks.

For example, if a file path in the dependency file is like this "*C:\pdfbox\pdfbox\src\main\java\org\apache\pdfbox\pdmodel\PageMode.java*", while in the revision history file the same file path would be relative to the project root, e.g. "*pdfbox\src\main\java\org\apache\pdfbox\pdmodel\PageMode.java*", then you need to replace the `<value> </value>` block (line 6) in *dv8-extended-config.xml* with "*C:\pdfbox\*" so that DV8 will automatically ignore it in the file path. For this option, the path should be enclosed in quotation marks (this is required in some terminals, like Windows PowerShell). See the section below for an example.

Example

```
>dv8-console minos:convert-matrix -outputFile structure-dsm.dv8-dsm -dependPath
dv8-dependency.mapping "-Dextended.config=dv8-extended-config.xml" minos.dsm
```

2. minos:convert-cluster

Convert a Minos-formatted CLSX file to a clustering file (`*.dv8-clsx`).

Usage

```
dv8-console minos:convert-cluster [-h] [-outputFile <OUTPUTFILE>] CLSX_FILE
```

Input

A clustering file in Minos format; this file format is used by Titan.

Options

-outputFile: The output file to create (`*.dv8-clsx`).

Example

```
>dv8-console minos:convert-cluster -outputFile clustering.dv8-clsx
clustering.clsx
```

History Commands

svn Commands (3)

1. svn-convert-matrix, scm:history:svn:convert-matrix

Convert a Subversion log file—(in XML format), which can be exported using the svn command: `svn log --xml -v repo`) —to a dependency matrix file.

Usage

```
dv8-console scm:history:svn:convert-matrix [-h] [-matrix <MATRIX>] [-
maxCochangeCount <MAXCOCHANGECOUNT>] [-outputFile <OUTPUTFILE>] [-
paramsOutputFile <PARAMSOUTPUTFILE>] [-start <DATETIME>] [-stop <DATETIME>]
INPUT_FILE
```

Input

A text file of commit log exported from software revision history. You can use `svn log --xml -v repo` to get the text from a svn repository. The file should be encoded in UTF-8 format.

Options

(1) matrix: Path of a structure matrix that models file dependencies of a particular snapshot. With this matrix, the files that in the revision history file but not in the structure matrix will be excluded. Without this option, DV8 will extract a history DSM with all the files in the revision history log, which may consume a very long time and a large amount of memory if the project has a very long history.

(2) maxCochangeCount: Maximum count of co-changed files per commit in history, default is 1000. You may reduce this number if you encounter performance issue due to hardware limitations.

(3) outputFile: The output dependency matrix file to create (`*.dv8-dsm`); this defaults to the same filename as the input history file under current working path.

(4) paramsOutputFile: The output file (*.json) used to record the parameters used when executing this command. You may not enter all the parameters, e.g., the start and end dates of the history. In this case, this command will use all the history in the log file, and the dates will be recorded in this file.

(5) start: Start of date/time range to convert (in ISO-8601 date/time format) (e.g. 2017-07-08T00:00:00Z).

(6) stop: End of date/time range to convert (in ISO-8601 date/time format) (e.g. 2018-01-08T00:00:00Z).

(7) Dextended.config: To match file names in different data sources, DV8 provides a preprocessor to fix the inconsistencies between the input files. These inconsistencies derive from the different prefixes of a file path in the dependency file and the revision history file. You may have to configure the prefix string to be removed in an extended configuration file to fix this common problem. This option is widely used in the converter commands of DV8. A sample of an extended configuration file can be found in "*dv8-console\samples\dv8-extended-config.xml*" (download). Specifically, you need to edit the `<value> </value>` block with the prefix strings to be excluded. You can exclude multiple prefix strings using multiple `<value> </value>` blocks.

For example, if a file path in the dependency file is like this "*C:\pdfbox\pdfbox\src\main\java\org\apache\pdfbox\pdmodel\PageMode.java*", while in the revision history file the same file path would be relative to the project root, e.g. "*pdfbox\src\main\java\org\apache\pdfbox\pdmodel\PageMode.java*", then you need to replace the `<value> </value>` block (line 6) in *dv8-extended-config.xml* with "*C:\pdfbox\*" so that DV8 will automatically ignore it in the file path. For this option, the path should be enclosed in quotation marks (this is required in some terminals, like Windows PowerShell). See the section below for an example.

Example

```
>dv8-console scm:history:svn:convert-matrix -outputFile history.dv8-dsm "-
Dextended.config=dv8-extended-config.xml" gitlog.txt
```

2. svn-generate-changelist, scm:history:svn:generate-changelist

Generate a CSV-formatted change list file from a subversion log (in XML format), which can be exported using svn command: `svn log --xml -v repo`.

Usage

```
dv8-console scm:history:svn:generate-changelist [-h] [-outputFolder
<OUTPUTFOLDER>] [-start <DATETIME>] [-stop <DATETIME>] INPUT_FILE
```

Input

A text file of the commit log exported from your software revision history. You can use `svn log --xml -v repo` to get the text from your svn repository. The file should be encoded in UTF-8 format.

Options

(1) outputFolder: The output folder of the change list, including change frequency and change churn.

(2) start: Start of date/time range to convert (in ISO-8601 date/time format) (e.g. 2017-07-08T00:00:00Z).

(3) stop: End of date/time range to convert (in ISO-8601 date/time format) (e.g. 2018-01-08T00:00:00Z).

(4) Dextended.config: To match file names in different data sources, DV8 provides a preprocessor to fix the inconsistencies between the input files. These inconsistencies derive from the different prefixes of a file path in the dependency file and the revision history file. You may have to configure the prefix string to be removed in an extended configuration file to fix this common problem. This option is widely used in the converter commands of DV8. A sample of an extended configuration file can be found in "*dv8-console\samples\dv8-extended-config.xml*" (download). Specifically, you need to edit the `<value> </value>` block with the prefix strings to be excluded. You can exclude multiple prefix strings using multiple `<value> </value>` blocks.

For example, if a file path in the dependency file is like this "*C:\pdfbox\pdfbox\src\main\java\org\apache\pdfbox\pdmodel\PageMode.java*", while in the revision history file the same file path would be relative to the project root, e.g. "*pdfbox\src\main\java\org\apache\pdfbox\pdmodel\PageMode.java*", then you need to replace the `<value>`

`</value>` block (line 6) in *dv8-extended-config.xml* with "*C:\pdfbox\*" so that DV8 will automatically ignore it in the file path. For this option, the path should be enclosed in quotation marks (this is required in some terminals, like Windows PowerShell). See the section below for an example.

Example

```
>dv8-console scm:history:svn:generate-changelist -outputFolder list "-
Dextended.config=dv8-extended-config.xml" gitlog.txt
```

3. svn-generate-targetlist, scm:history:svn:generate-targetlist

Generate a CSV-formatted target list file from a regular expression (used to match issue ids) and a subversion log (in XML format), which can be exported using svn command: `svn log --xml -v repo`.

Usage

```
dv8-console scm:history:svn:generate-targetlist [-h] [-outputFolder
<OUTPUTFOLDER>] [-start <DATETIME>] [-stop <DATETIME>] INPUT_FILE
```

Input

A text file of a commit log exported from your software revision history. You can use `svn log --xml -v repo` to get the text from your svn repository. The file should be encoded in UTF-8 format.

Options

(1) outputFolder: The output folder of the target list, including target frequency and target churn.

(2) regex: Regular expression used to extract issue ids from commit messages in your revision history to match the target id.

(3) start: Start of date/time range to convert (in ISO-8601 date/time format) (e.g. 2017-07-08T00:00:00Z).

(4) stop: End of date/time range to convert (in ISO-8601 date/time format) (e.g. 2018-01-08T00:00:00Z).

(5) targetissuecsv: The path of the target issue csv file, with the first column being issue id.

(6) Dextended.config: To match file names in different data sources, DV8 provides a preprocessor to fix the inconsistencies between the input files. These inconsistencies derive from the different prefixes of a file path in the dependency file and the revision history file. You may have to configure the prefix string to be removed in an extended configuration file to fix this common problem. This option is widely used in the converter commands of DV8. A sample of an extended configuration file can be found in "*dv8-console\samples\dv8-extended-config.xml*" (download). Specifically, you need to edit the `<value> </value>` block with the prefix strings to be excluded. You can exclude multiple prefix strings using multiple `<value> </value>` blocks.

For example, if a file path in the dependency file is like this "*C:\pdfbox\pdfbox\src\main\java\org\apache\pdfbox\pdmodel\PageMode.java*", while in the revision history file the same file path would be relative to the project root, e.g. "*pdfbox\src\main\java\org\apache\pdfbox\pdmodel\PageMode.java*", then you need to replace the `<value> </value>` block (line 6) in *dv8-extended-config.xml* with "*C:\pdfbox\*" so that DV8 will automatically ignore it in the file path. For this option, the path should be enclosed in quotation marks (this is required in some terminals, like Windows PowerShell). See the section below for an example.

Example

```
>dv8-console scm:history:svn:generate-targetlist -regex PDFBOX-[0-9]+ -
outputFolder list "-Dextended.config=dv8-extended-config.xml" gitlog.txt
```

Git Commands (3)

1. gittxt-convert-matrix, scm:history:gittxt:convert-matrix

Convert a Git log file (in txt format)—which can be exported using the git command: `git log --numstat --date=iso`—to a dependency matrix file.

Usage

```
dv8-console scm:history:gittxt:convert-matrix [-h] [-matrix <MATRIX>] [-
maxCochangeCount <MAXCOCHANGECOUNT>] [-outputFile <OUTPUTFILE>] [-
paramsOutputFile <PARAMSOUTPUTFILE>] [-start <DATETIME>] [-stop <DATETIME>]
INPUT_FILE
```

Input

A text git log exported from your software revision history. You can use `git log --numstat --date=iso` to get the text from your git repository. The file should be encoded in UTF-8 format.

Options

(1) matrix: Path of a structure matrix that models file dependencies of a particular snapshot. With this matrix, the files that in the revision history file but not in the structure matrix will be excluded. Without this option, DV8 will extract a history DSM with all the files in the revision history log, which may consume a very long time and a large amount of memory if the project has a very long history.

(2) maxCochangeCount: Maximum count of co-changed files per commit in history, default is 1000. You may reduce this number if you encounter performance issues due to hardware limitations.

(3) outputFile: The output dependency matrix file to create (`*.dv8-dsm`); this defaults to the same filename as the input history file under current working path.

(4) paramsOutputFile: The output file (`*.json`) used to record the parameters used when executing this command. You may not enter all the parameters, e.g., the start and end dates of the history. In this case, this command will use all the history in the log file, and the dates will be recorded in this file.

(5) start: Start of date/time range to convert (in ISO-8601 date/time format) (e.g. 2017-07-08T00:00:00Z).

(6) stop: End of date/time range to convert (in ISO-8601 date/time format) (e.g. 2018-01-08T00:00:00Z).

(7) Dextended.config: To match file names in different data sources, DV8 provides a preprocessor to fix the inconsistencies between the input files. These inconsistencies derive from the different prefixes of a file path in the dependency file and the revision history file. You may have to configure the prefix string to be removed in an extended configuration file to fix this common problem. This option is widely used in the converter commands of DV8. A sample of an extended configuration file can be found in "*dv8-console\samples\dv8-extended-config.xml*" (download). Specifically, you need to edit the `<value> </value>` block with the prefix strings to be excluded. You can exclude multiple prefix strings using multiple `<value> </value>` blocks.

For example, if a file path in the dependency file is like this "*C:\pdfbox\pdfbox\src\main\java\org\apache\pdfbox\pdmodel\PageMode.java*", while in the revision history file the same file path would be relative to the project root, e.g. "*pdfbox\src\main\java\org\apache\pdfbox\pdmodel\PageMode.java*", then you need to replace the `<value> </value>` block (line 6) in *dv8-extended-config.xml* with "*C:\pdfbox\*" so that DV8 will automatically ignore it in the file path. For this option, the path should be enclosed in quotation marks (this is required in some terminals, like Windows PowerShell). See the section below for an example.

Example

```
>dv8-console scm:history:gittxt:convert-matrix -outputFile history.dv8-dsm "-
```

```
Dextended.config=dv8-extended-config.xml" gitlog.txt
```

2. gittxt-generate-changelist, scm:history:gittxt:generate-changelist

Generate a CSV-formatted change list file from a git log (`git log --numstat --date=iso`).

Usage

```
dv8-console scm:history:gittxt:generate-changelist [-h] [-outputFolder
<OUTPUTFOLDER>] [-start <DATETIME>] [-stop <DATETIME>] INPUT_FILE
```

Input

A text file of a git log exported from your software revision history. You can use `git log --numstat --date=iso` to get the text from your git repository. The file should be encoded in UTF-8 format.

Options

(1) outputFolder: The output folder of change list, including change frequency and change churn.

(2) start: Start of date/time range to convert (in ISO-8601 date/time format) (e.g. 2017-07-08T00:00:00Z).

(3) stop: End of date/time range to convert (in ISO-8601 date/time format) (e.g. 2018-01-08T00:00:00Z).

(4) Dextended.config: To match file names in different data sources, DV8 provides a preprocessor to fix the inconsistencies between the input files. These inconsistencies derive from the different prefixes of a file path in the dependency file and the revision history file. You may have to configure the prefix string to be removed in an extended configuration file to fix this common problem. This option is widely used in the converter commands of DV8. A sample of an extended configuration file can be found in "*dv8-console\samples\dv8-extended-config.xml*" (download). Specifically, you need to edit the `<value> </value>` block with the prefix strings to be excluded. You can exclude multiple prefix strings using multiple `<value> </value>` blocks.

For example, if a file path in the dependency file is like this "*C:\pdfbox\pdfbox\src\main\java\org\apache\pdfbox\pdmodel\PageMode.java*", while in the revision history file the same file path would be relative to the project root, e.g. "*pdfbox\src\main\java\org\apache\pdfbox\pdmodel\PageMode.java*", then you need to replace the `<value> </value>` block (line 6) in *dv8-extended-config.xml* with "*C:\pdfbox\*" so that DV8 will automatically ignore it in the file path. For this option, the path should be enclosed in quotation marks (this is required in some terminals, like Windows PowerShell). See the section below for an example.

Example

```
>dv8-console scm:history:gittxt:generate-changelist -outputFolder list "-
Dextended.config=dv8-extended-config.xml" gitlog.txt
```

3. gittxt-generate-targetlist, scm:history:gittxt:generate-targetlist

Generate a CSV-formatted target list file from a regular expression (used to match issue ids) and a git log (`git log --numstat --date=iso`).

Usage

```
dv8-console scm:history:gittxt:generate-targetlist [-h] [-outputFolder
<OUTPUTFOLDER>] [-regex <REGEX>] [-start <DATETIME>] [-stop <DATETIME>] [-
targetissuecsv <TARGETISSUECSV>] INPUT_FILE
```

Input

A text file of a commit log exported from software revision history. You can use `git log --numstat --date=iso` to get the text from your git repository. The file should be encoded in UTF-8 format.

Options

(1) outputFolder: The output folder of the target list, including target frequency and target churn.

(2) regex: Regular expression used to extract ids from commit messages in the revision history to match the target id.

(3) start: Start of date/time range to convert (in ISO-8601 date/time format) (e.g. 2017-07-08T00:00:00Z).

(4) stop: End of date/time range to convert (in ISO-8601 date/time format) (e.g. 2018-01-08T00:00:00Z).

(5) targetissuecsv: The path of the target issue csv file with the first column being issue id.

(6) Dextended.config: To match file names in different data sources, DV8 provides a preprocessor to fix the inconsistencies between the input files. These inconsistencies derive from the different prefixes of a file path in the dependency file and the revision history file. You may have to configure the prefix string to be removed in an extended configuration file to fix this common problem. This option is widely used in the converter commands of DV8. A sample of an extended configuration file can be found in "*dv8-console\samples\dv8-extended-config.xml*" (download). Specifically, you need to edit the `<value> </value>` block with the prefix strings to be excluded. You can exclude multiple prefix strings using multiple `<value> </value>` blocks.

For example, if a file path in the dependency file is like this "*C:\pdfbox\pdfbox\src\main\java\org\apache\pdfbox\pdmodel\PageMode.java*", while in the revision history file the same file path would be relative to the project root, e.g. "*pdfbox\src\main\java\org\apache\pdfbox\pdmodel\PageMode.java*", then you need to replace the `<value> </value>` block (line 6) in *dv8-extended-config.xml* with "*C:\pdfbox\*" so that DV8 will automatically ignore it in the file path. For this option, the path should be enclosed in quotation marks (this is required in some terminals, like Windows PowerShell). See the section below for an example.

Example

```
>dv8-console scm:history:gittxt:generate-targetlist -regex PDFBOX-[0-9]+ -
outputFolder list "-Dextended.config=dv8-extended-config.xml" gitlog.txt
```

Office Command (1)

1. export-spreadsheet, office:export-spreadsheet

Export a matrix into a spreadsheet.

Usage

```
dv8-console export-spreadsheet [-cluster <CLUSTER>] [-detail] [-drhier] [-h] [-
namespace] [-outputFile <OUTPUTFILE>] DSM_FILE
```

Input

A DSM (*.dv8-dsm) file generated by DV8.

Options

(1) cluster: The clustering file used to order files in the output spreadsheet.

(2) detail: Display detailed dependencies within each cell in the spreadsheet generated.

(3) drhier: In the output spreadsheet, files are clustered using the recursive drh clustering.

(4) namespace: In the output spreadsheet, files are clustered using the namespace clustering.

(5) outputFile: The output spreadsheet file to create (`*.xlsx`).

Note: you can use one of the three options to set how the files will be clustered, "`-cluster`", "`-drhier`", or "`-namespace`". If more than one parameters are set, the priority order is as follows: "`-cluster`" > "`-drhier`" > "`-namespace`"

Example

```
>dv8-console export-spreadsheet -outputFile structure-dsm.xlsx structure-
dsm.dv8-dsm
```

Analysis Commands

Arch Issue Command (1)

1. arch-issue, arch-issue:arch-issue

Detect Architecture anti-patterns, as defined in section Architecture Anti-patterns.

Usage

```
dv8-console arch-issue:arch-issue [-cliqueDepends <CLIQUE-DEPENDTYPE>] [-
crossingCochange <CROSSING-COCHANGE>] [-crossingFanIn <CROSSING-FANIN>] [-
crossingFanOut <CROSSING-FANOUT>] [-h] [-mvCochange <MODULARITYVIOLATION-
COCHANGE>] -outputFolder <OUTPUTFOLDER> [-uiCochange <UNSTABLEINTERFACE-
COCHANGE>] [-uihDepends <UNHEALTHYINHERITANCE-DEPENDTYPE>] [-uihInheritance
<UNHEALTHYINHERITANCE-INHERITANCETYPE>] [-uiHistoryImpact <UNSTABLEINTERFACES-
HISTORYIMPACT>] [-uiStructImpact <UNSTABLEINTERFACE-STRUCTIMPACT>] DSM_FILE
```

Input

A DSM file (`*.dv8-dsm`) file generated by DV8. If the DSM file contains history data, then the output anti-pattern DSMs will also have history information, and DV8 will detect all six types of anti-patterns. Otherwise only the three types of anti-patterns without history will be detected.

Options

(1) cliqueDepends: (For Clique detection) Filtered dependencies for clique detection. Multiple dependencies should be delimited using ",". Ex. imports,includes,writes. Use "*" to select all the dependency types. Default value is "call,use".

(2) crossingCochange: (For Crossing detection) Threshold of co-change between two files. Default is 2.

(3) crossingFanIn: (For Crossing detection) The number of other files that depend on it >= "crossingFanIn". Default is 4.

(4) crossingFanOut: (For Crossing detection) The number of other files it depends on >= "crossingFanOut". Default is 4.

(5) mvCochange: (For Modularity Violation detection) Threshold of co-change between two files. Default is 2.

(6) outputFolder: The output directory, the data of detected ArchIssues will be saved in this directory.

(7) uiCochange: (For Unstable Interface detection) Threshold of co-change between two files. Default is 2.

(8) uihDepends: (For Unhealthy Inheritance detection) Filtered dependencies for unhealthy inheritance detection. Multiple dependencies should be delimited using ",". Ex. imports,includes,writes. Use "*" to select all the dependency types. Default value is "call,use".

(9) uihInheritance: (For Unhealthy Inheritance detection) Dependencies for unhealthy inheritance detection. Multiple dependencies should be delimited using ",". Ex. extend,implement,base. Default value is "extend,implement,public,private,virtual public".

(10) historyImpact: (For Unstable Interface detection) Threshold of the number of co-changed (more than "co-change" times) files. For example, 10 means at least 10 other co-changed files. Default is 10.

(11) uiStructImpact: (For Unstable Interface detection) Threshold of value < 1 means % of all files are dependents. For example, 0.01 means 1% of all files are dependents; a value >=1 refers to the absolute number of dependents. For example, 10 means a file has at least 10 dependents. Default is 0.01.

Example

```
>dv8-console arch-issue:arch-issue -outputFolder arch-issue -uiCochange 2 -
crossingCochange 2 -mvCochange 2 -uiStructImpact 0.01 -historyImpact 10 -
crossingFanIn 4 -crossingFanOut 4 -cliqueDepends "call,use" -uihDepends
"call,use" -uihInheritance "extend,implement,public,private,virtual public"
merged-dsm.dv8-dsm
```

Metrics Commands (3)

1. decoupling-level, metrics:decoupling-level

Calculates the Decoupling Level (DL) score of a dependency matrix.

Usage

```
dv8-console metrics:decoupling-level [-h] [-outputFile <OUTPUTFILE>] INPUT_FILE
```

Input

A DSM file (`*.dv8-dsm`) file generated by DV8.

Options

(1) outputFile: The output file to create (`*.json`). Defaults to the same filename as the input dependency matrix file in the current working folder.

Example

```
>dv8-console metrics:decoupling-level structure-dsm.dv8-dsm
```

2. independence-level, metrics:independence-level

Calculates the Independence Level (IL) score of a dependency matrix.

Usage

```
dv8-console metrics:independence-level [-h] [-outputFile <OUTPUTFILE>]
INPUT_FILE
```

Input

A DSM (*.dv8-dsm) file generated by DV8.

Options

(1) outputFile: The output file to create (`*.json`). The output file to create (`*.json`). Defaults to the same filename as the input dependency matrix file in the current working folder.

Example

```
>dv8-console metrics:independence-level structure-dsm.dv8-dsm
```

3. propagation-cost, metrics:propagation-cost

Calculates the Propagation Cost (PC) score of a dependency matrix.

Usage

```
dv8-console metrics:propagation-cost [-h] [-outputFile <OUTPUTFILE>] INPUT_FILE
```

Input

A DSM (*.dv8-dsm) file generated by DV8.

Options

(1) outputFile: The output file to create (`*.json`). Defaults to the same filename as the input dependency matrix file in the current working folder.

Example

```
>dv8-console metrics:propagation-cost structure-dsm.dv8-dsm
```

Debt Commands (2)

1. arch-issue-cost, debt:arch-issue-cost

Calculate the cost of each issue instance, export into a `.csv` file, and summarize to a spreadsheet (`.xlsx`).

Usage

```
dv8-console debt:arch-issue-cost -asDir <ASDIR> [-BC <BC>] [-BF <BF>] [-
bugCover <BUGCOVER>] [-CC <CC>] [-CF <CF>] [-changeCover <CHANGECOVER>] [-csv]
[-h] -outputFolder <OUTPUTFOLDER> DSM_FILE
```

Input

A DSM (`*.dv8-dsm`) file generated by DV8.

Options

(1) asDir: Path to the folder containing all detected issue instances.

(2) BC: Path to the bug churn list.

(3) BF: Path to the bug frequency list.

(4) bugCover: Threshold of bug-prone files: used for calculating bug coverage and bugginess.

(5) CC: Path to the change churn list.

(6) CF: Path to the change frequency list.

(7) changeCover: Threshold of change-prone files: used for calculating change coverage and change-

proneness.

(8) csv: Keep a *.csv copy of the summary report (default is *.xlsx only).

(9) outputFolder: The output folder containing all reports.

Example

```
>dv8-console debt:arch-issue-cost -asDir arch-issue -outputFolder arch-issue-
cost -BF targetcommitfreqlist.csv -BC targetchurnlist.csv -CF
changefreqlist.csv -CC changechurnlist.csv structure-dsm.dv8-dsm
```

2. arch-root-debt, debt:arch-root-debt

Calculate the return on investment of an architecture debt, in the form of architecture roots.

Usage

```
dv8-console debt:arch-root-debt [-BC <BC>] [-BF <BF>] [-CC <CC>] [-CF <CF>] [-
csv] [-h] -outputFile <OUTPUTFILE> -rootDir <ROOTDIR> -rootIndex <ROOTINDEX>
DSM_FILE
```

Input

A DSM (`*.dv8-dsm`) file generated by DV8.

Options

(1) BC: Path to the bug churn list.

(2) BF: Path to the bug frequency list.

(3) CC: Path to the change churn list.

(4) CF: Path to the change frequency list.

(5) csv: Keep *.csv copies of the debt report file (default is *.xlsx only).

(6) outputFile: The output debt report file.

(7) rootDir: Path to the root folder containing the root project files.

(8) rootIndex: Path to the rootIndex file.

Example

```
>dv8-console debt:arch-root-debt -rootDir arch-root -rootIndex arch-root\root-
index.csv -BF targetcommitfreqlist.csv -BC targetchurnlist.csv -CF
changefreqlist.csv -CC changechurnlist.csv -outputFile arch-root\root-debt.xlsx
structure-dsm.dv8-dsm
```

Arch Root Command (1)

1. arch-root, arch-root:arch-root

RootCover Service.

Usage

```
dv8-console arch-root:arch-root [-frequency <FREQUENCY>] [-h] -outputFolder
<OUTPUTFOLDER> [-percentage <PERCENTAGE>] -targetlist <TARGETLIST> DSM_FILE
```

Input

A DSM (`*.dv8-dsm`) file generated by DV8. If the DSM file contains history data, then the output root DSMs will also have history information.

Options

(1) frequency: Target frequency threshold to determine error-proneness; default is 2. The files that have changed fewer than threshold times in the target issues will be ignored. For example, if the target is bug issues, a frequency threshold of 2 means that files that have changed for fewer than 2 bug fixes will be ignored.

(2) outputFolder: The folder where the root list file will be exported.

(3) percentage: The percentage of target issues to be covered, default is 0.8.

(4) targetlist: The target issue list file.

Example

```
>dv8-console arch-root:arch-root -frequency 2 -targetlist
targetcommitfreqlist.csv -percentage 0.8 -outputFolder root\arch-root
structure-dsm.dv8-dsm
```

Arch Report Command (1)

1. arch-report, arch-report:arch-report

Generate an architecture analysis report.

Usage

```
dv8-console arch-report:arch-report [-h] -paramsFile <PARAMSFILE>
```

Options

(1) paramsFile: The parameters file path. Please refer to Report Configuration File that explains how to construct this file.

Example

```
>dv8-console arch-report:arch-report -paramsFile archreport.properties
```

Matrix Commands (3)

1. merge-matrix, core:merge-matrix

Merges two or more matrices into one new matrix.

Usage

```
dv8-console core:merge-matrix [-h] -outputFile <OUTPUTFILE> MATRIX_NAME1
MATRIX_NAME2...
```

Input

Multiple DSM (`*.dv8-dsm`) files generated by DV8.

Options

(1) outputFile: The output file to create (`*.dv8-dsm`).

Example

```
>dv8-console core:merge-matrix -outputFile merged-dsm.dv8-dsm structure-
dsm.dv8-dsm history-dsm.dv8-dsm
```

2. dr-split, dr-hier:dr-split

Splits a matrix into one or more sub-matrices, based on selected design rules.

Usage

```
dv8-console dr-hier:dr-split [-h] -outputFile <OUTPUTFILE> -rule <RULES...>
INPUT_FILE
```

Input

A DSM (*.dv8-dsm) file generated by DV8.

Options

(1) outputFile: The output dependency matrix file to create (*.dv8-dsm).

(2) rule: Design rules separated by space, wrapped in double quotes if multiple rules are given.

Example

```
>dv8-console dr-hier:dr-split -outputFile pdfbox-split.dv8-dsm -rule
debugger.src.main.java.org.apache.pdfbox.debugger.hexviewer.HexChangeListener_j
ava structure-dsm.dv8-dsm
```

3. split-subsystem, core:split-subsystem

Separate a big matrix into one or more sub-matrices based on specified folders or files.

Usage

```
dv8-console core:split-subsystem -delimiter <DELIMITER> [-h] -outputFile
<OUTPUTFILE> -subsystem <SUBSYSTEMS...> INPUT_FILE
```

Input

A DSM (`*.dv8-dsm`) file generated by DV8.

Options

(1) delimiter: Delimiter for splitting (e.g., path separator, package separator)

(2) outputFile: The output file to create (`*.dv8-dsm`).

(3) subsystem: Subsystems separated by space, wrapped in double quotes if multiple subsystems are given.

Example

```
>dv8-console core:split-subsystem -delimiter . -outputFile pdfbox-split.dv8-dsm
-subsystem
debugger.src.main.java.org.apache.pdfbox.debugger.hexviewer.HexChangeListener_j
ava structure-dsm.dv8-dsm
```

Clustering Commands (2)

1. namespace-cluster, core:namespace-cluster

Computes the namespace cluster for a dependency matrix.

Usage

```
dv8-console core:namespace-cluster [-h] [-outputFile <OUTPUTFILE>] INPUT_FILE
```

Input

A DSM (`*.dv8-dsm`) file generated by DV8.

Options

(1) outputFile: The output clustering file to create (`*.dv8-clsx`). Defaults to the same filename as the input dependency matrix file in the current working folder.

Example

```
>dv8-console core:namespace-cluster -outputFile clustering.dv8-clsx structure-
dsm.dv8-dsm
```

2. dr-hier, dr-hier:dr-hier

Computes the design rule hierarchy for a dependency matrix.

Usage

```
dv8-console dr-hier:dr-hier [-h] [-maxDepth <INTEGER>] [-modules] [-outputFile
<OUTPUTFILE>] [-recursive] INPUT_FILE
```

Input

A DSM (`*.dv8-dsm`) file generated by DV8.

Options

(1) maxDepth: Recursion depth limit.

(2) modules: Merge modules following the same set of design rules, if switched on.

(3) outputFile: The output clustering file to create (`*.dv8-clsx`). Defaults to the same filename as the input dependency matrix file in the current working folder.

(4) recursive: Use recursive algorithm if switched on.

Example

```
>dv8-console dr-hier:dr-hier -outputFile clustering.dv8-clsx structure-dsm.dv8-
dsm
```