CSV2DCC User Guide

**Introduction**

The purpose of the CSV2DCC application is to copy calibration result information from a CSV file\* into the relevant places within a DCC template. The application has been written to be applicable to all types of XML type DCC file

\* Actually a semi-colon separated file will be used to avoid regional specific use of commas and decimal points across Europe. For example

UK representation 1,200,300.456

Deutsch representation 1.200.300,456

**Important Note:-** If every DCC you generate (excluding the results) is unique, with differing structure and length, then CSV2DCC in its current version (1.0) is unlikely to be of use. to you. If you generate many DCCs with the same structure (but differing results) then CSV2DCC will probably be of use to you.

**Installation**

See Appendix A for instructions on how to install the CSV2DCC application.

**How to use CSV2DCC**

In order to explain the principles of how to use the CSV2DCC application a very small XML file will be used as an example.

Consider the following example xml file

<?xml version="1.0"?>

<Employee\_Info

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="employee.xs">

<Employee Employee\_Number="105">

<Name>Masashi Okamura</Name>

<Department>Design Department</Department>

<Telephone>03-1452-4567</Telephone>

<Email>okamura@xmltr.co.jp</Email>

</Employee>

</Employee\_Info>

Script 1. A basic XML file containing data

Although there is only a small amount data, this XML file is complete and for the purposes of demonstrating how CSV2DCC works is adequate.

We now show how to use the CSV2DCC application in 3 steps (not quite to heaven but as close as we can get! :o).

**Step 1. Create a Parameterised XML DCC file**

First we need to convert this example into a parameterised version as described below.

We will replace all items of data with separate codes, as shown below in Script 2. By comparing Script 1 and Script 2 you will see that all data in the elements has been replaced with a code; and also the single example of an attribute "Employee\_Number" with data (105) has also been replaced. Let's call this file "EmployeePar.xml". It is recommended that "Par" is placed in the filename to indicate this is a parameterised file. (CSV2DCC will check to see that a parameter file includes the string "par" in the filename). Doing this makes it easier to see which files are the Parameterised versions.

<?xml version="1.0"?>

<Employee\_Info

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="employee.xs">

<Employee Employee\_Number="XYX01">

<Name>XYX02</Name>

<Department>XYX03</Department>

<Telephone>XYX04</Telephone>

<Email>XYX05</Email>

</Employee>

</Employee\_Info>

Script 2. The basic file with the data replaced with codes creating a Parameterised version.

Each code consists of a Stub "XYX" and an incremented digit. The stub was chosen so that it is unique in the file. You can (and should) use any stub you like. Choose a stub where the sequence of letters does not appear anywhere in the original file.

The stub text and be any length and the incrementing digit sequence can be as long as required. So for example,

XYZYLPLMHBYGX0000000019

will work, but obviously probably not necessary as this will increase the amount of typing required and the increase the chance of making a typing mistake. Choose a length that works for you.

**Note:-** If you have a sequence of data in an element, say

<Promotion\_Dates>1971 1980 1995</Promotion\_Dates)

Simply replace the whole sequence with a single code e.g.

<Promotion\_Dates>XYX55</Promotion\_Dates)

**Step 2. Create a Data Substitution CSV file**

We now need to create the substitution CSV file using "Excel" or "LibreCalc" or any similar spreadsheet application.

In a new "sheet" in your spreadsheet application, in Column A list all the codes you used in the parameterised XML file.

In Column B, put all the relevant data as shown in the example below in Figure 2.

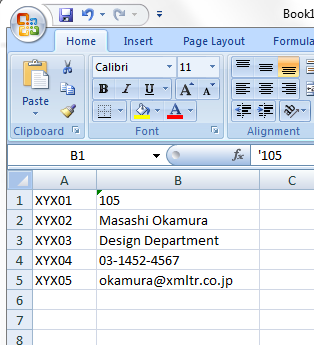


Figure 1. An example Substitution File

Please note that the entries in column B must all be text (XML files are by definition all text). It is necessary to 'force' Excel (or whichever application you are using) to treat numbers as text. This is done by placing an apostrophe (') in front of the number as shown in the figure 1 above. So instead of 105 the B1 cell contains '105.

You can easily see if an entry is being treated as a number (and not text) if it appears on the right hand side of the cell that contains it. So to check your file, simply scan down the Column B containing the substitution text and make sure all entries start from on the left hand side of the cell, as in the example above.

We'll save this as EmployeeSubs.csv. As was mentioned above, we need to actually use semicolons ";" as the separator and not commas. See Appendix B for instructions on how to do this if you need to.

**Step 3. Running the application CSV2DCC**

Now we are ready to use CSV2DCC. If you haven't done so already, please follow the instructions in Appendix B to install the application on a Windows PC.

Start the application by opening windows explorer, navigate to the folder where you installed the application and double click on CSV2DCC.bat. The screen in figure 2 will appear.

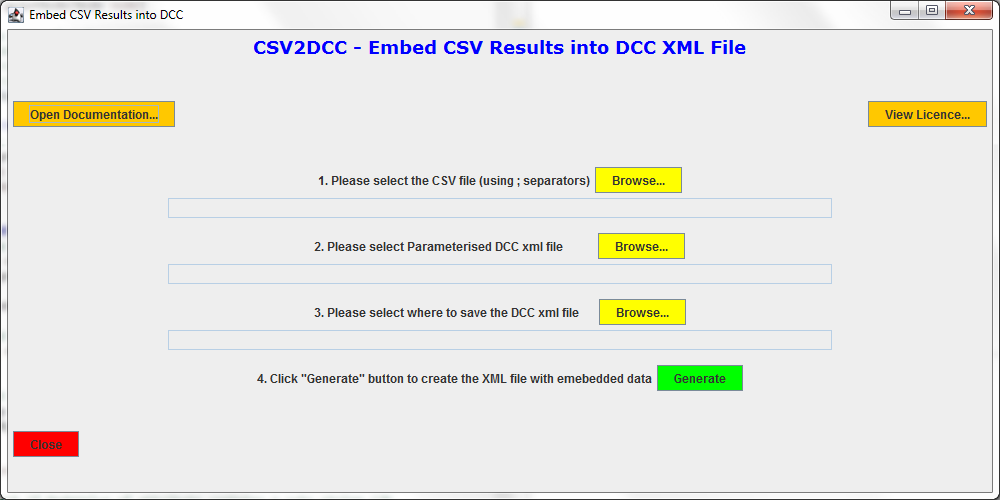


Figure 2. CSV2DCC - Main Screen

Use the first two yellow "Browse" buttons to individually select the CSV file (EmployeeSubs.csv), and the parameter file (EmployeePar.xml).

Using the 3rd yellow "Browse" button navigate to where you want the output to be saved and provide a name for the output file, e.g. "EmployeeDataOut.XML". You can select a file that already exists (Careful! not to overwrite the parameter file - there will be a warning but the application will not stop the parameter file from being overwritten).

Now click on the "Generate" green button and, if all goes well, you should receive a message telling you that the output file was created.

Use your preferred XML viewer (e.g. MS Edge) to open the newly created XML file (e.g. "EmployeeDataOut.XML"). You should see that the original data has now been put back into the XML file.

So far we have not seen CSV2DCC do anything useful. All we have done is reproduce the original file, not very useful.

But now we can edit the substitution file (EmployeeSubs.csv), by putting new data into this file. So edit the data in EmployeeSubs.CSV, make changes to the contents of Column B which contains the data and save this as EmployeeSubs2.CSV. Rerun CSV2DCC using this second version of the substitution file and this time the output will use the new information in the CSV file. Again open the output file to confirm your data has been added to the output file.

**How can CSV2DCC be used in practice?**

To actually use CSV2DCC in practice:

1. Develop a relevant DCC for your area of work (which will probably contain example data).

2. When you have an individual DCC file that is suitable, create a Parameterised version of this xml file replacing all the example data with codes (as described in Step 1 ).

3. In your current spreadsheet file that generates your results (assuming you use a spreadsheet application in your current process) add a new sheet, call it "CSVOutput" for example, and add the codes that we used in Step 2 into Column A. In Column B of the CSVOutput sheet, you will need to develop the Excel to create the relevant data. Hopefully this will just be a matter of copying information from other sheets in your spreadsheet, but this is very implementation dependent.

4. 'Run' your process spreadsheet to ensure that the correct data is put into Column B of the CSVOutput sheet. Confirm that this has worked.

5. Run the Excel export wizard to output the CSVOutput sheet to create a CSV (actually semicolon separated) file.

6. Run CSV2DCC using the above parameterised DCC file and the CSV file to create an output DCC.

**I do not use a Spreadsheet, can I still use CSV2DCC?**

Although many metrology practitioners use a spreadsheet to process raw data to create the final results, not everyone does.

You may instead use custom software to generate your results. If this is the case, in principle it is possible to add functionality to a custom implementation to output a CSV file. Please contact your local software engineer or provider to discuss this option. This document should provide the developer with enough information to implement the necessary additional functionality to output a CSV file containing the relevant information.

**Appendix A - How to install CSV2DCC [NOT FINISHED YET]**

CSV2DCC will work on Windows and Linux PCs that are running the Java Runtime Environment (JRE) version 1.8 and later. To work out which version is running on your computer please refer to "Appendix C - Java Installation".

**Appendix B - How to make Excel output Semi Colons instead of Commas**

**Note:- This will change the way numbers are presented in the spreadsheet cells.**

1. Within Excel navigate to the "Excel Options" screen
2. Click on "Advanced"
3. Under "Editing Options"
   1. Uncheck the "Use System Separators" check box.
   2. In the "Decimal separator" text box put a "," (i.e. a single comma)
   3. In the "Thousands separator" text box put a "." (i.e. a single full-stop)

Now when you output to a csv file, semi-colon separators will be used.

**Appendix C - Java Installation**

**On Windows**

Open a Command window and type

java -version

**On Linux**

Open a terminal window and type

java -version

For both Windows and Linux, you should receive a message that contains the version of Java running on your machine, this needs to be 1.8 or higher.

If you receive a message that "Java" cannot be found, then it is unlikely there is a JRE running on your machine. Please refer to your local software support team to request JRE version 1.8 (as a minimum) be installed on your computer.