The Model Conversion Definition File

John Undrill

12 September 2011

1. General

The 3ic and 4ia data conversion programs get the definitions of the conversions to be made from Model Conversion Definition files (MCDF). 3ic and 4ia search for these files in the directory *lib*. Each MCDF defines a single conversion from one PSLF dynamic model to another. Only the MCDFs that are specified in the *rule.txt* file in the working directory are used in an execution of 3ic or 4ia.

A sample MCDF is shown in figure 1.

2. The rule.txt file and lib Directory

The MCDF files must reside in the directory *lib*. This directory must be a sub-directory of of the working directory in which 3ic and 4ia are executed.

The model conversions (and corresponding MCDFs that are to be used) are specified by entries in the file *rule.txt*. This file must be in the working directory. Each line of *rule.txt* activates one conversion.
A # character in the first character of a line in *rule.txt* makes the line a comment; comment lines are ignored by 3ic and 4ia. A representative file *rule.txt* is shown in figure 2.

Each line (other than comments) of *rule.txt* must have the form

imod omod

where *imod* is the name of a PSLF dynamic model that is to be replaced and *omod* is the name of the model that is to replace it.

An MCDF must exist for each active entry in rule-text and must have the name

imod-omod

3. Model Parameter Names

The parameter names used by the PSLF dynamic models are used in the MCDF. The parameter names used in the MCDF must be identical to the names used in the PSLF documentation pages of the input and output models, with the addition of a leading @ or $ character is follows:

Parameters in the input model data record must be specified in the MCDF in the form
@name-in
where *name-in* is the name assigned to the parameter in its PSLF model documentation page.

Parameters in the output model data record must be specified in the MCDF in the form
@name-out
where *name-out* is the name assigned to the parameter in its PSLF model documentation page.

4. File Structure

4.1 Formal Structure

The structure of the Model Conversion Definition File is as follows; it consists of two stanzas:

An 'input' stanza describing the parameter sequence of the input model and assigning names to the parameters. The entires in this stanza MUST be in the order in which the parameters appear in the data record of the input model.

An 'output' stanza that specifies multiple functions:

- description of the parameter sequence of the output model
- assignment of values to the output parameters in terms of the values of the input parameters
- checking of the values assigned to the output parameters
- replacement of invalid output parameter values with 'safe' values

The entries in the output stanza MUST be in the order in which the parameters appear in the data record of the output model.

The formal arrangement of the file is as follows:

```
# comment    any line beginning with # is a comment and is skipped
# first stanza describes the parameter sequence of the input model
namea;   defaulta
nameb;   defaultb
. . . . . .
namen;   defaultn
@end
# second stanza describes the parameter sequence of the output model, assigns, and checks values
pexpa;    mina maxa   cexpa;    rexpa;
pexpb;    minb maxb   cexpa;    rexpa;
. . . . . .
pexpm;    minm maxm   cexpm;    rexpm;
$end
```

In this file :

namea,...,namen	are the names of the n parameters of the input model, with @ as the first character of each
defaulta,...,default	are default values assigned to the n parameters of the input model
mina,...,minm	are min values assigned to the m parameters of the output model
maxa,...,maxm	are max values assigned to the m parameters of the output model
pexpa,...,pexpm	are parameter assignment statements of the form *name-out = f(name-in)* stating the values of the output parameters in terms of the input parameters
cexpa,...,cexpm	are logical expressions of the form *f(name-in, name-out)* defining checks of the output parameter values resulting from the corresponding *pexpa,...,pexpm* expressions
rexpa,...,rexpm	are replacement assignment statements of the form *name-out = f(name-in,name-out)* stating the value of the output parameters in terms of the input parameters

All elements of the file must be present except for the replacement assignment statement, *rexpa,...,rexpm.* Any or all of *rexpa,...,rexpm* may be omitted.

Note that the *default, min,* and *max* entires in the file are for information only and are not used by 3ic.

4.2 Example File and Example Expressions

Figure 1 shows a sample MCDF specifying a conversion of the **exdc1** model into the corresponding **esdc1** model.

The simplest parameter assignment expression used in figure 1 is

```
$tr = @tr
```

The simple assignment expression states that the first parameter of the input **exdc1** model is to be copied directly to the first parameter of the output **esdc1a** model.

The checking statement associated with this assignment statement is

```
$tr>=0
```

It states that the value of *$tr* must be zero or positive to be correct.

There is no replacement statement following the above checking statement. In the absence of the replacement 3ic flags the output value of *$tr* if the checking statement is not TRUE, but takes no corrective action.

The line

```
$tf=@tf1         0.1   5.0       $tf>=0.0
```

in the MCDF illustrates the assignment of the value to an output parameters whose name is different from that of the corresponding input parameter. Note that the checking statement, $tf>=0, examines the output parameter.

The line

```
$tc=@tc;         0.0   5.0    ($tc>=0)&&($tc<=$tb);       $tc=1;
```

specifies that the parameter *tc* of the exst1 model is to be copied without change to the parameter *tc* of the esst1a model. The output value of *tc* is checked against zero and against the output value of the parameter *tb*. The replacement statement `$tc=1;` specifies that *tc* must be set to zero if the check statement does not evaluate to TRUE.

5. Sequence of Operation of 3ic

3ic operates in the following sequence:

the data record of the input model is parsed and the parameter values are stored as the *name-in* variables assigned in the first stanza of the MCDF

values are assigned to the *name-out* variables in accordance with the parameter assignments in the second stanza of the MCDF

the checking logical expressions are evaluated and, if a logical expression evaluates to FALSE the corresponding replacement statement (if present) is executed immediately

the data record for the output model is written to the output dyd file

6. Variables

The statements and logical expressions specified in the MCDF are parsed, interpreted, and executed with the same logical and arithmetic operators and with the same precedence rules as used in the C language. The names, *name-in* and *name-out*, are the names of the variables that are available to these 'programs'. Each 'program' specified in the second stanza has access to all variables specified in the first. It is important to note, however, that the parameter assignment statements in the second stanza can use ONLY *name-out* variables that have been assigned by a preceding statement.

Because all assignments specified in the second stanza are made before the checking statements are executed, the checking and replacement statements can refer to all named parameters, both *name-in* and *name-out*.

7. Expressions and Operators

Statements and logical expressions can use the following operators:

+	plus	
-	minus	
*	multiply	
/	divide	
>	great than	(logical)
<	less than	
>=	greater than or equal	(logical)
<=	less than or equal (logical)	
&&	and	(logical)
||		(logical)
(	open parenthesis	
)	close parenthesis	

Statements and expressions must contain NO spaces. They may be terminated by a semicolon, **;**, as an aid to clarity.

Figure 1 Sample Model Conversion Definition File

```
#exdc1  esdc1a
@tr      0.0
@ka    400.0
@ta      0.02
@tb      0.0
@tc      0.0
@vrmax  20.0
@vrmin -20.0
@ke      1.0
@te      1.0
@kf      0.02
@tf1     0.0
@tf2     0.0
@e1      3.0
@se1     0.01
@e2      4.0
@se2     0.1
@end
#
#
#Assignment     Min  Max   Check                                          Replacement
#statement      val  val    statement                                      statement
#
$tr=@tr;        0.0  0.5   $tr>=0
$ka=@ka         5.0  1000  $ka>=1
$ta=@ta         0.0  0.5   $ta>=0.0
$tb=@tb;        0.0 20.0   (($tb==0)&&($tc==0))||(($tb>=@tc)&&($tb>0));     $tb=1;
$tc=@tc;        0.0  5.0   ($tc>=0)&&($tc<=$tb);                            $tc=1;
$vrmax=@vrmax   1.0 40.0   $vrmax>=1
$vrmin=@vrmin -40.0  0.0   $vrmin<=0
$ke=@ke         0.0  2.0   $ke>=-0.2
$te=@te         0.05 2.5   $te>=0.05
$kf=@kf         0.0  0.5   $kf>=0
$tf=@tf1        0.1  5.0   $tf>=0.0
$spdmlt=0       0.0  1.0   $spdmlt==0
$e1=@e1         1.0  5.0   $e1>=1
$se1=@se1       0.0  0.2   $se1>=0
$e2=@e2         1.2  6.0   $e2>=0
$se2=@se2       0.0  1.0   $se2>=0;
$uelin=0        0.0  2.0   $uelin==0;
$exclim=0       0.0  0.0   $exclim==0;
```

Figure 2 Sample file rule.txt

```
#Input              Output
#Model              Model
exdc4               esdc3a
exst3               esst3a
exst3a              esst3a
exst2               esst2a
exst2a              esst2a
exac4               esac4a
exac6a              esac6a
scrx                esst1a
#sexs               esst1a
exst1               esst1a
exst4b              esst4b
ieeet1              esdc1a
exdc1               esdc1a
exdc2               rexs
exdc2a              esdc2a
exac1               esac1a
exac1a              rexs
exac2               esac2a
exac3a              esac3a
exac8b              esac8b
#pidgov              hypid
#ieeeg3              hypid
#gpwscc              hypid
#hygov               hypid
#lcfb1               lcfb1
hygovr              hygovr
```