Chemistry Preprocessor

Introduction

The chemistry preprocessor is a tool that generates CAM Fortran source code that numerically solve a set of differential equations which represent the chemical reactions. Solving this set of differential equations provides the temporal evolution of the chemical tracers.

The chemistry preprocessor generates code that is numerically efficient (with hard-wired values) and provides provides flexibility when changing the chemical mechanism. The input is an ascii file that uses a straightforward syntax to describe the chemistry of a simulation. The input file defines the chemical species, photolysis and gas phase reactions and rate constants, partitions species by numerical solution algorithm, and specifies which species have heterogeneous removal by washout, external forcing, and which species are not transported.

Upon successful completion the preprocessor produces all the chemistry source files required to build a CAM-Chem executable that represents the simulation defined in the preprocessor input file. Additionally, the preprocessor outputs a “document” file containing a summary of the processed input file. If the preprocessor detects a syntax or logic error the diagnostics relating to the error can be found at the end of the “document” file. The preprocessor can alter the simulation chemistry from the narrow perspective of a single reaction rate to the broad scale of entirely replacing the “standard” chemistry sets.

Completely replacing the "standard chemistry", although rather easy to do with the preprocessor, has far ranging code consequences. Several source code files in the “base” source directories use fortran modules produced by the preprocessor.

The MOZART chemistry preprocessor has been adopted for use in CAM-Chem.

Invocation

The chemistry preprocessor can be invoked via the CAM configure utility or as a stand-alone executable. Invoking the preprocessor from CAM configure makes it relatively easy to customize the chemistry. Executing the preprocessor as a stand-alone tool allows the user to test input files outside of the CAM cofigure.

CAM configure

The user has the ability to customize the CAM-Chem chemistry mechanism when CAM is configured. The chemistry preprocessor can be engaged with CAM configure options:

```
setenv CAMCHEM_EDITOR emacs

$biddir/configure
  -usr_mech_infile $mechanism_filepath
  -edit_chem_mech

where

  $mechanism_filepath is the full path name of the preprocessor input file
```

The "-edit_chem_mech" option uses the editor specified by the $CAMCHEM_EDITOR environment variable if set, otherwise vi. This typically would be used when small changes to the mechanism are desired. The input file for "CAM configure" mode of invocation is the same as the stand-alone mode without the BEGSIM, ENDSIM, and the preprocessor files entries.

When the chemistry is customized via these options, CAM configure will invoke the preprocessor and include the newly generated fortran source code in the compile search path.

Stand-alone

Preprocessor Source:

The location of the preprocessor in the CAM source distribution is

`$CCSMROOT/models/atm/cam/chem_proc/`

Executable Build

At the unix prompt enter the follow commmands :

```
prompt> cd $CCSMROOT/models/atm/cam/chem_proc/src
prompt> gmake
```

This will create an executable named "campp" in the directory `$CCSMROOT/models/atm/cam/chem_proc/bin`.

Run

At the unix prompt enter :

```
prompt> path/campp path/input_file
```

This will produce output files as specified in the input_file "Preprocessor Files" section. The necessary fortran source code files, contained in a .tar file. This will also produce a human readable ascii .doc file which is a description of the chemistry mechanism.

There are many sample input files in `$CCSMROOT/models/atm/cam/chem_proc/inputs`.

Input File

Input file

```
BEGSIM

  Preprocessor Specs

  Comments
  End Comments

  SPECIES
  End SPECIES

  Solution Classes
  End Solution Classes

  CHEMISTRY
  END CHEMISTRY

  SIMULATION PARAMETERS
  END SIMULATION PARAMETERS

ENDSIM
```

A mozart preprocessor input file, a simple ascii file, has the above general structure. Note how section keyword pairs are of the form :

```
    keyword
    end keyword
```

except for the BEGSIM, ENDSIM pair and the “Preprocessor Files” entries. The comments and chemistry sections are optional; all others are mandatory. All input between the BEGSIM, ENDSIM pair, if present, must be order as above. What’s this, the chemistry section is optional ? Yes, you can setup a two species tracer simulation with no chemical reactions.

It’s hard to discern from this skeleton example but the mozart preprocessor is in general case insensitive. For instance, in the “general structure” above you could have the SPECIES line be input as SpEcieS and the preprocessor would still interpret this as the species keyword. Furthermore, in general, white space is ignored. In “general structure” above there are several blank lines. The preprocessor always ignores blank lines. Just as you could input the SPECIES line as SpEcieS, you could also input this line as “S pE cie S” and it would be a valid, although less readable.

This is a good point to bring up the basic syntax rules of the mozart preprocessor.

1. Input lines are limited to 120 characters.
2. White space at the beginning of a line counts in the character limit.
3. Input lines greater than 120 characters are truncated and can cause preprocessing errors.
4. Input lines are rather free form; they may start in any column.
5. Blank lines are ignored
6. Lines in which the first character is an "*" are considered comment lines and are ignored.

7. The valid character set for the mozart preprocessor is the alpha numeric set, letters and numbers, and the following characters: `- + * = [ ] < > . : ,` (Note: the “*” can be used as either a comment indicator or as part of a numeric expression such as `2*N2O5`.)
8. In general white space within an input line is ignored. The important exceptions to this rule are the entries in the “Species” section and any subsequent specifications dependent on entries in the species section. Of course, sensible white space use makes preprocessor input files more readable.

“Preprocessor Specs” represents a special set of preprocessor specifications that are different in form than all others. These specify the directory in which the output files are generated, the mechanism document file name, and location of some needed preprocessor code files.

The following will go through each of the above remaining listed sections in greater detail.

Comments (optional)

```
Comments
"This is a mozart4 simulation with:"
"(1) The Lin and Rood advection routine"
"(2) mozart inputs"
"(3) New isoprene chemistry"
" added species: CH3OH, C2H5OH, GLYALD, HYAC, EO2, EO, HYDRALD"
End Comments
```

The comments section is a good, simple starting place.

Purpose

Allows the user to add notes to the preprocessing input file. These notes are also output to the document file.

Syntax and limits

1. Limit of 100 comment lines. Go over this limit and the preprocessor error halts.
2. The character set for comments is limited only by what you can type in.
3. As you can see from above enclosing lines in the comment section within double quotes, "...", perserves the input line intact; white space and all.
   If we had entered the line :
   `"(2) mozart inputs"`
   as
   `(2) mozart inputs`
   then this line in the document file would look like :
   `(2)mozartinputs`

Species (mandatory)

```
Species
   Solution
   End Solution

   Fixed
   End Fixed

   Not-Transported
   End Not-Transported

   Col-int
   End Col-int
End SPECIES
```

This is the general form of the species section. There are 4 sub-sections: `solution`, `fixed`, `not-transported`, and `col-int` and they must be entered in that order. Only the `not-transported` and `col-int` sub-sections are optional.

The Solution sub-section (mandatory)

```
Solution
   O3, O, O1D -> O, N2O, N, NO, NO2, NO3, HNO3, HO2NO2, N2O5, CH4, CH3O2
   CH3OOH, CH2O, CO, OH, HO2, H2O2, C3H6, ISOP -> C5H8, PO2 -> C3H6OHO2
   CH3CHO, POOH->C3H6OHOOH, CH3CO3, CH3COOOH, PAN -> CH3CO3NO2
   ONIT -> CH3COCH2ONO2, C2H6, C2H4, C4H10, MPAN->CH2CCH3CO3NO2
   ISOPO2 -> HOCH2COOCH3CHCH2, MVK-> CH2CHCOCH3, MACR->CH2CCH3CHO
   MACRO2->CH2CCH3CO3, MACROOH->CH3COCHOOHCH2OH
   MCO3 -> CH2CCH3CO2, C2H5O2, C2H5OOH, C10H16, BIGALK, BIGENE, C3H8
   C3H7O2, C3H7OOH, CH3COCH3, ROOH -> CH3COCH2OOH, CH3OH, C2H5OH
   GLYALD -> HOCH2CHO, HYAC -> CH3COCH2OH, EO2->HOCH2CH2O2
   RO2-> CH3COCH2O2, ISOPNO3 -> CH2CHCCH3OOCH2ONO2, H2, O3S -> O3
   O3INERT -> O3, O3RAD->O3
End Solution
```

Purpose

Define the solution species. As a part of this input the molecular weight of each solution species is also defined.

Syntax and limits

1. 300 solution species limit per simulation
2. Solution species are limited to eight alphanumeric characters
3. Solution species are case sensitive
4. Solution species "aliases" are limited to 32 alphanumeric characters
5. Species and “species->alias” may not be broken across lines

The mozart model employs both mass and volumetric mixing ratio units internally. Specifically, only the chemistry modules utilize volumetric mixing ratio. Thus mozart has to convert to and from volumetric mixing ratios. This requires the molecular weight of each solution species. One way or another all solution species need to have their symbolic name relate to their chemical composition as represented by the periodic table. For instance, in the above input ozone is specified as `O3` not `o3` since the symbol in the periodic table `O` represents atomic oxygen whereas `o` is not represented. Numbers following valid periodic table elements act exactly as you would expect – they indicate quantity. Invalid periodic elements are ignored in the molecular weight specification; they do not cause a preprocessor error halt.

Note the aliasing mechanism in the solution species sub-section. For example, the entry `PAN -> CH3CO3NO2`. Here the solution species labeled `PAN` actually represents the compound `CH3CO3NO2`. Why is this necessary ? Remember: the preprocessor allows only eight characters for the each solution species. What if I forget to “alias” the `PAN` species ? In that case the combined molecular weight of phosphorus and atomic nitrogen will be assigned to `PAN (there is no element represent by A and thus it is ignored). And note that the excited oxygen atom, {{O1D`, is aliased to atomic oxygen, `O`. Solution species aliasing is for assigning descriptive names and masses for species with long compounds or those such as `O1D` that don’t properly map into the periodic table. It does not imply that the solution species has chemical behavior similar to the alias. That is determined by the chemical reactions in the “chemistry” section.

This is the most critical section of all. A subtle typo here can be hard to track down later. One of the most common errors is to misrepresent the chemical formulation of a compound. In that case the simulation molecular mass will be erroneous and you may get simulation volumetric mixing ratios in your outputs that don't correspond to expected values. Note that `CO` and `Co` are not the same species; the first is carbon monoxide and the second is cobalt. While `HNO4` and `HO2NO2` have the same molecular weight they are two separate species to the preprocessor. As indicated above you can’t assign ozone the symbol “o3”. However, you could use aliasing to define ozone as :
`o3 -> O3`
Although perfectly valid this construct is not advised. Why not ? Some mozart chemistry subroutines use hardwired species symbols that are consistent with a direct mapping to the periodic table. Thus some chemistry routines will look for the species symbol “`O3`” but not “`o3`”.

The Fixed sub-section (mandatory)

```
   Fixed
     M, N2, O2, H2O
   End Fixed
```

Purpose

Specify the "fixed" species. Fixed species participate in chemical reactions but their values are assigned not chemically derived. Separate routines exist to specify their values at each time step during a simulation.

Syntax and limits

1. 300 fixed species limit per simulation
2. Fixed species are limited to eight alphanumeric characters
3. The symbol "`M`" represents total atmospheric density and must appear in the Fixed subsection

The example fixed sub-section is fairly typical with molecular nitrogen, molecular oxygen, and water vapor declared as fixed species as well as the mandatory total atmospheric density. Note that unlike the solution species fixed species do not have any periodic table matching requirements and their molecular weight is not computed.

The Not-Transported *sub-section (optional)

```
    Not-transported
      OH, ...
    End Not-transported
```

Purpose

The not-transported sub-section, defines those solution species for which are not transported in CAM.

Syntax and limits

1. 300 not-transported entries per simulation
2. All not-transported entries must be solution species; they must be entered exactly as specified in the solution species sub-section

The not-transported species are species that have a short chemical life-time relative to the model time step size, e.g., `OH`.

The* Col-int sub-section (optional)

```
    Col-int
      O3
      O2
    End Col-int
```

Purpose

The col-int sub-section, short for column integral, defines those solution species for which a vertical column integral is required.

Syntax and limits

1. 300 col-int entries per simulation
2. All col-int entries must be either solution or fixed species; they must be entered exactly as specified in the solution or fixed species sub-section

The example col-int sub-section specifies that `O3` and `O2` will have column integrals formed in the mozart simulation. If no col-int species are defined then mozart standard code in the photolysis routine will not call any column integral routines. The entries in the col-int example are placed one per line for clarity. They could have been specified in one line via :
`O3, O2`

The base photolysis routines in mozart require column integrals for `O3` and `O2`. A simulation with no photorates normally would not require a col-int section.

Solution Classes (mandatory)

```
Solution Classes
   Explicit
   End Explicit

   Implicit
   End Implicit

   Rodas
   End Rodas
End Solution Classes
```

This is the general form of the solution classes section. There are 3 sub-sections: explicit, implicit, and rodas. Solution classes order is immaterial. Each solution species must be uniquely mapped to one of the three classes. Solution classes partition solution species into distinct chemical numerical solvers. The list entry “All” may be used for any single solution class to place all the species in that class.

The Explicit sub-section (optional)

```
Explicit
   CH4, N2O, C
End Explicit
```

Purpose

List of species to be solved via the forward Euler or explicit algorithm. Be careful with this solution class. Note that all the species listed in this class are presumed to have rather gentle chemistry with longer lifetimes. It would be disastrous to place a highly reactive species such as `OH` in the explicit solution class.

Syntax and limits

1. 300 species limit per simulation
2. All explicit class members must be solution species; they must be entered exactly as specified in the solution or groups sub-section of the species section

The Implicit sub-section (optional)

```
Implicit
  O3, N, NO, NO2, NO3, HNO3, HO2NO2, N2O5, CH3O2
  CH3OOH, CH2O, OH, HO2, H2O2, C3H6, ISOP, PO2, CH3CHO
  POOH, CH3CO3, CH3COOOH, PAN, ONIT, C2H6, C2H4, C4H10, MPAN
  ISOPO2, MVK, MACR, MACRO2, MACROOH
  MCO3, C2H5O2, C2H5OOH, C10H16
  C3H8, C3H7O2, C3H7OOH, CH3COCH3, ROOH
  CH3OH, C2H5OH, GLYALD, HYAC, EO2, RO2, ISOPNO3, O3RAD
End Implicit
```

Purpose

Specify all species to be solved via the backward Euler or implicit algorithm. This is the "work horse" solution class. If in doubt put a species in this class.

Syntax and limits

The limits and syntax rules for the implicit class as the same as those for the explicit class. There is no harm in placing all of the solution species in the implicit class. This is a good place to use the “all” list option as in :

```
Implicit
  All
End Implicit
```

The Rodas sub-section (optional)

```
Rodas
End Rodas
```

Purpose

List all species to be solved via the implicit Rosenbrock solver Rodas. This is the "cadillac " solution class. Again as with the implicit solver class you may put anything in this class. This class is about twice as expensive as the implicit class and should only be considered for situations where the implicit solver exhibits repeated converge failure. If you use this class be careful with the per species relative error criterion. Setting too strigent a criteria, generally < 1.e-3, can cause the computational time to increase greatly.

Syntax and limits

The limits and syntax rules for the rodas class as the same as those for the explicit class. The example preprocessor file does not use the rodas solver for any species. The mozart model has completed hundreds of simulation years with scientifically acceptable results without ever using the rodas solver.

Chemistry (optional)

```
CHEMISTRY
   Photolysis
   End Photolysis

   Reactions
   End Reactions

   Heterogeneous
   End Heterogeneous

   Ext Forcing
   End Ext Forcing
END CHEMISTRY
```

Above is the general form of the chemistry section. There are 4 sub-sections: photolysis, reactions, heterogeneous, and ext forcing. Interestingly, none of the sub-sections are mandatory. If they exist chemistry sub-sections must follow the ordering indicated above.

The following details general characteristics of both photo and gas phase reactions (Photolysis and Reactions sections).

Reactants and products are limited to eight characters. Reactants are restricted to be either solution species or the fixed species. Any reactant not in the solution or fixed lists will cause a preprocessor error halt. Any eight character alphanumeric string is allowed for products. Products that are neither solution or fixed species are flagged in the document file. They are enclosed in the {} pair as in {XYX}, assuming XYX is not a solution or fixed species.

Reactants may not have an explict coefficient; they always have implied unity coefficients. Reactants are separated from each other by the "+" operator. Products are separated from each other by either the "+" or "-" operator. It seems strange but you may in essence have negative products. Blame this on complex hydrocarbon chemistry. Products may have any valid fortran number as a coefficient. Again the default coefficient is unity. A coefficient, species pair is separated by the "*" operator as in "2.5*OH". The coefficients are checked for validity and can cause an error halt. Reactions that do not fit on one input line may be continued on subsequent lines. The first non-blank character of the continued lines must be a product separator; either "+" or "-". A reaction line must not end with a product separator. All reaction lines must have whole product and reactant symbols; breaking a reaction line in the middle of a reactant or product will cause a preprocessing error halt. Multi-line reactions must have at least one product on the first line.

Reactants are separated from products with either the "`->`" or "=" operator. Either is allowed in a given reaction. Specifics regarding reactant limits and gas phase reaction rate constants will be covered in the photolysis and reactions sub-sections below. All reactions may have up to 16 products. Although a photo or gas phase reaction may have no products the reaction must have the "`->`" or "=" reactant, product delimiter. Reactions involving only fixed species, such as `O2 + hv ->` …, must have a product even if it is a dummy symbol such as "`NULL`" or "`NONE`" (assuming that no solution for fixed species is assigned "`NULL`" or "`NONE`"). A reaction with no valid solution or fixed species will cause a preprocessor error halt.

The "tags" at the beginning of a reaction line, enclosed by the `[]` pair, are useful for tagging reactions. These tags can be used to identify individual reactions in the reaction matrix. Thus you can potentially change the order of reactions, add or delete reactions and not have to worry about the mapping of a given reaction to a location in the fortran reaction array. Tags, required for photorates and optional for gas phase reactions, are limited to 16 characters not including the enclosing `[]` pair.

The combined photolysis and gas phase reaction count is limited to 900.

The <u>Photolysis</u> sub-section (optional)

```
   Photolysis

     [jo2->,jo2_b]          O2 + hv -> 2*O
     [jo1d->,jo3_a]         O3 + hv -> O1D + O2
     [jo3p->,jo3_b]         O3 + hv -> O + O2
     [jn2o]                 N2O + hv -> O1D + N2
     [jno2]                 NO2 + hv -> NO + O
     [jn2o5->,jn2o5_a]      N2O5 + hv -> NO2 + NO3
     [jhno3]                HNO3 + hv -> NO2 + OH

     [jbigald->,.2*jno2]    BIGALD + hv -> .45*CO + .13*GLYOXAL + .56*HO2 + .13*CH3CO3 + .18*CH3COCHO
     [jch3ooh]              CH3OOH + hv -> CH2O + HO2 + OH
     [jch2o_a]              CH2O + hv -> CO + 2 * HO2
     [jch2o_b]              CH2O + hv -> CO + H2
     [jh2o2]                H2O2 + hv -> 2*OH
     [jch3cho]              CH3CHO + hv -> CH3O2 + CO + HO2
     [jpooh->,jch3ooh]      POOH + hv -> CH3CHO + CH2O + HO2 + OH

*------------------------------------------------------------------------------
*   photo-ionization
*------------------------------------------------------------------------------
     [jeuv_1=userdefined,userdefined]    O + hv -> Op + e
     [jeuv_2=userdefined,userdefined]    O + hv -> Op + e
     [jeuv_3=userdefined,userdefined]    O + hv -> Op + e

   End Photolysis
```

Purpose

Specify the photolysis reactions. Photolysis reactions have one and only one true reactant. They must have the "hv" symbol as a symbolic second “place holder” reactant.

Syntax and limits

1. Up to 900 photo and gas phase reactions per simulation
2. Reactants must be solution or fixed species
3. Reactants understood to have unity coefficients; no reactant coefficients allowed
4. 16 products allowed
5. "hv" symbol required as second reactant
6. No explicit rate constant may be assigned (see rate constant information in the reactions section below)
7. aliases can be assigned with the "->" or "=" symbol followed by long wave and short wave aliases which are separated by a comma
8. an alias can be "userdefined" which requires special code in the photolysis routine
9. an alias can include a numeric multiplication factor

Alias Tagging

[alias_tag{=,->short_coeff*short_alias_tag,long_coeff*long_alias_tag}]

- the {,} pair signify optional entries and are not part of the photorate aliasing syntax
- Only the enclosing [,] pair and the alias_tag are required. In this case the photorate alias is acting as "pure" tag as in : [jh2o2].
- If the photrate tag is to infer an aliasing then either the the "=" or "->" string must separate the alias_tag from the short, long alias tags.
- The short,long coefficients, if present, must be valid fortran90 numbers; integers are fine as in 2*.
- The short,long aliases must refer to an existing photorate alias_tag that precedes the alias_tag in the preprocessor input file (either or both short, long aliases may be set to the alias_tag as in : [jno2 = 1.23*jno2]).
- If only short aliasing is desired the "," separating the short, long aliases must NOT be present as in : [jn2o5->jno3]
- If only long aliasing is desired the "," separating the short, long aliases must be the first character after the the {=,->} delimiter as in : [jn2o5=,2.5*jhno3]

The Reactions sub-section (optional)

```
   Reactions
     [usr1]   O + O2 + M -> O3 + M
              O + O3 -> 2*O2 ; 8e-12, -2060
     [o1d_n2] O1D + N2 -> O + N2 ; 1.8e-11, 110
     [o1d_o2] O1D + O2 -> O + O2 ; 3.2e-11, 70
     [o3_l1]  O1D + H2O -> 2*OH ; 2.2e-10
              N2O + O1D -> 2*NO ; 6.7e-11
              N2O + O1D -> N2 + O2 ; 4.9e-11
     [o3_p1]  NO + HO2 -> NO2 + OH ; 3.5e-12, 250
              NO + O3 -> NO2 + O2 ; 3e-12, -1500
              NO2 + O -> NO + O2 ; 5.6e-12, 180
              NO2 + O3 -> NO3 + O2 ; 1.2e-13, -2450
              NO3 + HO2 -> OH + NO2 ; 2.3e-12, 170.
     [usr2]   NO2 + NO3 + M -> N2O5 + M ; 2.e-30,4.4, 1.4e-12,.7, .6
     [usr3]   N2O5 + M -> NO2 + NO3 + M
              N2O5 + H2O -> 2*HNO3 ; 0.
     [usr4]   NO2 + OH + M -> HNO3 + M ; 2.4e-30,3.1, 1.7e-11,2.1, .6
     [usr5]   HNO3 + OH -> NO3 + H2O
              NO3 + NO -> 2*NO2 ; 1.5e-11, 170
     [usr6]   NO2 + HO2 + M -> HO2NO2 + M ; 1.8e-31,3.2, 4.7e-12,1.4, .6
              HO2NO2 + OH -> H2O + NO2 + O2 ; 1.3e-12, 380
     [usr7]   HO2NO2 + M -> HO2 + NO2 + M
              CH4 + OH -> CH3O2 + H2O ; 2.45e-12, -1775
              CH4 + O1D -> .75*CH3O2 + .75*OH + .25*CH2O + .4*HO2 + .05*H2 ; 1.5e-10
     [o3_p2]  CH3O2 + NO -> CH2O + NO2 + HO2 ; 3.e-12, 280
     [usr11]  CH3CO3 + NO2 + M -> PAN + M ; 8.5e-29,6.5, 1.1e-11,1., .6
              CH3CO3 + HO2 -> .7*CH3COOOH + .3*CH3COOH + .3*O3 ; 4.3e-13, 1040
              CH3CO3 + CH3O2 -> .9*CH3O2 + CH2O + .9*HO2 + .9*CO2 + .1*CH3COOH ; 1.3e-12,640
 *-----------------------------------------------------------------------
 * note the reaction immediately below is "commented out" and will not
 * be in the reaction mechanism
 *-----------------------------------------------------------------------
 *            CH3COOOH + OH -> CH3CO3 + H2O ; 1e-12
              CH3COOOH + OH -> .5*CH3CO3 + .5*CH2O + .5*CO2 + H2O ; 1e-12
      [usr12] PAN + M -> CH3CO3 + NO2 + M
              CH3CO3 + CH3CO3 -> 2*CH3O2 + 2*CO2 ; 2.5e-12, 500
      [o3_l5] ISOP + O3 -> .4*MACR + .2*MVK + .07*C3H6 + .27*OH ; 1.05e-14, -2000
                         + .06 * HO2 + .6 * CH2O + .3 * CO + .1 * O3
                         + .2 * MCO3 + .2 * CH3COOH
              OH + C2H6 -> C2H5O2 + H2O ; 8.7e-12, -1070
      [O3_p5] C2H5O2 + NO -> CH3CHO + HO2 + NO2 ; 2.6e-12, 365
              C2H5O2 + HO2 -> C2H5OOH + O2 ; 7.5e-13, 700
              C2H5O2 + CH3O2 -> .7 * CH2O + .8 * CH3CHO + HO2 ; 2.e-13
                              + .3 * CH3OH + .2 * C2H5OH
              C2H5O2 + C2H5O2 -> 1.6*CH3CHO + 1.2*HO2 + .4*C2H5OH ; 6.8e-14
              C2H5OOH + OH -> .5*C2H5O2 + .5*CH3CHO + .5*OH ; 3.8e-12, 200
   End Reactions
```

Purpose

Specify all gas phase reactions. Gas phase reactions must have at least one reactant and may have up to three reactants. All reactants must be either solution or fixed species. If a reaction has three reactants then at most two can be solution species and at least one must be a fixed species. For example, if OH and HO2 are solution species, the following is an invalid gas phase reaction :

```
OH + OH + HO2 -> H2O2
```

Gas phase reactions may additionally define rate constants. Rate constants are delimited from reaction products by the ";" character. There are two types of rate constants; arrenhius and troe.

The general Arrenhius rate constant is of the form:

```
a0 * exp( b0/t )
```

where `a0` and `b0` are constants to be specified and t is temperature(K).

An example is:

`C2H5O2 + HO2 -> C2H5OOH + O2 ; 7.5e-13, 700`

where `a0 = 7.5e-13` and `b0 = 700`; the rate constant is `7.5e-13*exp( 700/t )`.

Rate constants `a0` and `b0` are checked for fortran numeric validity. They may be positive or negative.

NOTE: The JPL book provides A and (E/R) in `k(T) = A*exp ((-E/R) (1/T))`, whereas here `b0 = (-E/R)`.

A temperature independent arrenhius rate only has the `a0` term as in :

`CH3COOOH + OH -> .5*CH3CO3 + .5*CH2O + .5*CO2 + H2O ; 1e-12`

where:

`a0 = 1e-12`

Each arrenhius rate constant parameter {`a0, b0`} is limited to 16 characters. Parameters are delimited by the "," character.

The general troe rate constant is of the form :

```
  alpha**x/(1+beta**2)
where:
  alpha = k0*M/kinf
  beta  = log10( alpha )
  M     = total atmospheric density (molecules/cm**3)
  x     = "exponential factor"
  k0    = a0*(300/t)**a1
  kinf  = b0*(300/t)**b1
  t     = temperature (degrees Kelvin)
```

a0, a1, x, b0, b1 are rate constant inputs to be specified in that order as in :

`[usr11] CH3CO3 + NO2 + M -> PAN + M ; 8.5e-29, 6.5, 1.1e-11, 1., .6`

Here `a0 = 8.5e-29`, `a1 = 6.5`, `b0 = 1.1e-11`, `b1 = 1.`, and `x = .6`

Each troe rate constant parameter {`a0, a1, x, b0, b1`} is limited to 16 characters. Parameters are delimited by the "," character.

Whether arrenhius or troe rates, the reaction rate constants must be input on the first line of a multi-line reaction.

Gas phase reactions with no specified rate constant are labeled as "user defined" in the document file and their rate constant must be supplied in a user supplied subroutine(mo_usrrxt.F90). Failure to supply a rate constant for such a reaction can lead to a bogus simulation. At best the simulation will rapidly break down with some sort of runtime exception. At worst the simulation will complete without incident, however the results will be erroneous.

Syntax and limits

1. Up to 900 photolysis and gas phase reactions limit per simulation
2. Reactants must be solution or fixed species
3. Limit of two solution species reactants per reaction
4. Reactants understood to have unity coefficients; no reactant coefficients allowed
5. 16 products allowed

The Heterogeneous sub-section (optional)

```
   Heterogeneous
     H2O2, HNO3, CH2O, CH3OOH, POOH, CH3COOOH, HO2NO2, ONIT, MVK, MACR,
     C3H7OOH, ROOH, CH3COCHO, Pb, MACROOH, XOOH, ONITR, ISOPOOH
     CH3OH, C2H5OH, GLYALD, HYAC, HYDRALD, CH3CHO, ISOPNO3
   End Heterogeneous
```

Purpose

List all solution species that are removed by wet deposition(washout).

Although this section is optional only species listed in the Heterogeneous sub-section will undergo wet removal in the simulation.

Syntax and limits

1. Up to 300 heterogeneous entries
2. Only solution species allowed
3. Each individual species must appear only once

The Ext Forcing sub-section (optional)

```
Ext Forcing
  NO <- dataset, CO <- dataset, CH4
End Ext Forcing
```

Purpose

Specify all solution species that have "external" or in situ (vertically distributed) chemical forcing and if applicable declare a solution species to use input from a dataset in forming the external forcing.

Although this section is optional only species listed in the Ext forcing sub-section will have external forcing in the simulation.

Syntax and limits

1. Up to 300 ext forcing entries
2. Only solution species allowed
3. Each individual species must appear only once

In the above example `NO` and `CO` will use values read in from a dataset while `CH4` does not.

MOZART4_preprocessor.pdf
pht_tags.txt