MODFLOW 6 Program Documentation

Release 6.2.0

MODFLOW Development Team

Oct 27, 2020

CONTENTS

Contents:

CHAPTER

ONE

MODFLOW 6 INPUT GUIDE

The latest version of the complete MODFLOW 6 input output guide can be found here.

1.1 Simulation

1.1.1 SIM-NAM

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [CONTINUE]
  [NOCHECK]
  [MEMORY_PRINT_OPTION <memory_print_option>]
  [MAXERRORS <maxerrors>]
END OPTIONS
```

```
BEGIN TIMING
  TDIS6 <tdis6>
END TIMING
```

```
BEGIN MODELS
  <mtype> <mfname> <mname>
  <mtype> <mfname> <mname>
  ...
END MODELS
```

```
BEGIN EXCHANGES
  <exgtype> <exgfile> <exgmnamea> <exgmnameb>
  <exgtype> <exgfile> <exgmnamea> <exgmnameb>
  ...
END EXCHANGES
```

```
BEGIN SOLUTIONGROUP <group_num>
  [MXITER <mxiter>]
  <slntype> <slnfname> <slnmnames(:)>
  <slntype> <slnfname> <slnmnames(:)>
  ...
END SOLUTIONGROUP
```

Explanation of Variables

Block: OPTIONS

- `CONTINUE` keyword flag to indicate that the simulation should continue even if one or more solutions do not converge.
- `NOCHECK` keyword flag to indicate that the model input check routines should not be called prior to each time step. Checks are performed by default.
- `memory_print_option` is a flag that controls printing of detailed memory manager usage to the end of the simulation list file. NONE means do not print detailed information. SUMMARY means print only the total memory for each simulation component. ALL means print information for each variable stored in the memory manager. NONE is default if MEMORY_PRINT_OPTION is not specified.
- `maxerrors` maximum number of errors that will be stored and printed.

Block: TIMING

- `tdis6` is the name of the Temporal Discretization (TDIS) Input File.

Block: MODELS

- `mtype` is the type of model to add to simulation.
- `mfname` is the file name of the model name file.
- `mname` is the user-assigned name of the model. The model name cannot exceed 16 characters and must not have blanks within the name. The model name is case insensitive; any lowercase letters are converted and stored as upper case letters.

Block: EXCHANGES

- `exgtype` is the exchange type.
- `exgfile` is the input file for the exchange.
- `exgmnamea` is the name of the first model that is part of this exchange.
- `exgmnameb` is the name of the second model that is part of this exchange.

Block: SOLUTIONGROUP

- `group_num` is the group number of the solution group. Solution groups must be numbered sequentially, starting with group number one.
- `mxiter` is the maximum number of outer iterations for this solution group. The default value is 1. If there is only one solution in the solution group, then MXITER must be 1.
- `slntype` is the type of solution. The Integrated Model Solution (IMS6) is the only supported option in this version.
- `slnfname` name of file containing solution input.
- `slnmnames` is the array of model names to add to this solution. The number of model names is determined by the number of model names the user provides on this line.

Example Input File

```
# This block is optional
BEGIN OPTIONS
END OPTIONS

# Simulation timing information
BEGIN TIMING
  TDIS6 simulation.tdis
END TIMING

# List of models in the simulation
BEGIN MODELS
  #modeltype      namefile      modelname
         GWF6    model1.nam    GWF_Model_1
         GWF6    model2.nam    GWF_Model_2
END MODELS

# List of exchanges in the simulation
BEGIN EXCHANGES
  GWF6-GWF6 simulation.exg GWF_Model_1 GWF_Model_2
END EXCHANGES

# Models are part of the same numerical solution
BEGIN SOLUTIONGROUP 1
  IMS6 simulation.ims GWF_Model_1 GWF_Model_2
END SOLUTIONGROUP
```

1.1.2 SIM-TDIS

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [TIME_UNITS <time_units>]
  [START_DATE_TIME <start_date_time>]
END OPTIONS
```

```
BEGIN DIMENSIONS
  NPER <nper>
END DIMENSIONS
```

FOR ANY STRESS PERIOD

```
BEGIN PERIODDATA
  <perlen> <nstp> <tsmult>
  <perlen> <nstp> <tsmult>
  ...
END PERIODDATA
```

Explanation of Variables

Block: OPTIONS

- `time_units` is the time units of the simulation. This is a text string that is used as a label within model output files. Values for time_units may be “unknown”, “seconds”, “minutes”, “hours”, “days”, or “years”. The default time unit is “unknown”.
- `start_date_time` is the starting date and time of the simulation. This is a text string that is used as a label within the simulation list file. The value has no affect on the simulation. The recommended format for the starting date and time is described at https://www.w3.org/TR/NOTE-datetime.

Block: DIMENSIONS

- `nper` is the number of stress periods for the simulation.

Block: PERIODDATA

- `perlen` is the length of a stress period.
- `nstp` is the number of time steps in a stress period.
- `tsmult` is the multiplier for the length of successive time steps. The length of a time step is calculated by multiplying the length of the previous time step by TSMULT. The length of the first time step, Delta t_1, is related to PERLEN, NSTP, and TSMULT by the relation Delta t_1= perlen frac{tsmult - 1}{tsmultnstp-1}.

Example Input File

```
# Comment for this TDIS input file

BEGIN OPTIONS
  TIME_UNITS DAYS
END OPTIONS

BEGIN DIMENSIONS
  NPER 2
END DIMENSIONS

BEGIN PERIODDATA
   365.00  1 1.0   Items: PERLEN NSTP TSMULT
   365.00 10 1.2   Items: PERLEN NSTP TSMULT
END PERIODDATA
```

1.2 Iterative Model Solution

1.2.1 SLN-IMS

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [PRINT_OPTION <print_option>]
  [COMPLEXITY <complexity>]
  [CSV_OUTER_OUTPUT FILEOUT <outer_csvfile>]
  [CSV_INNER_OUTPUT FILEOUT <inner_csvfile>]
  [NO_PTC [<no_ptc_option>]]
END OPTIONS
```

```
BEGIN NONLINEAR
  OUTER_DVCLOSE <outer_dvclose>
  OUTER_MAXIMUM <outer_maximum>
  [UNDER_RELAXATION <under_relaxation>]
  [UNDER_RELAXATION_GAMMA <under_relaxation_gamma>]
  [UNDER_RELAXATION_THETA <under_relaxation_theta>]
  [UNDER_RELAXATION_KAPPA <under_relaxation_kappa>]
  [UNDER_RELAXATION_MOMENTUM <under_relaxation_momentum>]
  [BACKTRACKING_NUMBER <backtracking_number>]
  [BACKTRACKING_TOLERANCE <backtracking_tolerance>]
  [BACKTRACKING_REDUCTION_FACTOR <backtracking_reduction_factor>]
  [BACKTRACKING_RESIDUAL_LIMIT <backtracking_residual_limit>]
END NONLINEAR
```

```
BEGIN LINEAR
  INNER_MAXIMUM <inner_maximum>
  INNER_DVCLOSE <inner_dvclose>
  INNER_RCLOSE <inner_rclose> [<rclose_option>]
  LINEAR_ACCELERATION <linear_acceleration>
  [RELAXATION_FACTOR <relaxation_factor>]
  [PRECONDITIONER_LEVELS <preconditioner_levels>]
  [PRECONDITIONER_DROP_TOLERANCE <preconditioner_drop_tolerance>]
  [NUMBER_ORTHOGONALIZATIONS <number_orthogonalizations>]
  [SCALING_METHOD <scaling_method>]
  [REORDERING_METHOD <reordering_method>]
END LINEAR
```

Explanation of Variables

Block: OPTIONS

- `print_option` is a flag that controls printing of convergence information from the solver. NONE means print nothing. SUMMARY means print only the total number of iterations and nonlinear residual reduction summaries. ALL means print linear matrix solver convergence information to the solution listing file and model specific linear matrix solver convergence information to each model listing file in addition to SUMMARY information. NONE is default if PRINT_OPTION is not specified.

- complexity is an optional keyword that defines default non-linear and linear solver parameters. SIMPLE - indicates that default solver input values will be defined that work well for nearly linear models. This would be used for models that do not include nonlinear stress packages and models that are either confined or consist of a single unconfined layer that is thick enough to contain the water table within a single layer. MODERATE - indicates that default solver input values will be defined that work well for moderately nonlinear models. This would be used for models that include nonlinear stress packages and models that consist of one or more unconfined layers. The MODERATE option should be used when the SIMPLE option does not result in successful convergence. COMPLEX - indicates that default solver input values will be defined that work well for highly nonlinear models. This would be used for models that include nonlinear stress packages and models that consist of one or more unconfined layers representing complex geology and surface-water/groundwater interaction. The COMPLEX option should be used when the MODERATE option does not result in successful convergence. Non-linear and linear solver parameters assigned using a specified complexity can be modified in the NONLINEAR and LINEAR blocks. If the COMPLEXITY option is not specified, NONLINEAR and LINEAR variables will be assigned the simple complexity values.
- CSV_OUTER_OUTPUT keyword to specify that the record corresponds to the comma separated values outer iteration convergence output.
- FILEOUT keyword to specify that an output filename is expected next.
- outer_csvfile name of the ascii comma separated values output file to write maximum dependent-variable (for example, head) change convergence information at the end of each outer iteration for each time step.
- CSV_INNER_OUTPUT keyword to specify that the record corresponds to the comma separated values solver convergence output.
- inner_csvfile name of the ascii comma separated values output file to write solver convergence information. Comma separated values output includes maximum dependent-variable (for example, head) change and maximum residual convergence information for the solution and each model (if the solution includes more than one model) and linear acceleration information for each inner iteration.
- NO_PTC is a flag that is used to disable pseudo-transient continuation (PTC). Option only applies to steady-state stress periods for models using the Newton-Raphson formulation. For many problems, PTC can significantly improve convergence behavior for steady-state simulations, and for this reason it is active by default. In some cases, however, PTC can worsen the convergence behavior, especially when the initial conditions are similar to the solution. When the initial conditions are similar to, or exactly the same as, the solution and convergence is slow, then the NO_PTC FIRST option should be used to deactivate PTC for the first stress period. The NO_PTC ALL option should also be used in order to compare convergence behavior with other MODFLOW versions, as PTC is only available in MODFLOW 6.
- no_ptc_option is an optional keyword that is used to define options for disabling pseudo-transient continuation (PTC). FIRST is an optional keyword to disable PTC for the first stress period, if steady-state and one or more model is using the Newton-Raphson formulation. ALL is an optional keyword to disable PTC for all steady-state stress periods for models using the Newton-Raphson formulation. If NO_PTC_OPTION is not specified, the NO_PTC ALL option is used.

Block: NONLINEAR

- outer_dvclose real value defining the dependent-variable (for example, head) change criterion for convergence of the outer (nonlinear) iterations, in units of the dependent-variable (for example, length for head). When the maximum absolute value of the dependent-variable change at all nodes during an iteration is less than or equal to OUTER_DVCLOSE, iteration stops. Commonly, OUTER_DVCLOSE equals 0.01. The keyword, OUTER_HCLOSE can be still be specified instead of OUTER_DVCLOSE for backward compatibility with previous versions of MODFLOW 6 but eventually OUTER_HCLOSE will be deprecated and specification of OUTER_HCLOSE will cause MODFLOW 6 to terminate with an error.

- `outer_maximum` integer value defining the maximum number of outer (nonlinear) iterations – that is, calls to the solution routine. For a linear problem OUTER_MAXIMUM should be 1.
- `under_relaxation` is an optional keyword that defines the nonlinear under-relaxation schemes used. Under-relaxation is also known as dampening, and is used to reduce the size of the calculated dependent variable before proceeding to the next outer iteration. Under-relaxation can be an effective tool for highly nonlinear models when there are large and often counteracting changes in the calculated dependent variable between successive outer iterations. By default under-relaxation is not used. NONE - under-relaxation is not used (default). SIMPLE - Simple under-relaxation scheme with a fixed relaxation factor (UNDER_RELAXATION_GAMMA) is used. COOLEY - Cooley under-relaxation scheme is used. DBD - delta-bar-delta under-relaxation is used. Note that the under-relaxation schemes are often used in conjunction with problems that use the Newton-Raphson formulation, however, experience has indicated that they also work well non-Newton problems, such as those with the wet/dry options of MODFLOW 6.
- `under_relaxation_gamma` real value defining either the relaxation factor for the SIMPLE scheme or the history or memory term factor of the Cooley and delta-bar-delta algorithms. For the SIMPLE scheme, a value of one indicates that there is no under-relaxation and the full head change is applied. This value can be gradually reduced from one as a way to improve convergence; for well behaved problems, using a value less than one can increase the number of outer iterations required for convergence and needlessly increase run times. UNDER_RELAXATION_GAMMA must be greater than zero for the SIMPLE scheme or the program will terminate with an error. For the Cooley and delta-bar-delta schemes, UNDER_RELAXATION_GAMMA is a memory term that can range between zero and one. When UNDER_RELAXATION_GAMMA is zero, only the most recent history (previous iteration value) is maintained. As UNDER_RELAXATION_GAMMA is increased, past history of iteration changes has greater influence on the memory term. The memory term is maintained as an exponential average of past changes. Retaining some past history can overcome granular behavior in the calculated function surface and therefore helps to overcome cyclic patterns of non-convergence. The value usually ranges from 0.1 to 0.3; a value of 0.2 works well for most problems. UNDER_RELAXATION_GAMMA only needs to be specified if UNDER_RELAXATION is not NONE.
- `under_relaxation_theta` real value defining the reduction factor for the learning rate (under-relaxation term) of the delta-bar-delta algorithm. The value of UNDER_RELAXATION_THETA is between zero and one. If the change in the dependent-variable (for example, head) is of opposite sign to that of the previous iteration, the under-relaxation term is reduced by a factor of UNDER_RELAXATION_THETA. The value usually ranges from 0.3 to 0.9; a value of 0.7 works well for most problems. UNDER_RELAXATION_THETA only needs to be specified if UNDER_RELAXATION is DBD.
- `under_relaxation_kappa` real value defining the increment for the learning rate (under-relaxation term) of the delta-bar-delta algorithm. The value of UNDER_RELAXATION_kappa is between zero and one. If the change in the dependent-variable (for example, head) is of the same sign to that of the previous iteration, the under-relaxation term is increased by an increment of UNDER_RELAXATION_KAPPA. The value usually ranges from 0.03 to 0.3; a value of 0.1 works well for most problems. UNDER_RELAXATION_KAPPA only needs to be specified if UNDER_RELAXATION is DBD.
- `under_relaxation_momentum` real value defining the fraction of past history changes that is added as a momentum term to the step change for a nonlinear iteration. The value of UNDER_RELAXATION_MOMENTUM is between zero and one. A large momentum term should only be used when small learning rates are expected. Small amounts of the momentum term help convergence. The value usually ranges from 0.0001 to 0.1; a value of 0.001 works well for most problems. UNDER_RELAXATION_MOMENTUM only needs to be specified if UNDER_RELAXATION is DBD.
- `backtracking_number` integer value defining the maximum number of backtracking iterations allowed for residual reduction computations. If BACKTRACKING_NUMBER = 0 then the backtracking iterations are omitted. The value usually ranges from 2 to 20; a value of 10 works well for most problems.
- `backtracking_tolerance` real value defining the tolerance for residual change that is allowed for residual reduction computations. BACKTRACKING_TOLERANCE should not be less than one to avoid getting stuck in local minima. A large value serves to check for extreme residual increases, while a low value serves to control

step size more severely. The value usually ranges from 1.0 to 106; a value of 104 works well for most problems but lower values like 1.1 may be required for harder problems. BACKTRACKING_TOLERANCE only needs to be specified if BACKTRACKING_NUMBER is greater than zero.

- `backtracking_reduction_factor` real value defining the reduction in step size used for residual reduction computations. The value of BACKTRACKING_REDUCTION_FACTOR is between zero and one. The value usually ranges from 0.1 to 0.3; a value of 0.2 works well for most problems. BACKTRACKING_REDUCTION_FACTOR only needs to be specified if BACKTRACKING_NUMBER is greater than zero.
- `backtracking_residual_limit` real value defining the limit to which the residual is reduced with backtracking. If the residual is smaller than BACKTRACKING_RESIDUAL_LIMIT, then further backtracking is not performed. A value of 100 is suitable for large problems and residual reduction to smaller values may only slow down computations. BACKTRACKING_RESIDUAL_LIMIT only needs to be specified if BACKTRACKING_NUMBER is greater than zero.

Block: LINEAR

- `inner_maximum` integer value defining the maximum number of inner (linear) iterations. The number typically depends on the characteristics of the matrix solution scheme being used. For nonlinear problems, INNER_MAXIMUM usually ranges from 60 to 600; a value of 100 will be sufficient for most linear problems.
- `inner_dvclose` real value defining the dependent-variable (for example, head) change criterion for convergence of the inner (linear) iterations, in units of the dependent-variable (for example, length for head). When the maximum absolute value of the dependent-variable change at all nodes during an iteration is less than or equal to INNER_DVCLOSE, the matrix solver assumes convergence. Commonly, INNER_DVCLOSE is set an order of magnitude less than the OUTER_DVCLOSE value specified for the NONLINEAR block. The keyword, INNER_HCLOSE can be still be specified instead of INNER_DVCLOSE for backward compatibility with previous versions of MODFLOW 6 but eventually INNER_HCLOSE will be deprecated and specification of INNER_HCLOSE will cause MODFLOW 6 to terminate with an error.
- `inner_rclose` real value that defines the flow residual tolerance for convergence of the IMS linear solver and specific flow residual criteria used. This value represents the maximum allowable residual at any single node. Value is in units of length cubed per time, and must be consistent with MODFLOW 6 length and time units. Usually a value of 1.0 x 10-1 is sufficient for the flow-residual criteria when meters and seconds are the defined MODFLOW 6 length and time.
- `rclose_option` an optional keyword that defines the specific flow residual criterion used. STRICT–an optional keyword that is used to specify that INNER_RCLOSE represents a infinity-Norm (absolute convergence criteria) and that the dependent-variable (for example, head) and flow convergence criteria must be met on the first inner iteration (this criteria is equivalent to the criteria used by the MODFLOW-2005 PCG package). L2NORM_RCLOSE–an optional keyword that is used to specify that INNER_RCLOSE represents a L-2 Norm closure criteria instead of a infinity-Norm (absolute convergence criteria). When L2NORM_RCLOSE is specified, a reasonable initial INNER_RCLOSE value is 0.1 times the number of active cells when meters and seconds are the defined MODFLOW 6 length and time. RELATIVE_RCLOSE–an optional keyword that is used to specify that INNER_RCLOSE represents a relative L-2 Norm reduction closure criteria instead of a infinity-Norm (absolute convergence criteria). When RELATIVE_RCLOSE is specified, a reasonable initial INNER_RCLOSE value is 1.0 x 10-4 and convergence is achieved for a given inner (linear) iteration when Delta h INNER_DVCLOSE and the current L-2 Norm is the product of the RELATIVE_RCLOSE and the initial L-2 Norm for the current inner (linear) iteration. If RCLOSE_OPTION is not specified, an absolute residual (infinity-norm) criterion is used.
- `linear_acceleration` a keyword that defines the linear acceleration method used by the default IMS linear solvers. CG - preconditioned conjugate gradient method. BICGSTAB - preconditioned bi-conjugate gradient stabilized method.

- `relaxation_factor` optional real value that defines the relaxation factor used by the incomplete LU factorization preconditioners (MILU(0) and MILUT). RELAXATION_FACTOR is unitless and should be greater than or equal to 0.0 and less than or equal to 1.0. RELAXATION_FACTOR values of about 1.0 are commonly used, and experience suggests that convergence can be optimized in some cases with relax values of 0.97. A RELAXATION_FACTOR value of 0.0 will result in either ILU(0) or ILUT preconditioning (depending on the value specified for PRECONDITIONER_LEVELS and/or PRECONDITIONER_DROP_TOLERANCE). By default, RELAXATION_FACTOR is zero.
- `preconditioner_levels` optional integer value defining the level of fill for ILU decomposition used in the ILUT and MILUT preconditioners. Higher levels of fill provide more robustness but also require more memory. For optimal performance, it is suggested that a large level of fill be applied (7 or 8) with use of a drop tolerance. Specification of a PRECONDITIONER_LEVELS value greater than zero results in use of the ILUT preconditioner. By default, PRECONDITIONER_LEVELS is zero and the zero-fill incomplete LU factorization preconditioners (ILU(0) and MILU(0)) are used.
- `preconditioner_drop_tolerance` optional real value that defines the drop tolerance used to drop preconditioner terms based on the magnitude of matrix entries in the ILUT and MILUT preconditioners. A value of 10-4 works well for most problems. By default, PRECONDITIONER_DROP_TOLERANCE is zero and the zero-fill incomplete LU factorization preconditioners (ILU(0) and MILU(0)) are used.
- `number_orthogonalizations` optional integer value defining the interval used to explicitly recalculate the residual of the flow equation using the solver coefficient matrix, the latest dependent-variable (for example, head) estimates, and the right hand side. For problems that benefit from explicit recalculation of the residual, a number between 4 and 10 is appropriate. By default, NUMBER_ORTHOGONALIZATIONS is zero.
- `scaling_method` an optional keyword that defines the matrix scaling approach used. By default, matrix scaling is not applied. NONE - no matrix scaling applied. DIAGONAL - symmetric matrix scaling using the POLCG preconditioner scaling method in Hill (1992). L2NORM - symmetric matrix scaling using the L2 norm.
- `reordering_method` an optional keyword that defines the matrix reordering approach used. By default, matrix reordering is not applied. NONE - original ordering. RCM - reverse Cuthill McKee ordering. MD - minimum degree ordering.

Example Input File

```
BEGIN OPTIONS
  PRINT_OPTION ALL
  COMPLEXITY MODERATE
END OPTIONS

BEGIN NONLINEAR
  OUTER_DVCLOSE 1.E-4
  OUTER_MAXIMUM 2000
  UNDER_RELAXATION DBD
  UNDER_RELAXATION_THETA 0.70
  UNDER_RELAXATION_KAPPA 0.100000E-03
  UNDER_RELAXATION_GAMMA 0.
  UNDER_RELAXATION_MOMENTUM 0.
  BACKTRACKING_NUMBER 20
  BACKTRACKING_TOLERANCE 2.
  BACKTRACKING_REDUCTION_FACTOR 0.6
  BACKTRACKING_RESIDUAL_LIMIT 5.000000E-04
END NONLINEAR

BEGIN LINEAR
  INNER_MAXIMUM 100
```

(continues on next page)

(continued from previous page)

```
    INNER_DVCLOSE 1.0E-4
    INNER_RCLOSE 0.001
    LINEAR_ACCELERATION BICGSTAB
    RELAXATION_FACTOR 0.97
    SCALING_METHOD NONE
    REORDERING_METHOD NONE
  END LINEAR
```

1.3 Groundwater Flow

1.3.1 GWF-BUY

Structure of Blocks

FOR EACH SIMULATION

```
  BEGIN OPTIONS
    [HHFORMULATION_RHS]
    [DENSEREF <denseref>]
    [DENSITY FILEOUT <densityfile>]
  END OPTIONS
```

```
  BEGIN DIMENSIONS
    NRHOSPECIES <nrhospecies>
  END DIMENSIONS
```

```
  BEGIN PACKAGEDATA
    <irhospec> <drhodc> <crhoref> <modelname> <auxspeciesname>
    <irhospec> <drhodc> <crhoref> <modelname> <auxspeciesname>
    ...
  END PACKAGEDATA
```

Explanation of Variables

Block: OPTIONS

- `HHFORMULATION_RHS` use the variable-density hydraulic head formulation and add off-diagonal terms to the right-hand. This option will prevent the BUY Package from adding asymmetric terms to the flow matrix.
- `denseref` fluid reference density used in the equation of state. This value is set to 1000. if not specified as an option.
- `DENSITY` keyword to specify that record corresponds to density.
- `FILEOUT` keyword to specify that an output filename is expected next.
- `densityfile` name of the binary output file to write density information. The density file has the same format as the head file. Density values will be written to the density file whenever heads are written to the binary head file. The settings for controlling head output are contained in the Output Control option.

Block: DIMENSIONS

- `nrhospecies` number of species used in density equation of state. This value must be one or greater. The value must be one if concentrations are specified using the CONCENTRATION keyword in the PERIOD block below.

Block: PACKAGEDATA

- `irhospec` integer value that defines the species number associated with the specified PACKAGEDATA data on the line. IRHOSPECIES must be greater than zero and less than or equal to NRHOSPECIES. Information must be specified for each of the NRHOSPECIES species or the program will terminate with an error. The program will also terminate with an error if information for a species is specified more than once.
- `drhodc` real value that defines the slope of the density-concentration line for this species used in the density equation of state.
- `crhoref` real value that defines the reference concentration value used for this species in the density equation of state.
- `modelname` name of GWT model used to simulate a species that will be used in the density equation of state. This name will have no affect if the simulation does not include a GWT model that corresponds to this GWF model.
- `auxspeciesname` name of an auxiliary variable in a GWF stress package that will be used for this species to calculate a density value. If a density value is needed by the Buoyancy Package then it will use the concentration values in this AUXSPECIESNAME column in the density equation of state. For advanced stress packages (LAK, SFR, MAW, and UZF) that have an associated advanced transport package (LKT, SFT, MWT, and UZT), the FLOW_PACKAGE_AUXILIARY_NAME option in the advanced transport package can be used to transfer simulated concentrations into the flow package auxiliary variable. In this manner, the Buoyancy Package can calculate density values for lakes, streams, multi-aquifer wells, and unsaturated zone flow cells using simulated concentrations.

Example Input File

```
BEGIN OPTIONS
  DENSEREF 1000.
END OPTIONS

BEGIN DIMENSIONS
  NRHOSPECIES 2
END DIMENSIONS

BEGIN PACKAGEDATA
  #ISPEC DRHODC   CRHOREF MODELNAME   AUXSPECIESNAME
       1     0.7        0.     GWT-1         SALINITY
       2  -0.375       25.     GWT-2      TEMPERATURE
END PACKAGEDATA
```

1.3.2 GWF-CHD

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [AUXILIARY <auxiliary(naux)>]
  [AUXMULTNAME <auxmultname>]
  [BOUNDNAMES]
  [PRINT_INPUT]
  [PRINT_FLOWS]
  [SAVE_FLOWS]
  [TS6 FILEIN <ts6_filename>]
  [OBS6 FILEIN <obs6_filename>]
END OPTIONS
```

```
BEGIN DIMENSIONS
  MAXBOUND <maxbound>
END DIMENSIONS
```

FOR ANY STRESS PERIOD

```
BEGIN PERIOD <iper>
  <cellid(ncelldim)> <head> [<aux(naux)>] [<boundname>]
  <cellid(ncelldim)> <head> [<aux(naux)>] [<boundname>]
  ...
END PERIOD
```

Explanation of Variables

Block: OPTIONS

- `auxiliary` defines an array of one or more auxiliary variable names. There is no limit on the number of auxiliary variables that can be provided on this line; however, lists of information provided in subsequent blocks must have a column of data for each auxiliary variable name defined here. The number of auxiliary variables detected on this line determines the value for naux. Comments cannot be provided anywhere on this line as they will be interpreted as auxiliary variable names. Auxiliary variables may not be used by the package, but they will be available for use by other parts of the program. The program will terminate with an error if auxiliary variables are specified on more than one line in the options block.
- `auxmultname` name of auxiliary variable to be used as multiplier of CHD head value.
- `BOUNDNAMES` keyword to indicate that boundary names may be provided with the list of constant-head cells.
- `PRINT_INPUT` keyword to indicate that the list of constant-head information will be written to the listing file immediately after it is read.
- `PRINT_FLOWS` keyword to indicate that the list of constant-head flow rates will be printed to the listing file for every stress period time step in which "BUDGET PRINT" is specified in Output Control. If there is no Output Control option and "PRINT_FLOWS" is specified, then flow rates are printed for the last time step of each stress period.
- `SAVE_FLOWS` keyword to indicate that constant-head flow terms will be written to the file specified with "BUDGET FILEOUT" in Output Control.
- `TS6` keyword to specify that record corresponds to a time-series file.

- FILEIN keyword to specify that an input filename is expected next.
- ts6_filename defines a time-series file defining time series that can be used to assign time-varying values. See the "Time-Variable Input" section for instructions on using the time-series capability.
- OBS6 keyword to specify that record corresponds to an observations file.
- obs6_filename name of input file to define observations for the constant-head package. See the "Observation utility" section for instructions for preparing observation input files. Tables ref{table:gwf-obstypetable} and ref{table:gwt-obstypetable} lists observation type(s) supported by the constant-head package.

Block: DIMENSIONS

- maxbound integer value specifying the maximum number of constant-head cells that will be specified for use during any stress period.

Block: PERIOD

- iper integer value specifying the starting stress period number for which the data specified in the PERIOD block apply. IPER must be less than or equal to NPER in the TDIS Package and greater than zero. The IPER value assigned to a stress period block must be greater than the IPER value assigned for the previous PERIOD block. The information specified in the PERIOD block will continue to apply for all subsequent stress periods, unless the program encounters another PERIOD block.
- cellid is the cell identifier, and depends on the type of grid that is used for the simulation. For a structured grid that uses the DIS input file, CELLID is the layer, row, and column. For a grid that uses the DISV input file, CELLID is the layer and CELL2D number. If the model uses the unstructured discretization (DISU) input file, CELLID is the node number for the cell.
- head is the head at the boundary. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- aux represents the values of the auxiliary variables for each constant head. The values of auxiliary variables must be present for each constant head. The values must be specified in the order of the auxiliary variables specified in the OPTIONS block. If the package supports time series and the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- boundname name of the constant head boundary cell. BOUNDNAME is an ASCII character variable that can contain as many as 40 characters. If BOUNDNAME contains spaces in it, then the entire name must be enclosed within single quotes.

Example Input File

```
#The OPTIONS block is optional
BEGIN OPTIONS
  AUXILIARY temperature
  BOUNDNAMES
  PRINT_INPUT
  PRINT_FLOWS
  SAVE_FLOWS
END OPTIONS
```

(continues on next page)

(continued from previous page)

```
#The DIMENSIONS block is required
BEGIN DIMENSIONS
  MAXBOUND 2
END DIMENSIONS

#The following block of constant-head cells will be activated
#for stress period 1.  This block will remain active throughout
#the simulation.

BEGIN PERIOD 1
#l r c  head  temperature  boundname
 1 1 2  100.  20.5         chd_1_2
 1 1 3  100.  20.4         chd_1_3
END PERIOD 1
```

Available Observation Types

Example Observation Input File

```
BEGIN OPTIONS
  DIGITS 8
  PRINT_INPUT
END OPTIONS

BEGIN CONTINUOUS FILEOUT my_model.chd01.csv
# obsname  obstype   ID
  chd_2_1  CHD       1 1 2
  chd_2_2  CHD       1 2 2
  chd_2_3  CHD       1 3 2
  chd_2_4  CHD       1 4 2
END SINGLE

BEGIN CONTINUOUS FILEOUT my_model.chd02.csv
# obsname     obstype   ID
  chd_3_flow  CHD       CHD_1_3
END CONTINUOUS
```

1.3.3 GWF-CSUB

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [BOUNDNAMES]
  [PRINT_INPUT]
  [SAVE_FLOWS]
  [GAMMAW <gammaw>]
  [BETA <beta>]
  [HEAD_BASED]
  [INITIAL_PRECONSOLIDATION_HEAD]
  [NDELAYCELLS <ndelaycells>]
```

(continues on next page)

(continued from previous page)

```
      [COMPRESSION_INDICES]
      [UPDATE_MATERIAL_PROPERTIES]
      [CELL_FRACTION]
      [SPECIFIED_INITIAL_INTERBED_STATE]
      [SPECIFIED_INITIAL_PRECONSOLIDATION_STRESS]
      [SPECIFIED_INITIAL_DELAY_HEAD]
      [EFFECTIVE_STRESS_LAG]
      [STRAIN_CSV_INTERBED FILEOUT <interbedstrain_filename>]
      [STRAIN_CSV_COARSE FILEOUT <coarsestrain_filename>]
      [COMPACTION FILEOUT <compaction_filename>]
      [COMPACTION_ELASTIC FILEOUT <elastic_compaction_filename>]
      [COMPACTION_INELASTIC FILEOUT <inelastic_compaction_filename>]
      [COMPACTION_INTERBED FILEOUT <interbed_compaction_filename>]
      [COMPACTION_COARSE FILEOUT <coarse_compaction_filename>]
      [ZDISPLACEMENT FILEOUT <zdisplacement_filename>]
      [PACKAGE_CONVERGENCE FILEOUT <package_convergence_filename>]
      [TS6 FILEIN <ts6_filename>]
      [OBS6 FILEIN <obs6_filename>]
    END OPTIONS
```

```
    BEGIN DIMENSIONS
      NINTERBEDS <ninterbeds>
      [MAXSIG0 <maxsig0>]
    END DIMENSIONS
```

```
    BEGIN GRIDDATA
      CG_SKE_CR
            <cg_ske_cr(nodes)> -- READARRAY
      CG_THETA
            <cg_theta(nodes)> -- READARRAY
      [SGM
            <sgm(nodes)> -- READARRAY]
      [SGS
            <sgs(nodes)> -- READARRAY]
    END GRIDDATA
```

```
    BEGIN PACKAGEDATA
      <icsubno> <cellid(ncelldim)> <cdelay> <pcs0> <thick_frac> <rnb> <ssv_cc> <sse_
↪cr> <theta> <kv> <h0> [<boundname>]
      <icsubno> <cellid(ncelldim)> <cdelay> <pcs0> <thick_frac> <rnb> <ssv_cc> <sse_
↪cr> <theta> <kv> <h0> [<boundname>]
      ...
    END PACKAGEDATA
```

FOR ANY STRESS PERIOD

```
    BEGIN PERIOD <iper>
      <cellid(ncelldim)> <sig0>
      <cellid(ncelldim)> <sig0>
      ...
    END PERIOD
```

Explanation of Variables

Block: OPTIONS

- BOUNDNAMES keyword to indicate that boundary names may be provided with the list of CSUB cells.
- PRINT_INPUT keyword to indicate that the list of CSUB information will be written to the listing file immediately after it is read.
- SAVE_FLOWS keyword to indicate that cell-by-cell flow terms will be written to the file specified with "BUDGET SAVE FILE" in Output Control.
- gammaw unit weight of water. For freshwater, GAMMAW is 9806.65 Newtons/cubic meters or 62.48 lb/cubic foot in SI and English units, respectively. By default, GAMMAW is 9806.65 Newtons/cubic meters.
- beta compressibility of water. Typical values of BETA are 4.6512e-10 1/Pa or 2.2270e-8 lb/square foot in SI and English units, respectively. By default, BETA is 4.6512e-10 1/Pa.
- HEAD_BASED keyword to indicate the head-based formulation will be used to simulate coarse-grained aquifer materials and no-delay and delay interbeds. Specifying HEAD_BASED also specifies the INITIAL_PRECONSOLIDATION_HEAD option.
- INITIAL_PRECONSOLIDATION_HEAD keyword to indicate that preconsolidation heads will be specified for no-delay and delay interbeds in the PACKAGEDATA block. If the SPECIFIED_INITIAL_INTERBED_STATE option is specified in the OPTIONS block, user-specified preconsolidation heads in the PACKAGEDATA block are absolute values. Otherwise, user-specified preconsolidation heads in the PACKAGEDATA block are relative to steady-state or initial heads.
- ndelaycells number of nodes used to discretize delay interbeds. If not specified, then a default value of 19 is assigned.
- COMPRESSION_INDICES keyword to indicate that the recompression (CR) and compression (CC) indices are specified instead of the elastic specific storage (SSE) and inelastic specific storage (SSV) coefficients. If not specified, then elastic specific storage (SSE) and inelastic specific storage (SSV) coefficients must be specified.
- UPDATE_MATERIAL_PROPERTIES keyword to indicate that the thickness and void ratio of coarse-grained and interbed sediments (delay and no-delay) will vary during the simulation. If not specified, the thickness and void ratio of coarse-grained and interbed sediments will not vary during the simulation.
- CELL_FRACTION keyword to indicate that the thickness of interbeds will be specified in terms of the fraction of cell thickness. If not specified, interbed thicknness must be specified.
- SPECIFIED_INITIAL_INTERBED_STATE keyword to indicate that absolute preconsolidation stresses (heads) and delay bed heads will be specified for interbeds defined in the PACKAGEDATA block. The SPECIFIED_INITIAL_INTERBED_STATE option is equivalent to specifying the SPECIFIED_INITIAL_PRECONSOLITATION_STRESS and SPECIFIED_INITIAL_DELAY_HEAD. If SPECIFIED_INITIAL_INTERBED_STATE is not specified then preconsolidation stress (head) and delay bed head values specified in the PACKAGEDATA block are relative to simulated values of the first stress period if steady-state or initial stresses and GWF heads if the first stress period is transient.
- SPECIFIED_INITIAL_PRECONSOLIDATION_STRESS keyword to indicate that absolute preconsolidation stresses (heads) will be specified for interbeds defined in the PACKAGEDATA block. If SPECIFIED_INITIAL_PRECONSOLITATION_STRESS and SPECIFIED_INITIAL_INTERBED_STATE are not specified then preconsolidation stress (head) values specified in the PACKAGEDATA block are relative to simulated values if the first stress period is steady-state or initial stresses (heads) if the first stress period is transient.
- SPECIFIED_INITIAL_DELAY_HEAD keyword to indicate that absolute initial delay bed head will be specified for interbeds defined in the PACKAGEDATA block. If SPECIFIED_INITIAL_DELAY_HEAD and SPECIFIED_INITIAL_INTERBED_STATE are not specified then delay bed head values specified in the PACKAGE-

DATA block are relative to simulated values if the first stress period is steady-state or initial GWF heads if the first stress period is transient.

- EFFECTIVE_STRESS_LAG keyword to indicate the effective stress from the previous time step will be used to calculate specific storage values. This option can 1) help with convergence in models with thin cells and water table elevations close to land surface; 2) is identical to the approach used in the SUBWT package for MODFLOW-2005; and 3) is only used if the effective-stress formulation is being used. By default, current effective stress values are used to calculate specific storage values.
- STRAIN_CSV_INTERBED keyword to specify the record that corresponds to final interbed strain output.
- FILEOUT keyword to specify that an output filename is expected next.
- interbedstrain_filename name of the comma-separated-values output file to write final interbed strain information.
- STRAIN_CSV_COARSE keyword to specify the record that corresponds to final coarse-grained material strain output.
- coarsestrain_filename name of the comma-separated-values output file to write final coarse-grained material strain information.
- COMPACTION keyword to specify that record corresponds to the compaction.
- compaction_filename name of the binary output file to write compaction information.
- COMPACTION_ELASTIC keyword to specify that record corresponds to the elastic interbed compaction binary file.
- elastic_compaction_filename name of the binary output file to write elastic interbed compaction information.
- COMPACTION_INELASTIC keyword to specify that record corresponds to the inelastic interbed compaction binary file.
- inelastic_compaction_filename name of the binary output file to write inelastic interbed compaction information.
- COMPACTION_INTERBED keyword to specify that record corresponds to the interbed compaction binary file.
- interbed_compaction_filename name of the binary output file to write interbed compaction information.
- COMPACTION_COARSE keyword to specify that record corresponds to the elastic coarse-grained material compaction binary file.
- coarse_compaction_filename name of the binary output file to write elastic coarse-grained material compaction information.
- ZDISPLACEMENT keyword to specify that record corresponds to the z-displacement binary file.
- zdisplacement_filename name of the binary output file to write z-displacement information.
- PACKAGE_CONVERGENCE keyword to specify that record corresponds to the package convergence comma spaced values file.
- package_convergence_filename name of the comma spaced values output file to write package convergence information.
- TS6 keyword to specify that record corresponds to a time-series file.
- FILEIN keyword to specify that an input filename is expected next.
- ts6_filename defines a time-series file defining time series that can be used to assign time-varying values. See the "Time-Variable Input" section for instructions on using the time-series capability.

- OBS6 keyword to specify that record corresponds to an observations file.
- obs6_filename name of input file to define observations for the CSUB package. See the "Observation utility" section for instructions for preparing observation input files. Tables ref{table:gwf-obstypetable} and ref{table:gwt-obstypetable} lists observation type(s) supported by the CSUB package.

Block: DIMENSIONS

- ninterbeds is the number of CSUB interbed systems. More than 1 CSUB interbed systems can be assigned to a GWF cell; however, only 1 GWF cell can be assigned to a single CSUB interbed system.
- maxsig0 is the maximum number of cells that can have a specified stress offset. More than 1 stress offset can be assigned to a GWF cell. By default, MAXSIG0 is 0.

Block: GRIDDATA

- cg_ske_cr is the initial elastic coarse-grained material specific storage or recompression index. The recompression index is specified if COMPRESSION_INDICES is specified in the OPTIONS block. Specified or calculated elastic coarse-grained material specific storage values are not adjusted from initial values if HEAD_BASED is specified in the OPTIONS block.
- cg_theta is the initial porosity of coarse-grained materials.
- sgm is the specific gravity of moist or unsaturated sediments. If not specified, then a default value of 1.7 is assigned.
- sgs is the specific gravity of saturated sediments. If not specified, then a default value of 2.0 is assigned.

Block: PACKAGEDATA

- icsubno integer value that defines the CSUB interbed number associated with the specified PACKAGEDATA data on the line. CSUBNO must be greater than zero and less than or equal to NINTERBEDS. CSUB information must be specified for every CSUB cell or the program will terminate with an error. The program will also terminate with an error if information for a CSUB interbed number is specified more than once.
- cellid is the cell identifier, and depends on the type of grid that is used for the simulation. For a structured grid that uses the DIS input file, CELLID is the layer, row, and column. For a grid that uses the DISV input file, CELLID is the layer and CELL2D number. If the model uses the unstructured discretization (DISU) input file, CELLID is the node number for the cell.
- cdelay character string that defines the subsidence delay type for the interbed. Possible subsidence package CDELAY strings include: NODELAY–character keyword to indicate that delay will not be simulated in the interbed. DELAY–character keyword to indicate that delay will be simulated in the interbed.
- pcs0 is the initial offset from the calculated initial effective stress or initial preconsolidation stress in the interbed, in units of height of a column of water. PCS0 is the initial preconsolidation stress if SPECIFIED_INITIAL_INTERBED_STATE or SPECIFIED_INITIAL_PRECONSOLIDATION_STRESS are specified in the OPTIONS block. If HEAD_BASED is specified in the OPTIONS block, PCS0 is the initial offset from the calculated initial head or initial preconsolidation head in the CSUB interbed and the initial preconsolidation stress is calculated from the calculated initial effective stress or calculated initial geostatic stress, respectively.
- thick_frac is the interbed thickness or cell fraction of the interbed. Interbed thickness is specified as a fraction of the cell thickness if CELL_FRACTION is specified in the OPTIONS block.

- `rnb` is the interbed material factor equivalent number of interbeds in the interbed system represented by the interbed. RNB must be greater than or equal to 1 if CDELAY is DELAY. Otherwise, RNB can be any value.
- `ssv_cc` is the initial inelastic specific storage or compression index of the interbed. The compression index is specified if COMPRESSION_INDICES is specified in the OPTIONS block. Specified or calculated interbed inelastic specific storage values are not adjusted from initial values if HEAD_BASED is specified in the OPTIONS block.
- `sse_cr` is the initial elastic coarse-grained material specific storage or recompression index of the interbed. The recompression index is specified if COMPRESSION_INDICES is specified in the OPTIONS block. Specified or calculated interbed elastic specific storage values are not adjusted from initial values if HEAD_BASED is specified in the OPTIONS block.
- `theta` is the initial porosity of the interbed.
- `kv` is the vertical hydraulic conductivity of the delay interbed. KV must be greater than 0 if CDELAY is DELAY. Otherwise, KV can be any value.
- `h0` is the initial offset from the head in cell cellid or the initial head in the delay interbed. H0 is the initial head in the delay bed if SPECIFIED_INITIAL_INTERBED_STATE or SPECIFIED_INITIAL_DELAY_HEAD are specified in the OPTIONS block. H0 can be any value if CDELAY is NODELAY.
- `boundname` name of the CSUB cell. BOUNDNAME is an ASCII character variable that can contain as many as 40 characters. If BOUNDNAME contains spaces in it, then the entire name must be enclosed within single quotes.

Block: PERIOD

- `iper` integer value specifying the starting stress period number for which the data specified in the PERIOD block apply. IPER must be less than or equal to NPER in the TDIS Package and greater than zero. The IPER value assigned to a stress period block must be greater than the IPER value assigned for the previous PERIOD block. The information specified in the PERIOD block will continue to apply for all subsequent stress periods, unless the program encounters another PERIOD block.
- `cellid` is the cell identifier, and depends on the type of grid that is used for the simulation. For a structured grid that uses the DIS input file, CELLID is the layer, row, and column. For a grid that uses the DISV input file, CELLID is the layer and CELL2D number. If the model uses the unstructured discretization (DISU) input file, CELLID is the node number for the cell.
- `sig0` is the stress offset for the cell. SIG0 is added to the calculated geostatic stress for the cell. SIG0 is specified only if MAXSIG0 is specified to be greater than 0 in the DIMENSIONS block. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.

Example Input File

```
  BEGIN OPTIONS
    COMPRESSION_INDICES
    SPECIFIED_INITIAL_INTERBED_STATE
    BOUNDNAMES
    SAVE_FLOWS
  END OPTIONS

  BEGIN DIMENSIONS
    NINTERBEDS  4
    MAXSIG0  1
```

(continues on next page)

(continued from previous page)

```
  END DIMENSIONS

  BEGIN GRIDDATA
    # compression indices of coarse grained aquifer materials
    cg_ske_cr LAYERED
      CONSTANT       0.01
      CONSTANT       0.01
      CONSTANT       0.01
      CONSTANT       0.01
    # porosity of coarse grained aquifer materials
    cg_theta LAYERED
      CONSTANT       0.45
      CONSTANT       0.45
      CONSTANT       0.45
      CONSTANT       0.45
    # specific gravity of saturated sediment
    SGS LAYERED
        CONSTANT 2.0
        CONSTANT 2.0
        CONSTANT 2.0
        CONSTANT 2.0
    # specific gravity of moist sediment
    SGM LAYERED
        CONSTANT 1.7
        CONSTANT 1.7
        CONSTANT 1.7
        CONSTANT 1.7
  END GRIDDATA

  BEGIN PACKAGEDATA
  # icsubsno cellid cdelay     pcs0  thick_frac rnb ssv_cc sse_cr theta     kv  h0␣
↪boundname
            1 1 1 6   delay    15.0       0.450 1.0   0.25    0.01    0.45 0.1 15.␣
↪nsystm0
            2 1 1 7   nodelay 15.0        0.450 1.0   0.25    0.01    0.45 0.0 0.0␣
↪nsystm1
            3 1 1 8   nodelay 15.0        0.450 1.0   0.25    0.01    0.45 0.0 0.0␣
↪nsystm1
            4 1 1 9   delay    15.0       0.450 1.0   0.25    0.01    0.45 0.1 15.␣
↪nsystm2
  END PACKAGEDATA

  BEGIN PERIOD 1
  # stress offset for stress period 1
    1 1 6    1700.00000000
  END PERIOD
```

Available Observation Types

Example Observation Input File

```
BEGIN CONTINUOUS FILEOUT  my_model.csub.csv
  tcomp3        compaction-cell        1 1 7
  ibcensystm0   elastic-compaction     nsystm0
  ibcinsystm0   inelastic-compaction   nsystm0
END CONTINUOUS
```

1.3.4 GWF-DIS

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [LENGTH_UNITS <length_units>]
  [NOGRB]
  [XORIGIN <xorigin>]
  [YORIGIN <yorigin>]
  [ANGROT <angrot>]
END OPTIONS
```

```
BEGIN DIMENSIONS
  NLAY <nlay>
  NROW <nrow>
  NCOL <ncol>
END DIMENSIONS
```

```
BEGIN GRIDDATA
  DELR
        <delr(ncol)> -- READARRAY
  DELC
        <delc(nrow)> -- READARRAY
  TOP
        <top(ncol, nrow)> -- READARRAY
  BOTM [LAYERED]
        <botm(ncol, nrow, nlay)> -- READARRAY
  [IDOMAIN [LAYERED]
        <idomain(ncol, nrow, nlay)> -- READARRAY]
END GRIDDATA
```

Explanation of Variables

Block: OPTIONS

- `length_units` is the length units used for this model. Values can be "FEET", "METERS", or "CENTIMETERS". If not specified, the default is "UNKNOWN".
- `NOGRB` keyword to deactivate writing of the binary grid file.
- `xorigin` x-position of the lower-left corner of the model grid. A default value of zero is assigned if not specified. The value for XORIGIN does not affect the model simulation, but it is written to the binary grid file so that postprocessors can locate the grid in space.
- `yorigin` y-position of the lower-left corner of the model grid. If not specified, then a default value equal to zero is used. The value for YORIGIN does not affect the model simulation, but it is written to the binary grid file so that postprocessors can locate the grid in space.
- `angrot` counter-clockwise rotation angle (in degrees) of the lower-left corner of the model grid. If not specified, then a default value of 0.0 is assigned. The value for ANGROT does not affect the model simulation, but it is written to the binary grid file so that postprocessors can locate the grid in space.

Block: DIMENSIONS

- `nlay` is the number of layers in the model grid.
- `nrow` is the number of rows in the model grid.
- `ncol` is the number of columns in the model grid.

Block: GRIDDATA

- `delr` is the column spacing in the row direction.
- `delc` is the row spacing in the column direction.
- `top` is the top elevation for each cell in the top model layer.
- `botm` is the bottom elevation for each cell.
- `idomain` is an optional array that characterizes the existence status of a cell. If the IDOMAIN array is not specified, then all model cells exist within the solution. If the IDOMAIN value for a cell is 0, the cell does not exist in the simulation. Input and output values will be read and written for the cell, but internal to the program, the cell is excluded from the solution. If the IDOMAIN value for a cell is 1 or greater, the cell exists in the simulation. If the IDOMAIN value for a cell is -1, the cell does not exist in the simulation. Furthermore, the first existing cell above will be connected to the first existing cell below. This type of cell is referred to as a "vertical pass through" cell.

Example Input File

Example 1

```
#The OPTIONS block is optional
BEGIN OPTIONS
  LENGTH_UNITS METERS
END OPTIONS

#The DIMENSIONS block is required
BEGIN DIMENSIONS
  NLAY 10
  NROW 1
  NCOL 21
END DIMENSIONS

#The GRIDDATA block is required
BEGIN GRIDDATA
  DELR
    INTERNAL FACTOR 1.
        .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 0.01
  DELC
    CONSTANT 1.0
  TOP LAYERED
    CONSTANT 1.
  BOTM LAYERED
    CONSTANT 0.9
    CONSTANT 0.8
    CONSTANT 0.7
    CONSTANT 0.6
    CONSTANT 0.5
    CONSTANT 0.4
    CONSTANT 0.3
    CONSTANT 0.2
    CONSTANT 0.1
    CONSTANT 0.0
END GRIDDATA
```

Example 2

```
BEGIN OPTIONS
  LENGTH_UNITS METERS
END OPTIONS

BEGIN DIMENSIONS
  NODES 9
  NJA 33
END DIMENSIONS

BEGIN GRIDDATA
  TOP
    CONSTANT 0.
  BOT
    CONSTANT -10
  AREA
    INTERNAL FACTOR 1
      10000 10000 10000 10000 10000 10000 10000 10000 10000
END GRIDDATA
```

(continues on next page)

(continued from previous page)

```
  BEGIN CONNECTIONDATA
    IHC
      CONSTANT 1
    IAC
      INTERNAL FACTOR 1
      3 4 3 4 5 4 3 4 3
    JA
      INTERNAL FACTOR 1
      1 2 4
      2 1 3 5
      3 2 6
      4 1 5 7
      5 2 4 6 8
      6 3 5 9
      7 4 8
      8 5 7 9
      9 6 8
    CL12
      INTERNAL FACTOR 1
      0 50 50
      0 50 50 50
      0 50 50
      0 50 50 50
      0 50 50 50 50
      0 50 50 50
      0 50 50
      0 50 50 50
      0 50 50
    HWVA
      INTERNAL FACTOR 1
      0 100 100
      0 100 100 100
      0 100 100
      0 100 100 100
      0 100 100 100 100
      0 100 100 100
      0 100 100
      0 100 100 100
      0 100 100
  END CONNECTIONDATA
```

Example 3

```
  #The OPTIONS block is optional
  BEGIN OPTIONS
    LENGTH_UNITS METERS
  END OPTIONS

  #The DIMENSIONS block is required
  BEGIN DIMENSIONS
    NCPL 4
    NLAY 3
    NVERT 9
  END DIMENSIONS

  #The GRIDDATA block is required
```

(continues on next page)

(continued from previous page)

```
  BEGIN GRIDDATA
    TOP
      CONSTANT 3.0
    BOTM LAYERED
      CONSTANT 2.0
      CONSTANT 1.0
      CONSTANT 0.0
    IDOMAIN LAYERED
      INTERNAL FACTOR 1
        1 1 1 0
      CONSTANT 1
      CONSTANT 1
  END GRIDDATA

  #The VERTICES block is required
  BEGIN VERTICES
    1 0. 1.
    2 .5 1.
    3 1. 1.
    4 0 .5
    5 .5 .5
    6 1. .5
    7 0. 0.
    8 .5 0.
    9 1. 0.
  END VERTICES

  BEGIN CELL2D
    1 .25 .75 4 1 2 5 4
    2 .75 .75 4 2 3 6 5
    3 .25 .25 4 4 5 8 7
    4 .75 .25 4 5 6 9 8
  END CELL2D
```

1.3.5 GWF-DISU

Structure of Blocks

FOR EACH SIMULATION

```
  BEGIN OPTIONS
    [LENGTH_UNITS <length_units>]
    [NOGRB]
    [XORIGIN <xorigin>]
    [YORIGIN <yorigin>]
    [ANGROT <angrot>]
  END OPTIONS
```

```
  BEGIN DIMENSIONS
    NODES <nodes>
    NJA <nja>
    [NVERT <nvert>]
  END DIMENSIONS
```

```
BEGIN GRIDDATA
  TOP
        <top(nodes)> -- READARRAY
  BOT
        <bot(nodes)> -- READARRAY
  AREA
        <area(nodes)> -- READARRAY
  [IDOMAIN
        <idomain(nodes)> -- READARRAY]
END GRIDDATA
```

```
BEGIN CONNECTIONDATA
  IAC
        <iac(nodes)> -- READARRAY
  JA
        <ja(nja)> -- READARRAY
  IHC
        <ihc(nja)> -- READARRAY
  CL12
        <cl12(nja)> -- READARRAY
  HWVA
        <hwva(nja)> -- READARRAY
  [ANGLDEGX
        <angldegx(nja)> -- READARRAY]
END CONNECTIONDATA
```

```
BEGIN VERTICES
  <iv> <xv> <yv>
  <iv> <xv> <yv>
  ...
END VERTICES
```

```
BEGIN CELL2D
  <icell2d> <xc> <yc> <ncvert> <icvert(ncvert)>
  <icell2d> <xc> <yc> <ncvert> <icvert(ncvert)>
  ...
END CELL2D
```

Explanation of Variables

Block: OPTIONS

- `length_units` is the length units used for this model. Values can be "FEET", "METERS", or "CENTIMETERS". If not specified, the default is "UNKNOWN".
- `NOGRB` keyword to deactivate writing of the binary grid file.
- `xorigin` x-position of the origin used for model grid vertices. This value should be provided in a real-world coordinate system. A default value of zero is assigned if not specified. The value for XORIGIN does not affect the model simulation, but it is written to the binary grid file so that postprocessors can locate the grid in space.
- `yorigin` y-position of the origin used for model grid vertices. This value should be provided in a real-world coordinate system. If not specified, then a default value equal to zero is used. The value for YORIGIN does not affect the model simulation, but it is written to the binary grid file so that postprocessors can locate the grid in space.

- angrot counter-clockwise rotation angle (in degrees) of the model grid coordinate system relative to a real-world coordinate system. If not specified, then a default value of 0.0 is assigned. The value for ANGROT does not affect the model simulation, but it is written to the binary grid file so that postprocessors can locate the grid in space.

Block: DIMENSIONS

- nodes is the number of cells in the model grid.
- nja is the sum of the number of connections and NODES. When calculating the total number of connections, the connection between cell n and cell m is considered to be different from the connection between cell m and cell n. Thus, NJA is equal to the total number of connections, including n to m and m to n, and the total number of cells.
- nvert is the total number of (x, y) vertex pairs used to define the plan-view shape of each cell in the model grid. If NVERT is not specified or is specified as zero, then the VERTICES and CELL2D blocks below are not read. NVERT and the accompanying VERTICES and CELL2D blocks should be specified for most simulations. If the XT3D or SAVE_SPECIFIC_DISCHARGE options are specified in the NPF Package, then this information is required.

Block: GRIDDATA

- top is the top elevation for each cell in the model grid.
- bot is the bottom elevation for each cell.
- area is the cell surface area (in plan view).
- idomain is an optional array that characterizes the existence status of a cell. If the IDOMAIN array is not specified, then all model cells exist within the solution. If the IDOMAIN value for a cell is 0, the cell does not exist in the simulation. Input and output values will be read and written for the cell, but internal to the program, the cell is excluded from the solution. If the IDOMAIN value for a cell is 1 or greater, the cell exists in the simulation. IDOMAIN values of -1 cannot be specified for the DISU Package.

Block: CONNECTIONDATA

- iac is the number of connections (plus 1) for each cell. The sum of all the entries in IAC must be equal to NJA.
- ja is a list of cell number (n) followed by its connecting cell numbers (m) for each of the m cells connected to cell n. The number of values to provide for cell n is IAC(n). This list is sequentially provided for the first to the last cell. The first value in the list must be cell n itself, and the remaining cells must be listed in an increasing order (sorted from lowest number to highest). Note that the cell and its connections are only supplied for the GWF cells and their connections to the other GWF cells. Also note that the JA list input may be divided such that every node and its connectivity list can be on a separate line for ease in readability of the file. To further ease readability of the file, the node number of the cell whose connectivity is subsequently listed, may be expressed as a negative number, the sign of which is subsequently converted to positive by the code.
- ihc is an index array indicating the direction between node n and all of its m connections. If IHC = 0 then cell n and cell m are connected in the vertical direction. Cell n overlies cell m if the cell number for n is less than m; cell m overlies cell n if the cell number for m is less than n. If IHC = 1 then cell n and cell m are connected in the horizontal direction. If IHC = 2 then cell n and cell m are connected in the horizontal direction, and the connection is vertically staggered. A vertically staggered connection is one in which a cell is horizontally connected to more than one cell in a horizontal connection.

- cl12 is the array containing connection lengths between the center of cell n and the shared face with each adjacent m cell.
- hwva is a symmetric array of size NJA. For horizontal connections, entries in HWVA are the horizontal width perpendicular to flow. For vertical connections, entries in HWVA are the vertical area for flow. Thus, values in the HWVA array contain dimensions of both length and area. Entries in the HWVA array have a one-to-one correspondence with the connections specified in the JA array. Likewise, there is a one-to-one correspondence between entries in the HWVA array and entries in the IHC array, which specifies the connection type (horizontal or vertical). Entries in the HWVA array must be symmetric; the program will terminate with an error if the value for HWVA for an n to m connection does not equal the value for HWVA for the corresponding n to m connection.
- angldegx is the angle (in degrees) between the horizontal x-axis and the outward normal to the face between a cell and its connecting cells. The angle varies between zero and 360.0 degrees, where zero degrees points in the positive x-axis direction, and 90 degrees points in the positive y-axis direction. ANGLDEGX is only needed if horizontal anisotropy is specified in the NPF Package, if the XT3D option is used in the NPF Package, or if the SAVE_SPECIFIC_DISCHARGE option is specifed in the NPF Package. ANGLDEGX does not need to be specified if these conditions are not met. ANGLDEGX is of size NJA; values specified for vertical connections and for the diagonal position are not used. Note that ANGLDEGX is read in degrees, which is different from MODFLOW-USG, which reads a similar variable (ANGLEX) in radians.

Block: VERTICES

- iv is the vertex number. Records in the VERTICES block must be listed in consecutive order from 1 to NVERT.
- xv is the x-coordinate for the vertex.
- yv is the y-coordinate for the vertex.

Block: CELL2D

- icell2d is the cell2d number. Records in the CELL2D block must be listed in consecutive order from 1 to NODES.
- xc is the x-coordinate for the cell center.
- yc is the y-coordinate for the cell center.
- ncvert is the number of vertices required to define the cell. There may be a different number of vertices for each cell.
- icvert is an array of integer values containing vertex numbers (in the VERTICES block) used to define the cell. Vertices must be listed in clockwise order.

Example Input File

```
BEGIN OPTIONS
  LENGTH_UNITS METERS
END OPTIONS

BEGIN DIMENSIONS
  NODES 9
  NJA 33
END DIMENSIONS
```

(continues on next page)

(continued from previous page)

```
  BEGIN GRIDDATA
    TOP
      CONSTANT 0.
    BOT
      CONSTANT -10
    AREA
      INTERNAL FACTOR 1
        10000 10000 10000 10000 10000 10000 10000 10000 10000
  END GRIDDATA

  BEGIN CONNECTIONDATA
    IHC
      CONSTANT 1
    IAC
      INTERNAL FACTOR 1
      3 4 3 4 5 4 3 4 3
    JA
      INTERNAL FACTOR 1
      1 2 4
      2 1 3 5
      3 2 6
      4 1 5 7
      5 2 4 6 8
      6 3 5 9
      7 4 8
      8 5 7 9
      9 6 8
    CL12
      INTERNAL FACTOR 1
      0 50 50
      0 50 50 50
      0 50 50
      0 50 50 50
      0 50 50 50 50
      0 50 50 50
      0 50 50
      0 50 50 50
      0 50 50
    HWVA
      INTERNAL FACTOR 1
      0 100 100
      0 100 100 100
      0 100 100
      0 100 100 100
      0 100 100 100 100
      0 100 100 100
      0 100 100
      0 100 100 100
      0 100 100
  END CONNECTIONDATA
```

1.3.6 GWF-DISV

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [LENGTH_UNITS <length_units>]
  [NOGRB]
  [XORIGIN <xorigin>]
  [YORIGIN <yorigin>]
  [ANGROT <angrot>]
END OPTIONS
```

```
BEGIN DIMENSIONS
  NLAY <nlay>
  NCPL <ncpl>
  NVERT <nvert>
END DIMENSIONS
```

```
BEGIN GRIDDATA
  TOP
        <top(ncpl)> -- READARRAY
  BOTM [LAYERED]
        <botm(nlay, ncpl)> -- READARRAY
  [IDOMAIN [LAYERED]
        <idomain(nlay, ncpl)> -- READARRAY]
END GRIDDATA
```

```
BEGIN VERTICES
  <iv> <xv> <yv>
  <iv> <xv> <yv>
  ...
END VERTICES
```

```
BEGIN CELL2D
  <icell2d> <xc> <yc> <ncvert> <icvert(ncvert)>
  <icell2d> <xc> <yc> <ncvert> <icvert(ncvert)>
  ...
END CELL2D
```

Explanation of Variables

Block: OPTIONS

- `length_units` is the length units used for this model. Values can be “FEET”, “METERS”, or “CENTIMETERS”. If not specified, the default is “UNKNOWN”.
- `NOGRB` keyword to deactivate writing of the binary grid file.
- `xorigin` x-position of the origin used for model grid vertices. This value should be provided in a real-world coordinate system. A default value of zero is assigned if not specified. The value for XORIGIN does not affect the model simulation, but it is written to the binary grid file so that postprocessors can locate the grid in space.

- `yorigin` y-position of the origin used for model grid vertices. This value should be provided in a real-world coordinate system. If not specified, then a default value equal to zero is used. The value for YORIGIN does not affect the model simulation, but it is written to the binary grid file so that postprocessors can locate the grid in space.
- `angrot` counter-clockwise rotation angle (in degrees) of the model grid coordinate system relative to a real-world coordinate system. If not specified, then a default value of 0.0 is assigned. The value for ANGROT does not affect the model simulation, but it is written to the binary grid file so that postprocessors can locate the grid in space.

Block: DIMENSIONS

- `nlay` is the number of layers in the model grid.
- `ncpl` is the number of cells per layer. This is a constant value for the grid and it applies to all layers.
- `nvert` is the total number of (x, y) vertex pairs used to characterize the horizontal configuration of the model grid.

Block: GRIDDATA

- `top` is the top elevation for each cell in the top model layer.
- `botm` is the bottom elevation for each cell.
- `idomain` is an optional array that characterizes the existence status of a cell. If the IDOMAIN array is not specified, then all model cells exist within the solution. If the IDOMAIN value for a cell is 0, the cell does not exist in the simulation. Input and output values will be read and written for the cell, but internal to the program, the cell is excluded from the solution. If the IDOMAIN value for a cell is 1 or greater, the cell exists in the simulation. If the IDOMAIN value for a cell is -1, the cell does not exist in the simulation. Furthermore, the first existing cell above will be connected to the first existing cell below. This type of cell is referred to as a "vertical pass through" cell.

Block: VERTICES

- `iv` is the vertex number. Records in the VERTICES block must be listed in consecutive order from 1 to NVERT.
- `xv` is the x-coordinate for the vertex.
- `yv` is the y-coordinate for the vertex.

Block: CELL2D

- `icell2d` is the CELL2D number. Records in the CELL2D block must be listed in consecutive order from the first to the last.
- `xc` is the x-coordinate for the cell center.
- `yc` is the y-coordinate for the cell center.
- `ncvert` is the number of vertices required to define the cell. There may be a different number of vertices for each cell.
- `icvert` is an array of integer values containing vertex numbers (in the VERTICES block) used to define the cell. Vertices must be listed in clockwise order. Cells that are connected must share vertices.

Example Input File

```
#The OPTIONS block is optional
BEGIN OPTIONS
  LENGTH_UNITS METERS
END OPTIONS

#The DIMENSIONS block is required
BEGIN DIMENSIONS
  NCPL 4
  NLAY 3
  NVERT 9
END DIMENSIONS

#The GRIDDATA block is required
BEGIN GRIDDATA
  TOP
    CONSTANT 3.0
  BOTM LAYERED
    CONSTANT 2.0
    CONSTANT 1.0
    CONSTANT 0.0
  IDOMAIN LAYERED
    INTERNAL FACTOR 1
      1 1 1 0
    CONSTANT 1
    CONSTANT 1
END GRIDDATA

#The VERTICES block is required
BEGIN VERTICES
  1 0. 1.
  2 .5 1.
  3 1. 1.
  4 0 .5
  5 .5 .5
  6 1. .5
  7 0. 0.
  8 .5 0.
  9 1. 0.
END VERTICES

BEGIN CELL2D
  1 .25 .75 4 1 2 5 4
  2 .75 .75 4 2 3 6 5
  3 .25 .25 4 4 5 8 7
  4 .75 .25 4 5 6 9 8
END CELL2D
```

1.3.7 GWF-DRN

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [AUXILIARY <auxiliary(naux)>]
  [AUXMULTNAME <auxmultname>]
  [AUXDEPTHNAME <auxdepthname>]
  [BOUNDNAMES]
  [PRINT_INPUT]
  [PRINT_FLOWS]
  [SAVE_FLOWS]
  [TS6 FILEIN <ts6_filename>]
  [OBS6 FILEIN <obs6_filename>]
  [MOVER]
END OPTIONS
```

```
BEGIN DIMENSIONS
  MAXBOUND <maxbound>
END DIMENSIONS
```

FOR ANY STRESS PERIOD

```
BEGIN PERIOD <iper>
  <cellid(ncelldim)> <elev> <cond> [<aux(naux)>] [<boundname>]
  <cellid(ncelldim)> <elev> <cond> [<aux(naux)>] [<boundname>]
  ...
END PERIOD
```

Explanation of Variables

Block: OPTIONS

- `auxiliary` defines an array of one or more auxiliary variable names. There is no limit on the number of auxiliary variables that can be provided on this line; however, lists of information provided in subsequent blocks must have a column of data for each auxiliary variable name defined here. The number of auxiliary variables detected on this line determines the value for naux. Comments cannot be provided anywhere on this line as they will be interpreted as auxiliary variable names. Auxiliary variables may not be used by the package, but they will be available for use by other parts of the program. The program will terminate with an error if auxiliary variables are specified on more than one line in the options block.
- `auxmultname` name of auxiliary variable to be used as multiplier of drain conductance.
- `auxdepthname` name of a variable listed in AUXILIARY that defines the depth at which drainage discharge will be scaled. If a positive value is specified for the AUXDEPTHNAME AUXILIARY variable, then ELEV is the elevation at which the drain starts to discharge and ELEV + DDRN (assuming DDRN is the AUXDEPTHNAME variable) is the elevation when the drain conductance (COND) scaling factor is 1. If a negative drainage depth value is specified for DDRN, then ELEV + DDRN is the elevation at which the drain starts to discharge and ELEV is the elevation when the conductance (COND) scaling factor is 1. A linear- or cubic-scaling is used to scale the drain conductance (COND) when the Standard or Newton-Raphson Formulation is used, respectively.
- `BOUNDNAMES` keyword to indicate that boundary names may be provided with the list of drain cells.

- PRINT_INPUT keyword to indicate that the list of drain information will be written to the listing file immediately after it is read.
- PRINT_FLOWS keyword to indicate that the list of drain flow rates will be printed to the listing file for every stress period time step in which "BUDGET PRINT" is specified in Output Control. If there is no Output Control option and "PRINT_FLOWS" is specified, then flow rates are printed for the last time step of each stress period.
- SAVE_FLOWS keyword to indicate that drain flow terms will be written to the file specified with "BUDGET FILEOUT" in Output Control.
- TS6 keyword to specify that record corresponds to a time-series file.
- FILEIN keyword to specify that an input filename is expected next.
- ts6_filename defines a time-series file defining time series that can be used to assign time-varying values. See the "Time-Variable Input" section for instructions on using the time-series capability.
- OBS6 keyword to specify that record corresponds to an observations file.
- obs6_filename name of input file to define observations for the Drain package. See the "Observation utility" section for instructions for preparing observation input files. Tables ref{table:gwf-obstypetable} and ref{table:gwt-obstypetable} lists observation type(s) supported by the Drain package.
- MOVER keyword to indicate that this instance of the Drain Package can be used with the Water Mover (MVR) Package. When the MOVER option is specified, additional memory is allocated within the package to store the available, provided, and received water.

Block: DIMENSIONS

- maxbound integer value specifying the maximum number of drains cells that will be specified for use during any stress period.

Block: PERIOD

- iper integer value specifying the starting stress period number for which the data specified in the PERIOD block apply. IPER must be less than or equal to NPER in the TDIS Package and greater than zero. The IPER value assigned to a stress period block must be greater than the IPER value assigned for the previous PERIOD block. The information specified in the PERIOD block will continue to apply for all subsequent stress periods, unless the program encounters another PERIOD block.
- cellid is the cell identifier, and depends on the type of grid that is used for the simulation. For a structured grid that uses the DIS input file, CELLID is the layer, row, and column. For a grid that uses the DISV input file, CELLID is the layer and CELL2D number. If the model uses the unstructured discretization (DISU) input file, CELLID is the node number for the cell.
- elev is the elevation of the drain. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- cond is the hydraulic conductance of the interface between the aquifer and the drain. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- aux represents the values of the auxiliary variables for each drain. The values of auxiliary variables must be present for each drain. The values must be specified in the order of the auxiliary variables specified in the OPTIONS block. If the package supports time series and the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.

- `boundname` name of the drain cell. BOUNDNAME is an ASCII character variable that can contain as many as 40 characters. If BOUNDNAME contains spaces in it, then the entire name must be enclosed within single quotes.

Example Input File

```
#The OPTIONS block is optional
BEGIN OPTIONS
  BOUNDNAMES
  PRINT_INPUT
  PRINT_FLOWS
  SAVE_FLOWS
END OPTIONS

#The DIMENSIONS block is required
BEGIN DIMENSIONS
  MAXBOUND 5
END DIMENSIONS

#The following block of drains will be activated for for the entire stress period
BEGIN PERIOD 1
  #node elevation conductance boundname
        73        10.2             1000.        my_drn
        76        10.2             1000.        my_drn
        79        10.2             1000.        my_drn
        80        10.2             1000.        my_drn
        81        10.2             1000.        my_drn
END PERIOD
```

Available Observation Types

Example Observation Input File

```
BEGIN OPTIONS
  DIGITS 8
  PRINT_INPUT
END OPTIONS

BEGIN CONTINUOUS FILEOUT my_model.drn01.csv
# obsname      obstype   ID
  drn_73       DRN       73
  drn_79       DRN       79
END CONTINUOUS

BEGIN CONTINUOUS FILEOUT my_model.drn02.csv
# obsname     obstype   ID
  drn_80      DRN       80
  drn_all     DRN       my_drn
END CONTINUOUS
```

1.3.8 GWF-EVT

Structure of Blocks

FOR EACH SIMULATION

```
    BEGIN OPTIONS
      [FIXED_CELL]
      [AUXILIARY <auxiliary(naux)>]
      [AUXMULTNAME <auxmultname>]
      [BOUNDNAMES]
      [PRINT_INPUT]
      [PRINT_FLOWS]
      [SAVE_FLOWS]
      [TS6 FILEIN <ts6_filename>]
      [OBS6 FILEIN <obs6_filename>]
      [SURF_RATE_SPECIFIED]
    END OPTIONS
```

```
    BEGIN DIMENSIONS
      MAXBOUND <maxbound>
      NSEG <nseg>
    END DIMENSIONS
```

FOR ANY STRESS PERIOD

```
    BEGIN PERIOD <iper>
      <cellid(ncelldim)> <surface> <rate> <depth> <pxdp(nseg-1)> <petm(nseg-1)> [
↪<petm0>] [<aux(naux)>] [<boundname>]
      <cellid(ncelldim)> <surface> <rate> <depth> <pxdp(nseg-1)> <petm(nseg-1)> [
↪<petm0>] [<aux(naux)>] [<boundname>]
      ...
    END PERIOD
```

Explanation of Variables

Block: OPTIONS

- FIXED_CELL indicates that evapotranspiration will not be reassigned to a cell underlying the cell specified in the list if the specified cell is inactive.
- auxiliary defines an array of one or more auxiliary variable names. There is no limit on the number of auxiliary variables that can be provided on this line; however, lists of information provided in subsequent blocks must have a column of data for each auxiliary variable name defined here. The number of auxiliary variables detected on this line determines the value for naux. Comments cannot be provided anywhere on this line as they will be interpreted as auxiliary variable names. Auxiliary variables may not be used by the package, but they will be available for use by other parts of the program. The program will terminate with an error if auxiliary variables are specified on more than one line in the options block.
- auxmultname name of auxiliary variable to be used as multiplier of evapotranspiration rate.
- BOUNDNAMES keyword to indicate that boundary names may be provided with the list of evapotranspiration cells.
- PRINT_INPUT keyword to indicate that the list of evapotranspiration information will be written to the listing file immediately after it is read.

- PRINT_FLOWS keyword to indicate that the list of evapotranspiration flow rates will be printed to the listing file for every stress period time step in which "BUDGET PRINT" is specified in Output Control. If there is no Output Control option and "PRINT_FLOWS" is specified, then flow rates are printed for the last time step of each stress period.
- SAVE_FLOWS keyword to indicate that evapotranspiration flow terms will be written to the file specified with "BUDGET FILEOUT" in Output Control.
- TS6 keyword to specify that record corresponds to a time-series file.
- FILEIN keyword to specify that an input filename is expected next.
- ts6_filename defines a time-series file defining time series that can be used to assign time-varying values. See the "Time-Variable Input" section for instructions on using the time-series capability.
- OBS6 keyword to specify that record corresponds to an observations file.
- obs6_filename name of input file to define observations for the Evapotranspiration package. See the "Observation utility" section for instructions for preparing observation input files. Tables ref{table:gwf-obstypetable} and ref{table:gwt-obstypetable} lists observation type(s) supported by the Evapotranspiration package.
- SURF_RATE_SPECIFIED indicates that the proportion of the evapotranspiration rate at the ET surface will be specified as PETM0 in list input.

Block: DIMENSIONS

- maxbound integer value specifying the maximum number of evapotranspiration cells cells that will be specified for use during any stress period.
- nseg number of ET segments. Default is one. When NSEG is greater than 1, PXDP and PETM arrays must be specified NSEG - 1 times each, in order from the uppermost segment down. PXDP defines the extinction-depth proportion at the bottom of a segment. PETM defines the proportion of the maximum ET flux rate at the bottom of a segment.

Block: PERIOD

- iper integer value specifying the starting stress period number for which the data specified in the PERIOD block apply. IPER must be less than or equal to NPER in the TDIS Package and greater than zero. The IPER value assigned to a stress period block must be greater than the IPER value assigned for the previous PERIOD block. The information specified in the PERIOD block will continue to apply for all subsequent stress periods, unless the program encounters another PERIOD block.
- cellid is the cell identifier, and depends on the type of grid that is used for the simulation. For a structured grid that uses the DIS input file, CELLID is the layer, row, and column. For a grid that uses the DISV input file, CELLID is the layer and CELL2D number. If the model uses the unstructured discretization (DISU) input file, CELLID is the node number for the cell.
- surface is the elevation of the ET surface (L). If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- rate is the maximum ET flux rate (LT-1). If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- depth is the ET extinction depth (L). If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.

- pxdp is the proportion of the ET extinction depth at the bottom of a segment (dimensionless). If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- petm is the proportion of the maximum ET flux rate at the bottom of a segment (dimensionless). If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- petm0 is the proportion of the maximum ET flux rate that will apply when head is at or above the ET surface (dimensionless). PETM0 is read only when the SURF_RATE_SPECIFIED option is used. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- aux represents the values of the auxiliary variables for each evapotranspiration. The values of auxiliary variables must be present for each evapotranspiration. The values must be specified in the order of the auxiliary variables specified in the OPTIONS block. If the package supports time series and the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- boundname name of the evapotranspiration cell. BOUNDNAME is an ASCII character variable that can contain as many as 40 characters. If BOUNDNAME contains spaces in it, then the entire name must be enclosed within single quotes.

Example Input File

Example 1

```
# Example for structured model with list-based input
BEGIN OPTIONS
  AUXNAMES Mult
  BOUNDNAMES
  TS6 FILEIN  EtRate.ts
  # Note: Time-series file EtRate.ts defines time series et_rate
  AUXMULTNAME  Mult
PRINT_INPUT

BEGIN DIMENSIONS
  MAXBOUND  10
END DIMENSIONS

BEGIN PERIOD 1
# Lay Row Col SURFACE  RATE   DEPTH PXPD1 PXPD2 PETM1 PETM2 Mult  Name
    1   1  13  110.0  et_rate  10.0  0.2   0.5   0.3   0.1   0.2  ET-1
    1   2  13  110.0  et_rate  10.0  0.2   0.5   0.3   0.1   0.4  ET-2
    1   3  13  110.0  et_rate  10.0  0.2   0.5   0.3   0.1   0.6  ET-3
    1   4  13  110.0  et_rate  10.0  0.2   0.5   0.3   0.1   0.8  ET-4
    1   5  13  110.0  2.e-2    10.0  0.2   0.5   0.3   0.1   1.0  ET-5
    1   6  13  110.0  2.e-2    10.0  0.2   0.5   0.3   0.1   1.0  ET-6
    1   7  13  110.0  2.e-2    10.0  0.2   0.5   0.3   0.1   0.7  ET-7
    1   8  13  110.0  2.e-2    10.0  0.2   0.5   0.3   0.1   0.5  ET-8
    1   9  13  110.0  2.e-2    10.0  0.2   0.5   0.3   0.1   0.3  ET-9
    1  10  13  110.0  et_rate  10.0  0.2   0.5   0.3   0.1   0.1  ET-10
END PERIOD
```

Example 2

```
  BEGIN OPTIONS
    READASARRAYS
    AUXILIARY var1 var2
    PRINT_INPUT
    PRINT_FLOWS
    SAVE_FLOWS
  END OPTIONS

  BEGIN PERIOD 1
    #For a structured grid, IEVT defaults to model
    # layer 1, so no need to enter IEVT here.

    #ET surface elevation
    SURFACE
      constant 150.0
    #Maximum ET rate
    RATE
      constant 0.007
    #ET extinction depth
    DEPTH
      constant 15.0
    #auxiliary variable (var1) array
    var1
      constant 100.0
    #auxiliary variable (var2) array
    var2
      constant 0.0
  END PERIOD
```

Available Observation Types

Example Observation Input File

```
  BEGIN OPTIONS
    PRINT_INPUT
  END OPTIONS

  BEGIN CONTINUOUS FILEOUT my_model.evt.csv
    et1-1    EVT   1  1  1
    et1-2    EVT   1  1  2
    et2-1    EVT   1  2  1
    et2-2    EVT   1  2  2
    et2-3    EVT   1  2  3
  END CONTINUOUS
```

1.3.9 GWF-EVTA

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  READASARRAYS
  [FIXED_CELL]
  [AUXILIARY <auxiliary(naux)>]
  [AUXMULTNAME <auxmultname>]
  [PRINT_INPUT]
  [PRINT_FLOWS]
  [SAVE_FLOWS]
  [TAS6 FILEIN <tas6_filename>]
  [OBS6 FILEIN <obs6_filename>]
END OPTIONS
```

FOR ANY STRESS PERIOD

```
BEGIN PERIOD <iper>
  [IEVT
        <ievt(ncol*nrow; ncpl)> -- READARRAY]
  SURFACE
        <surface(ncol*nrow; ncpl)> -- READARRAY
  RATE
        <rate(ncol*nrow; ncpl)> -- READARRAY
  DEPTH
        <depth(ncol*nrow; ncpl)> -- READARRAY
  AUX(IAUX)
        <aux(iaux)(ncol*nrow; ncpl)> -- READARRAY
END PERIOD
```

Explanation of Variables

Block: OPTIONS

- READASARRAYS indicates that array-based input will be used for the Evapotranspiration Package. This keyword must be specified to use array-based input.
- FIXED_CELL indicates that evapotranspiration will not be reassigned to a cell underlying the cell specified in the list if the specified cell is inactive.
- auxiliary defines an array of one or more auxiliary variable names. There is no limit on the number of auxiliary variables that can be provided on this line; however, lists of information provided in subsequent blocks must have a column of data for each auxiliary variable name defined here. The number of auxiliary variables detected on this line determines the value for naux. Comments cannot be provided anywhere on this line as they will be interpreted as auxiliary variable names. Auxiliary variables may not be used by the package, but they will be available for use by other parts of the program. The program will terminate with an error if auxiliary variables are specified on more than one line in the options block.
- auxmultname name of auxiliary variable to be used as multiplier of evapotranspiration rate.
- PRINT_INPUT keyword to indicate that the list of evapotranspiration information will be written to the listing file immediately after it is read.

- PRINT_FLOWS keyword to indicate that the list of evapotranspiration flow rates will be printed to the listing file for every stress period time step in which "BUDGET PRINT" is specified in Output Control. If there is no Output Control option and "PRINT_FLOWS" is specified, then flow rates are printed for the last time step of each stress period.
- SAVE_FLOWS keyword to indicate that evapotranspiration flow terms will be written to the file specified with "BUDGET FILEOUT" in Output Control.
- TAS6 keyword to specify that record corresponds to a time-array-series file.
- FILEIN keyword to specify that an input filename is expected next.
- tas6_filename defines a time-array-series file defining a time-array series that can be used to assign time-varying values. See the Time-Variable Input section for instructions on using the time-array series capability.
- OBS6 keyword to specify that record corresponds to an observations file.
- obs6_filename name of input file to define observations for the Evapotranspiration package. See the "Observation utility" section for instructions for preparing observation input files. Tables ref{table:gwf-obstypetable} and ref{table:gwt-obstypetable} lists observation type(s) supported by the Evapotranspiration package.

Block: PERIOD

- iper integer value specifying the starting stress period number for which the data specified in the PERIOD block apply. IPER must be less than or equal to NPER in the TDIS Package and greater than zero. The IPER value assigned to a stress period block must be greater than the IPER value assigned for the previous PERIOD block. The information specified in the PERIOD block will continue to apply for all subsequent stress periods, unless the program encounters another PERIOD block.
- ievt IEVT is the layer number that defines the layer in each vertical column where evapotranspiration is applied. If IEVT is omitted, evapotranspiration by default is applied to cells in layer 1. If IEVT is specified, it must be specified as the first variable in the PERIOD block or MODFLOW will terminate with an error.
- surface is the elevation of the ET surface (L).
- rate is the maximum ET flux rate (LT-1).
- depth is the ET extinction depth (L).
- aux(iaux) is an array of values for auxiliary variable AUX(IAUX), where iaux is a value from 1 to NAUX, and AUX(IAUX) must be listed as part of the auxiliary variables. A separate array can be specified for each auxiliary variable. If an array is not specified for an auxiliary variable, then a value of zero is assigned. If the value specified here for the auxiliary variable is the same as auxmultname, then the evapotranspiration rate will be multiplied by this array.

Example Input File

```
BEGIN OPTIONS
  READASARRAYS
  AUXILIARY var1 var2
  PRINT_INPUT
  PRINT_FLOWS
  SAVE_FLOWS
END OPTIONS

BEGIN PERIOD 1
  #For a structured grid, IEVT defaults to model
```

(continues on next page)

(continued from previous page)

```
  # layer 1, so no need to enter IEVT here.

  #ET surface elevation
  SURFACE
    constant 150.0
  #Maximum ET rate
  RATE
    constant 0.007
  #ET extinction depth
  DEPTH
    constant 15.0
  #auxiliary variable (var1) array
  var1
    constant 100.0
  #auxiliary variable (var2) array
  var2
    constant 0.0
END PERIOD
```

1.3.10 GWF-GHB

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [AUXILIARY <auxiliary(naux)>]
  [AUXMULTNAME <auxmultname>]
  [BOUNDNAMES]
  [PRINT_INPUT]
  [PRINT_FLOWS]
  [SAVE_FLOWS]
  [TS6 FILEIN <ts6_filename>]
  [OBS6 FILEIN <obs6_filename>]
  [MOVER]
END OPTIONS
```

```
BEGIN DIMENSIONS
  MAXBOUND <maxbound>
END DIMENSIONS
```

FOR ANY STRESS PERIOD

```
BEGIN PERIOD <iper>
  <cellid(ncelldim)> <bhead> <cond> [<aux(naux)>] [<boundname>]
  <cellid(ncelldim)> <bhead> <cond> [<aux(naux)>] [<boundname>]
  ...
END PERIOD
```

Explanation of Variables

Block: OPTIONS

- `auxiliary` defines an array of one or more auxiliary variable names. There is no limit on the number of auxiliary variables that can be provided on this line; however, lists of information provided in subsequent blocks must have a column of data for each auxiliary variable name defined here. The number of auxiliary variables detected on this line determines the value for naux. Comments cannot be provided anywhere on this line as they will be interpreted as auxiliary variable names. Auxiliary variables may not be used by the package, but they will be available for use by other parts of the program. The program will terminate with an error if auxiliary variables are specified on more than one line in the options block.
- `auxmultname` name of auxiliary variable to be used as multiplier of general-head boundary conductance.
- `BOUNDNAMES` keyword to indicate that boundary names may be provided with the list of general-head boundary cells.
- `PRINT_INPUT` keyword to indicate that the list of general-head boundary information will be written to the listing file immediately after it is read.
- `PRINT_FLOWS` keyword to indicate that the list of general-head boundary flow rates will be printed to the listing file for every stress period time step in which "BUDGET PRINT" is specified in Output Control. If there is no Output Control option and "PRINT_FLOWS" is specified, then flow rates are printed for the last time step of each stress period.
- `SAVE_FLOWS` keyword to indicate that general-head boundary flow terms will be written to the file specified with "BUDGET FILEOUT" in Output Control.
- `TS6` keyword to specify that record corresponds to a time-series file.
- `FILEIN` keyword to specify that an input filename is expected next.
- `ts6_filename` defines a time-series file defining time series that can be used to assign time-varying values. See the "Time-Variable Input" section for instructions on using the time-series capability.
- `OBS6` keyword to specify that record corresponds to an observations file.
- `obs6_filename` name of input file to define observations for the General-Head Boundary package. See the "Observation utility" section for instructions for preparing observation input files. Tables ref{table:gwf-obstypetable} and ref{table:gwt-obstypetable} lists observation type(s) supported by the General-Head Boundary package.
- `MOVER` keyword to indicate that this instance of the General-Head Boundary Package can be used with the Water Mover (MVR) Package. When the MOVER option is specified, additional memory is allocated within the package to store the available, provided, and received water.

Block: DIMENSIONS

- `maxbound` integer value specifying the maximum number of general-head boundary cells that will be specified for use during any stress period.

Block: PERIOD

- `iper` integer value specifying the starting stress period number for which the data specified in the PERIOD block apply. IPER must be less than or equal to NPER in the TDIS Package and greater than zero. The IPER value assigned to a stress period block must be greater than the IPER value assigned for the previous PERIOD block. The information specified in the PERIOD block will continue to apply for all subsequent stress periods, unless the program encounters another PERIOD block.
- `cellid` is the cell identifier, and depends on the type of grid that is used for the simulation. For a structured grid that uses the DIS input file, CELLID is the layer, row, and column. For a grid that uses the DISV input file, CELLID is the layer and CELL2D number. If the model uses the unstructured discretization (DISU) input file, CELLID is the node number for the cell.
- `bhead` is the boundary head. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `cond` is the hydraulic conductance of the interface between the aquifer cell and the boundary. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `aux` represents the values of the auxiliary variables for each general-head boundary. The values of auxiliary variables must be present for each general-head boundary. The values must be specified in the order of the auxiliary variables specified in the OPTIONS block. If the package supports time series and the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `boundname` name of the general-head boundary cell. BOUNDNAME is an ASCII character variable that can contain as many as 40 characters. If BOUNDNAME contains spaces in it, then the entire name must be enclosed within single quotes.

Example Input File

```
BEGIN OPTIONS
  PRINT_INPUT (echo input to listing file)
  PRINT_FLOWS  (print the flows to the listing file)
  TS6 FILEIN tides.ts
  BOUNDNAMES
END OPTIONS

# Dimensions block
BEGIN DIMENSIONS
  MAXBOUND 15
END DIMENSIONS

# Stress period block(s)
BEGIN PERIOD 1
#Lay Row  Col  Bhead Cond    boundname
  2   1   10   tides   15.0    Estuary-L2
  2   2   10   tides   15.0    Estuary-L2
  2   3   10   tides   15.0    Estuary-L2
  2   4   10   tides   15.0    Estuary-L2
  2   5   10   tides   15.0    Estuary-L2
  2   6   10   tides   15.0    Estuary-L2
  2   7   10   tides   15.0    Estuary-L2
  2   8   10   tides   15.0    Estuary-L2
```

(continues on next page)

(continued from previous page)

```
  2   9   10   tides   15.0    Estuary-L2
  2  10   10   tides   15.0    Estuary-L2
  2  11   10   tides   15.0    Estuary-L2
  2  12   10   tides   15.0    Estuary-L2
  2  13   10   tides   15.0    Estuary-L2
  2  14   10   tides   15.0    Estuary-L2
  2  15   10   tides   15.0    Estuary-L2
END PERIOD
```

Available Observation Types

Example Observation Input File

```
BEGIN OPTIONS
  DIGITS 7
  PRINT_INPUT
END OPTIONS

BEGIN CONTINUOUS FILEOUT my_model.ghb.obs.csv
# obsname      obstype  ID
  ghb-2-6-10   GHB      2   6   10
  ghb-2-7-10   GHB      2   7   10
END CONTINUOUS

BEGIN CONTINUOUS FILEOUT my_model.ghb.flows.csv
# obsname        obstype  ID
  Estuary2       GHB      Estuary-L2
END CONTINUOUS
```

1.3.11 GWF-GNC

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [PRINT_INPUT]
  [PRINT_FLOWS]
  [EXPLICIT]
END OPTIONS
```

```
BEGIN DIMENSIONS
  NUMGNC <numgnc>
  NUMALPHAJ <numalphaj>
END DIMENSIONS
```

```
BEGIN GNCDATA
  <cellidn> <cellidm> <cellidsj(numalphaj)> <alphasj(numalphaj)>
  <cellidn> <cellidm> <cellidsj(numalphaj)> <alphasj(numalphaj)>
  ...
END GNCDATA
```

Explanation of Variables

Block: OPTIONS

- PRINT_INPUT keyword to indicate that the list of GNC information will be written to the listing file immediately after it is read.
- PRINT_FLOWS keyword to indicate that the list of GNC flow rates will be printed to the listing file for every stress period time step in which "BUDGET PRINT" is specified in Output Control. If there is no Output Control option and "PRINT_FLOWS" is specified, then flow rates are printed for the last time step of each stress period.
- EXPLICIT keyword to indicate that the ghost node correction is applied in an explicit manner on the right-hand side of the matrix. The explicit approach will likely require additional outer iterations. If the keyword is not specified, then the correction will be applied in an implicit manner on the left-hand side. The implicit approach will likely converge better, but may require additional memory. If the EXPLICIT keyword is not specified, then the BICGSTAB linear acceleration option should be specified within the LINEAR block of the Sparse Matrix Solver.

Block: DIMENSIONS

- numgnc is the number of GNC entries.
- numalphaj is the number of contributing factors.

Block: GNCDATA

- cellidn is the cellid of the cell, n, in which the ghost node is located. For a structured grid that uses the DIS input file, CELLIDN is the layer, row, and column numbers of the cell. For a grid that uses the DISV input file, CELLIDN is the layer number and CELL2D number for the two cells. If the model uses the unstructured discretization (DISU) input file, then CELLIDN is the node number for the cell.
- cellidm is the cellid of the connecting cell, m, to which flow occurs from the ghost node. For a structured grid that uses the DIS input file, CELLIDM is the layer, row, and column numbers of the cell. For a grid that uses the DISV input file, CELLIDM is the layer number and CELL2D number for the two cells. If the model uses the unstructured discretization (DISU) input file, then CELLIDM is the node number for the cell.
- cellidsj is the array of CELLIDS for the contributing j cells, which contribute to the interpolated head value at the ghost node. This item contains one CELLID for each of the contributing cells of the ghost node. Note that if the number of actual contributing cells needed by the user is less than NUMALPHAJ for any ghost node, then a dummy CELLID of zero(s) should be inserted with an associated contributing factor of zero. For a structured grid that uses the DIS input file, CELLID is the layer, row, and column numbers of the cell. For a grid that uses the DISV input file, CELLID is the layer number and cell2d number for the two cells. If the model uses the unstructured discretization (DISU) input file, then CELLID is the node number for the cell.
- alphasj is the contributing factors for each contributing node in CELLIDSJ. Note that if the number of actual contributing cells is less than NUMALPHAJ for any ghost node, then dummy CELLIDS should be inserted with an associated contributing factor of zero. The sum of ALPHASJ should be less than one. This is because one minus the sum of ALPHASJ is equal to the alpha term (alpha n in equation 4-61 of the GWF Model report) that is multiplied by the head in cell n.

Example Input File

```
BEGIN OPTIONS
  PRINT_INPUT
  PRINT_FLOWS
END OPTIONS

BEGIN DIMENSIONS
  NUMGNC 24
  NUMALPHAJ 1
END DIMENSIONS

BEGIN GNCDATA
  10 41 9  0.333333333333
  10 43 11  0.333333333333
  11 44 10  0.333333333333
  11 46 12  0.333333333333
  12 47 11  0.333333333333
  12 49 13  0.333333333333
  16 41 9  0.333333333333
  16 59 20  0.333333333333
  17 49 13  0.333333333333
  17 67 21  0.333333333333
  20 68 16  0.333333333333
  20 86 24  0.333333333333
  21 76 17  0.333333333333
  21 94 25  0.333333333333
  24 95 20  0.333333333333
  24 113 28  0.333333333333
  25 103 21  0.333333333333
  25 121 32  0.333333333333
  29 113 28  0.333333333333
  29 115 30  0.333333333333
  30 116 29  0.333333333333
  30 118 31  0.333333333333
  31 119 30  0.333333333333
  31 121 32  0.333333333333
END GNCDATA
```

1.3.12 GWF-HFB

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [PRINT_INPUT]
END OPTIONS
```

```
BEGIN DIMENSIONS
  MAXHFB <maxhfb>
END DIMENSIONS
```

FOR ANY STRESS PERIOD

```
BEGIN PERIOD <iper>
  <cellid1(ncelldim)> <cellid2(ncelldim)> <hydchr>
  <cellid1(ncelldim)> <cellid2(ncelldim)> <hydchr>
  ...
END PERIOD
```

Explanation of Variables

Block: OPTIONS

- PRINT_INPUT keyword to indicate that the list of horizontal flow barriers will be written to the listing file immediately after it is read.

Block: DIMENSIONS

- maxhfb integer value specifying the maximum number of horizontal flow barriers that will be entered in this input file. The value of MAXHFB is used to allocate memory for the horizontal flow barriers.

Block: PERIOD

- iper integer value specifying the starting stress period number for which the data specified in the PERIOD block apply. IPER must be less than or equal to NPER in the TDIS Package and greater than zero. The IPER value assigned to a stress period block must be greater than the IPER value assigned for the previous PERIOD block. The information specified in the PERIOD block will continue to apply for all subsequent stress periods, unless the program encounters another PERIOD block.
- cellid1 identifier for the first cell. For a structured grid that uses the DIS input file, CELLID1 is the layer, row, and column numbers of the cell. For a grid that uses the DISV input file, CELLID1 is the layer number and CELL2D number for the two cells. If the model uses the unstructured discretization (DISU) input file, then CELLID1 is the node numbers for the cell. The barrier is located between cells designated as CELLID1 and CELLID2. For models that use the DIS and DISV grid types, the layer number for CELLID1 and CELLID2 must be the same. For all grid types, cells must be horizontally adjacent or the program will terminate with an error.
- cellid2 identifier for the second cell. See CELLID1 for description of how to specify.
- hydchr is the hydraulic characteristic of the horizontal-flow barrier. The hydraulic characteristic is the barrier hydraulic conductivity divided by the width of the horizontal-flow barrier. If the hydraulic characteristic is negative, then the absolute value of HYDCHR acts as a multiplier to the conductance between the two model cells specified as containing the barrier. For example, if the value for HYDCHR was specified as -1.5, the conductance calculated for the two cells would be multiplied by 1.5.

Example Input File

```
BEGIN OPTIONS
  PRINT_INPUT
END OPTIONS

BEGIN DIMENSIONS
  MAXHFB 1
END DIMENSIONS

BEGIN PERIOD 1
  #L1 R1 C1 L2 R2 C2 HYDCHR
    1  1  4  1  1  5 0.1
END PERIOD 1
```

1.3.13 GWF-IC

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN GRIDDATA
  STRT [LAYERED]
       <strt(nodes)> -- READARRAY
END GRIDDATA
```

Explanation of Variables

Block: GRIDDATA

- `strt` is the initial (starting) head—that is, head at the beginning of the GWF Model simulation. STRT must be specified for all simulations, including steady-state simulations. One value is read for every model cell. For simulations in which the first stress period is steady state, the values used for STRT generally do not affect the simulation (exceptions may occur if cells go dry and (or) rewet). The execution time, however, will be less if STRT includes hydraulic heads that are close to the steady-state solution. A head value lower than the cell bottom can be provided if a cell should start as dry.

Example Input File

```
#The OPTIONS block is optional
BEGIN OPTIONS
END OPTIONS

#The GRIDDATA block is required
BEGIN GRIDDATA
  STRT LAYERED
    CONSTANT   0.0  Initial Head layer   1
    CONSTANT   0.0  Initial Head layer   2
END GRIDDATA
```

1.3.14 GWF-LAK

Structure of Blocks

FOR EACH SIMULATION

```
    BEGIN OPTIONS
      [AUXILIARY <auxiliary(naux)>]
      [BOUNDNAMES]
      [PRINT_INPUT]
      [PRINT_STAGE]
      [PRINT_FLOWS]
      [SAVE_FLOWS]
      [STAGE FILEOUT <stagefile>]
      [BUDGET FILEOUT <budgetfile>]
      [PACKAGE_CONVERGENCE FILEOUT <package_convergence_filename>]
      [TS6 FILEIN <ts6_filename>]
      [OBS6 FILEIN <obs6_filename>]
      [MOVER]
      [SURFDEP <surfdep>]
      [TIME_CONVERSION <time_conversion>]
      [LENGTH_CONVERSION <length_conversion>]
    END OPTIONS
```

```
    BEGIN DIMENSIONS
      NLAKES <nlakes>
      NOUTLETS <noutlets>
      NTABLES <ntables>
    END DIMENSIONS
```

```
    BEGIN PACKAGEDATA
      <lakeno> <strt> <nlakeconn> [<aux(naux)>] [<boundname>]
      <lakeno> <strt> <nlakeconn> [<aux(naux)>] [<boundname>]
      ...
    END PACKAGEDATA
```

```
    BEGIN CONNECTIONDATA
      <lakeno> <iconn> <cellid(ncelldim)> <claktype> <bedleak> <belev> <telev>
↪<connlen> <connwidth>
      <lakeno> <iconn> <cellid(ncelldim)> <claktype> <bedleak> <belev> <telev>
↪<connlen> <connwidth>
      ...
    END CONNECTIONDATA
```

```
    BEGIN TABLES
      <lakeno> TAB6 FILEIN <tab6_filename>
      <lakeno> TAB6 FILEIN <tab6_filename>
      ...
    END TABLES
```

```
    BEGIN OUTLETS
      <outletno> <lakein> <lakeout> <couttype> <invert> <width> <rough> <slope>
      <outletno> <lakein> <lakeout> <couttype> <invert> <width> <rough> <slope>
      ...
    END OUTLETS
```

FOR ANY STRESS PERIOD

```
BEGIN PERIOD <iper>
  <number> <laksetting>
  <number> <laksetting>
  ...
END PERIOD
```

Explanation of Variables

Block: OPTIONS

- `auxiliary` defines an array of one or more auxiliary variable names. There is no limit on the number of auxiliary variables that can be provided on this line; however, lists of information provided in subsequent blocks must have a column of data for each auxiliary variable name defined here. The number of auxiliary variables detected on this line determines the value for naux. Comments cannot be provided anywhere on this line as they will be interpreted as auxiliary variable names. Auxiliary variables may not be used by the package, but they will be available for use by other parts of the program. The program will terminate with an error if auxiliary variables are specified on more than one line in the options block.
- `BOUNDNAMES` keyword to indicate that boundary names may be provided with the list of lake cells.
- `PRINT_INPUT` keyword to indicate that the list of lake information will be written to the listing file immediately after it is read.
- `PRINT_STAGE` keyword to indicate that the list of lake stages will be printed to the listing file for every stress period in which "HEAD PRINT" is specified in Output Control. If there is no Output Control option and PRINT_STAGE is specified, then stages are printed for the last time step of each stress period.
- `PRINT_FLOWS` keyword to indicate that the list of lake flow rates will be printed to the listing file for every stress period time step in which "BUDGET PRINT" is specified in Output Control. If there is no Output Control option and "PRINT_FLOWS" is specified, then flow rates are printed for the last time step of each stress period.
- `SAVE_FLOWS` keyword to indicate that lake flow terms will be written to the file specified with "BUDGET FILEOUT" in Output Control.
- `STAGE` keyword to specify that record corresponds to stage.
- `stagefile` name of the binary output file to write stage information.
- `BUDGET` keyword to specify that record corresponds to the budget.
- `FILEOUT` keyword to specify that an output filename is expected next.
- `budgetfile` name of the binary output file to write budget information.
- `PACKAGE_CONVERGENCE` keyword to specify that record corresponds to the package convergence comma spaced values file.
- `package_convergence_filename` name of the comma spaced values output file to write package convergence information.
- `TS6` keyword to specify that record corresponds to a time-series file.
- `FILEIN` keyword to specify that an input filename is expected next.
- `ts6_filename` defines a time-series file defining time series that can be used to assign time-varying values. See the "Time-Variable Input" section for instructions on using the time-series capability.
- `OBS6` keyword to specify that record corresponds to an observations file.

- obs6_filename name of input file to define observations for the LAK package. See the "Observation utility" section for instructions for preparing observation input files. Tables ref{table:gwf-obstypetable} and ref{table:gwt-obstypetable} lists observation type(s) supported by the LAK package.
- MOVER keyword to indicate that this instance of the LAK Package can be used with the Water Mover (MVR) Package. When the MOVER option is specified, additional memory is allocated within the package to store the available, provided, and received water.
- surfdep real value that defines the surface depression depth for VERTICAL lake-GWF connections. If specified, SURFDEP must be greater than or equal to zero. If SURFDEP is not specified, a default value of zero is used for all vertical lake-GWF connections.
- time_conversion value that is used in converting outlet flow terms that use Manning's equation or gravitational acceleration to consistent time units. TIME_CONVERSION should be set to 1.0, 60.0, 3,600.0, 86,400.0, and 31,557,600.0 when using time units (TIME_UNITS) of seconds, minutes, hours, days, or years in the simulation, respectively. CONVTIME does not need to be specified if no lake outlets are specified or TIME_UNITS are seconds.
- length_conversion real value that is used in converting outlet flow terms that use Manning's equation or gravitational acceleration to consistent length units. LENGTH_CONVERSION should be set to 3.28081, 1.0, and 100.0 when using length units (LENGTH_UNITS) of feet, meters, or centimeters in the simulation, respectively. LENGTH_CONVERSION does not need to be specified if no lake outlets are specified or LENGTH_UNITS are meters.

Block: DIMENSIONS

- nlakes value specifying the number of lakes that will be simulated for all stress periods.
- noutlets value specifying the number of outlets that will be simulated for all stress periods. If NOUTLETS is not specified, a default value of zero is used.
- ntables value specifying the number of lakes tables that will be used to define the lake stage, volume relation, and surface area. If NTABLES is not specified, a default value of zero is used.

Block: PACKAGEDATA

- lakeno integer value that defines the lake number associated with the specified PACKAGEDATA data on the line. LAKENO must be greater than zero and less than or equal to NLAKES. Lake information must be specified for every lake or the program will terminate with an error. The program will also terminate with an error if information for a lake is specified more than once.
- strt real value that defines the starting stage for the lake.
- nlakeconn integer value that defines the number of GWF cells connected to this (LAKENO) lake. There can only be one vertical lake connection to each GWF cell. NLAKECONN must be greater than zero.
- aux represents the values of the auxiliary variables for each lake. The values of auxiliary variables must be present for each lake. The values must be specified in the order of the auxiliary variables specified in the OPTIONS block. If the package supports time series and the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- boundname name of the lake cell. BOUNDNAME is an ASCII character variable that can contain as many as 40 characters. If BOUNDNAME contains spaces in it, then the entire name must be enclosed within single quotes.

Block: CONNECTIONDATA

- `lakeno` integer value that defines the lake number associated with the specified CONNECTIONDATA data on the line. LAKENO must be greater than zero and less than or equal to NLAKES. Lake connection information must be specified for every lake connection to the GWF model (NLAKECONN) or the program will terminate with an error. The program will also terminate with an error if connection information for a lake connection to the GWF model is specified more than once.
- `iconn` integer value that defines the GWF connection number for this lake connection entry. ICONN must be greater than zero and less than or equal to NLAKECONN for lake LAKENO.
- `cellid` is the cell identifier, and depends on the type of grid that is used for the simulation. For a structured grid that uses the DIS input file, CELLID is the layer, row, and column. For a grid that uses the DISV input file, CELLID is the layer and CELL2D number. If the model uses the unstructured discretization (DISU) input file, CELLID is the node number for the cell.
- `claktype` character string that defines the lake-GWF connection type for the lake connection. Possible lake-GWF connection type strings include: VERTICAL–character keyword to indicate the lake-GWF connection is vertical and connection conductance calculations use the hydraulic conductivity corresponding to the K_{33} tensor component defined for CELLID in the NPF package. HORIZONTAL–character keyword to indicate the lake-GWF connection is horizontal and connection conductance calculations use the hydraulic conductivity corresponding to the K_{11} tensor component defined for CELLID in the NPF package. EMBEDDEDH–character keyword to indicate the lake-GWF connection is embedded in a single cell and connection conductance calculations use the hydraulic conductivity corresponding to the K_{11} tensor component defined for CELLID in the NPF package. EMBEDDEDV–character keyword to indicate the lake-GWF connection is embedded in a single cell and connection conductance calculations use the hydraulic conductivity corresponding to the K_{33} tensor component defined for CELLID in the NPF package. Embedded lakes can only be connected to a single cell (NLAKECONN = 1) and there must be a lake table associated with each embedded lake.
- `bedleak` character string or real value that defines the bed leakance for the lake-GWF connection. BEDLEAK must be greater than or equal to zero or specified to be NONE. If BEDLEAK is specified to be NONE, the lake-GWF connection conductance is solely a function of aquifer properties in the connected GWF cell and lakebed sediments are assumed to be absent.
- `belev` real value that defines the bottom elevation for a HORIZONTAL lake-GWF connection. Any value can be specified if CLAKTYPE is VERTICAL, EMBEDDEDH, or EMBEDDEDV. If CLAKTYPE is HORIZONTAL and BELEV is not equal to TELEV, BELEV must be greater than or equal to the bottom of the GWF cell CELLID. If BELEV is equal to TELEV, BELEV is reset to the bottom of the GWF cell CELLID.
- `telev` real value that defines the top elevation for a HORIZONTAL lake-GWF connection. Any value can be specified if CLAKTYPE is VERTICAL, EMBEDDEDH, or EMBEDDEDV. If CLAKTYPE is HORIZONTAL and TELEV is not equal to BELEV, TELEV must be less than or equal to the top of the GWF cell CELLID. If TELEV is equal to BELEV, TELEV is reset to the top of the GWF cell CELLID.
- `connlen` real value that defines the distance between the connected GWF CELLID node and the lake for a HORIZONTAL, EMBEDDEDH, or EMBEDDEDV lake-GWF connection. CONLENN must be greater than zero for a HORIZONTAL, EMBEDDEDH, or EMBEDDEDV lake-GWF connection. Any value can be specified if CLAKTYPE is VERTICAL.
- `connwidth` real value that defines the connection face width for a HORIZONTAL lake-GWF connection. CONNWIDTH must be greater than zero for a HORIZONTAL lake-GWF connection. Any value can be specified if CLAKTYPE is VERTICAL, EMBEDDEDH, or EMBEDDEDV.

Block: TABLES

- `lakeno` integer value that defines the lake number associated with the specified TABLES data on the line. LAKENO must be greater than zero and less than or equal to NLAKES. The program will terminate with an error if table information for a lake is specified more than once or the number of specified tables is less than NTABLES.
- `TAB6` keyword to specify that record corresponds to a table file.
- `FILEIN` keyword to specify that an input filename is expected next.
- `tab6_filename` character string that defines the path and filename for the file containing lake table data for the lake connection. The TAB6_FILENAME file includes the number of entries in the file and the relation between stage, volume, and surface area for each entry in the file. Lake table files for EMBEDDEDH and EMBEDDEDV lake-GWF connections also include lake-GWF exchange area data for each entry in the file. Instructions for creating the TAB6_FILENAME input file are provided in Lake Table Input File section.

Block: OUTLETS

- `outletno` integer value that defines the outlet number associated with the specified OUTLETS data on the line. OUTLETNO must be greater than zero and less than or equal to NOUTLETS. Outlet information must be specified for every outlet or the program will terminate with an error. The program will also terminate with an error if information for a outlet is specified more than once.
- `lakein` integer value that defines the lake number that outlet is connected to. LAKEIN must be greater than zero and less than or equal to NLAKES.
- `lakeout` integer value that defines the lake number that outlet discharge from lake outlet OUTLETNO is routed to. LAKEOUT must be greater than or equal to zero and less than or equal to NLAKES. If LAKEOUT is zero, outlet discharge from lake outlet OUTLETNO is discharged to an external boundary.
- `couttype` character string that defines the outlet type for the outlet OUTLETNO. Possible COUTTYPE strings include: SPECIFIED–character keyword to indicate the outlet is defined as a specified flow. MANNING–character keyword to indicate the outlet is defined using Manning's equation. WEIR–character keyword to indicate the outlet is defined using a sharp weir equation.
- `invert` real value that defines the invert elevation for the lake outlet. Any value can be specified if COUTTYPE is SPECIFIED. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `width` real value that defines the width of the lake outlet. Any value can be specified if COUTTYPE is SPECIFIED. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `rough` real value that defines the roughness coefficient for the lake outlet. Any value can be specified if COUTTYPE is not MANNING. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `slope` real value that defines the bed slope for the lake outlet. Any value can be specified if COUTTYPE is not MANNING. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.

Block: PERIOD

- `iper` integer value specifying the starting stress period number for which the data specified in the PERIOD block apply. IPER must be less than or equal to NPER in the TDIS Package and greater than zero. The IPER value assigned to a stress period block must be greater than the IPER value assigned for the previous PERIOD block. The information specified in the PERIOD block will continue to apply for all subsequent stress periods, unless the program encounters another PERIOD block.
- `number` integer value that defines the lake or outlet number associated with the specified PERIOD data on the line. NUMBER must be greater than zero and less than or equal to NLAKES for a lake number and less than or equal to NOUTLETS for an outlet number.
- `laksetting` line of information that is parsed into a keyword and values. Keyword values that can be used to start the LAKSETTING string include both keywords for lake settings and keywords for outlet settings. Keywords for lake settings include: STATUS, STAGE, RAINFALL, EVAPORATION, RUNOFF, INFLOW, WITHDRAWAL, and AUXILIARY. Keywords for outlet settings include RATE, INVERT, WIDTH, SLOPE, and ROUGH.

```
STATUS <status>
STAGE <stage>
RAINFALL <rainfall>
EVAPORATION <evaporation>
RUNOFF <runoff>
INFLOW <inflow>
WITHDRAWAL <withdrawal>
RATE <rate>
INVERT <invert>
WIDTH <width>
SLOPE <slope>
ROUGH <rough>
AUXILIARY <auxname> <auxval>
```

- `status` keyword option to define lake status. STATUS can be ACTIVE, INACTIVE, or CONSTANT. By default, STATUS is ACTIVE.
- `stage` real or character value that defines the stage for the lake. The specified STAGE is only applied if the lake is a constant stage lake. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `rainfall` real or character value that defines the rainfall rate (LT-1) for the lake. Value must be greater than or equal to zero. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `evaporation` real or character value that defines the maximum evaporation rate (LT-1) for the lake. Value must be greater than or equal to zero. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `runoff` real or character value that defines the runoff rate (L3 T-1) for the lake. Value must be greater than or equal to zero. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `inflow` real or character value that defines the volumetric inflow rate (L3 T-1) for the lake. Value must be greater than or equal to zero. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value. By default, inflow rates are zero for each lake.

- withdrawal real or character value that defines the maximum withdrawal rate (L3 T-1) for the lake. Value must be greater than or equal to zero. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- rate real or character value that defines the extraction rate for the lake outflow. A positive value indicates inflow and a negative value indicates outflow from the lake. RATE only applies to active (IBOUND > 0) lakes. A specified RATE is only applied if COUTTYPE for the OUTLETNO is SPECIFIED. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value. By default, the RATE for each SPECIFIED lake outlet is zero.
- invert real or character value that defines the invert elevation for the lake outlet. A specified INVERT value is only used for active lakes if COUTTYPE for lake outlet OUTLETNO is not SPECIFIED. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- rough real value that defines the roughness coefficient for the lake outlet. Any value can be specified if COUTTYPE is not MANNING. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- width real or character value that defines the width of the lake outlet. A specified WIDTH value is only used for active lakes if COUTTYPE for lake outlet OUTLETNO is not SPECIFIED. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- slope real or character value that defines the bed slope for the lake outlet. A specified SLOPE value is only used for active lakes if COUTTYPE for lake outlet OUTLETNO is MANNING. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- AUXILIARY keyword for specifying auxiliary variable.
- auxname name for the auxiliary variable to be assigned AUXVAL. AUXNAME must match one of the auxiliary variable names defined in the OPTIONS block. If AUXNAME does not match one of the auxiliary variable names defined in the OPTIONS block the data are ignored.
- auxval value for the auxiliary variable. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.

Example Input File

```
BEGIN OPTIONS
  PRINT_INPUT
  BOUNDNAMES
  PRINT_STAGE
  PRINT_FLOWS
  STAGE FILEOUT lak-1.stage.bin
  BUDGET FILEOUT lak-1.cbc
END OPTIONS

BEGIN DIMENSIONS
  NLAKES 1
  NOUTLETS 1
END DIMENSIONS
```

(continues on next page)

(continued from previous page)

```
BEGIN PACKAGEDATA
# lakeno    strt lakeconn  boundname
       1  110.00       57  LAKE_1
END PACKAGEDATA

BEGIN CONNECTIONDATA
# lakeno iconn layer row column       ctype bedleak belev telev  dx width
       1     1     1   7      6  HORIZONTAL     0.1     0     0 500   500
       1     2     1   8      6  HORIZONTAL     0.1     0     0 500   500
       1     3     1   9      6  HORIZONTAL     0.1     0     0 500   500
       1     4     1  10      6  HORIZONTAL     0.1     0     0 500   500
       1     5     1  11      6  HORIZONTAL     0.1     0     0 500   500
       1     6     1   6      7  HORIZONTAL     0.1     0     0 500   500
       1     7     2   7      7    VERTICAL     0.1     0     0   0     0
       1     8     2   8      7    VERTICAL     0.1     0     0   0     0
       1     9     2   8      7  HORIZONTAL     0.1     0     0 250   500
       1    10     2   9      7    VERTICAL     0.1     0     0   0     0
       1    11     2   9      7  HORIZONTAL     0.1     0     0 250   500
       1    12     2  10      7    VERTICAL     0.1     0     0   0     0
       1    13     2  10      7  HORIZONTAL     0.1     0     0 250   500
       1    14     2  11      7    VERTICAL     0.1     0     0   0     0
       1    15     1  12      7  HORIZONTAL     0.1     0     0 500   500
       1    16     1   6      8  HORIZONTAL     0.1     0     0 500   500
       1    17     2   7      8    VERTICAL     0.1     0     0   0     0
       1    18     2   7      8  HORIZONTAL     0.1     0     0 250   500
       1    19     3   8      8    VERTICAL     0.1     0     0   0     0
       1    20     3   9      8    VERTICAL     0.1     0     0   0     0
       1    21     3  10      8    VERTICAL     0.1     0     0   0     0
       1    22     2  11      8    VERTICAL     0.1     0     0   0     0
       1    23     2  11      8  HORIZONTAL     0.1     0     0 250   500
       1    24     1  12      8  HORIZONTAL     0.1     0     0 500   500
       1    25     1   6      9  HORIZONTAL     0.1     0     0 500   500
       1    26     2   7      9    VERTICAL     0.1     0     0   0     0
       1    27     2   7      9  HORIZONTAL     0.1     0     0 250   500
       1    28     3   8      9    VERTICAL     0.1     0     0   0     0
       1    29     3   9      9    VERTICAL     0.1     0     0   0     0
       1    30     3  10      9    VERTICAL     0.1     0     0   0     0
       1    31     2  11      9    VERTICAL     0.1     0     0   0     0
       1    32     2  11      9  HORIZONTAL     0.1     0     0 250   500
       1    33     1  12      9  HORIZONTAL     0.1     0     0 500   500
       1    34     1   6     10  HORIZONTAL     0.1     0     0 500   500
       1    35     2   7     10    VERTICAL     0.1     0     0   0     0
       1    36     2   7     10  HORIZONTAL     0.1     0     0 250   500
       1    37     3   8     10    VERTICAL     0.1     0     0   0     0
       1    38     3   9     10    VERTICAL     0.1     0     0   0     0
       1    39     3  10     10    VERTICAL     0.1     0     0   0     0
       1    40     2  11     10    VERTICAL     0.1     0     0   0     0
       1    41     2  11     10  HORIZONTAL     0.1     0     0 250   500
       1    42     1  12     10  HORIZONTAL     0.1     0     0 500   500
       1    43     1   6     11  HORIZONTAL     0.1     0     0 500   500
       1    44     2   7     11    VERTICAL     0.1     0     0   0     0
       1    45     2   8     11    VERTICAL     0.1     0     0   0     0
       1    46     2   8     11  HORIZONTAL     0.1     0     0 250   500
       1    47     2   9     11    VERTICAL     0.1     0     0   0     0
       1    48     2   9     11  HORIZONTAL     0.1     0     0 250   500
       1    49     2  10     11    VERTICAL     0.1     0     0   0     0
```

(continues on next page)

(continued from previous page)

```
        1    50     2  10     11  HORIZONTAL     0.1     0     0 250   500
        1    51     2  11     11    VERTICAL     0.1     0     0   0     0
        1    52     1  12     11  HORIZONTAL     0.1     0     0 500   500
        1    53     1   7     12  HORIZONTAL     0.1     0     0 500   500
        1    54     1   8     12  HORIZONTAL     0.1     0     0 500   500
        1    55     1   9     12  HORIZONTAL     0.1     0     0 500   500
        1    56     1  10     12  HORIZONTAL     0.1     0     0 500   500
        1    57     1  11     12  HORIZONTAL     0.1     0     0 500   500
END CONNECTIONDATA

BEGIN OUTLETS
# outletno  lakein lakeout    couttype  invert   width   rough   slope
         1       1       0   SPECIFIED       0       0       0       0
END OUTLETS

BEGIN PERIOD 1
  1 RAINFALL 0.0116
  1 EVAPORATION 0.0103
END PERIOD

BEGIN PERIOD 100
  1 STATUS CONSTANT
  1 STAGE 110.
END PERIOD
```

Available Observation Types

Example Observation Input File

```
BEGIN OPTIONS
  DIGITS 10
  PRINT_INPUT
END OPTIONS

BEGIN CONTINUOUS FILEOUT my_model.lak.csv
  l1stage  stage  1
  l1vol    volume 1
  vflow    lak    1 1
  hflow1   lak    1 2
  hflow2   lak    1 3
  hflow3   lak    1 4
  hflow4   lak    1 5
  lakflow  lak    lake_1
END CONTINUOUS
```

1.3.15 GWF-MAW

Structure of Blocks

FOR EACH SIMULATION

```
    BEGIN OPTIONS
      [AUXILIARY <auxiliary(naux)>]
      [BOUNDNAMES]
      [PRINT_INPUT]
      [PRINT_HEAD]
      [PRINT_FLOWS]
      [SAVE_FLOWS]
      [HEAD FILEOUT <headfile>]
      [BUDGET FILEOUT <budgetfile>]
      [NO_WELL_STORAGE]
      [FLOW_CORRECTION]
      [FLOWING_WELLS]
      [SHUTDOWN_THETA <shutdown_theta>]
      [SHUTDOWN_KAPPA <shutdown_kappa>]
      [TS6 FILEIN <ts6_filename>]
      [OBS6 FILEIN <obs6_filename>]
      [MOVER]
    END OPTIONS
```

```
    BEGIN DIMENSIONS
      NMAWWELLS <nmawwells>
    END DIMENSIONS
```

```
    BEGIN PACKAGEDATA
      <wellno> <radius> <bottom> <strt> <condeqn> <ngwfnodes> [<aux(naux)>] [
↪<boundname>]
      <wellno> <radius> <bottom> <strt> <condeqn> <ngwfnodes> [<aux(naux)>] [
↪<boundname>]
      ...
    END PACKAGEDATA
```

```
    BEGIN CONNECTIONDATA
      <wellno> <icon> <cellid(ncelldim)> <scrn_top> <scrn_bot> <hk_skin> <radius_skin>
      <wellno> <icon> <cellid(ncelldim)> <scrn_top> <scrn_bot> <hk_skin> <radius_skin>
      ...
    END CONNECTIONDATA
```

FOR ANY STRESS PERIOD

```
    BEGIN PERIOD <iper>
      <wellno> <mawsetting>
      <wellno> <mawsetting>
      ...
    END PERIOD
```

Explanation of Variables

Block: OPTIONS

- `auxiliary` defines an array of one or more auxiliary variable names. There is no limit on the number of auxiliary variables that can be provided on this line; however, lists of information provided in subsequent blocks must have a column of data for each auxiliary variable name defined here. The number of auxiliary variables detected on this line determines the value for naux. Comments cannot be provided anywhere on this line as they will be interpreted as auxiliary variable names. Auxiliary variables may not be used by the package, but they will be available for use by other parts of the program. The program will terminate with an error if auxiliary variables are specified on more than one line in the options block.
- `BOUNDNAMES` keyword to indicate that boundary names may be provided with the list of multi-aquifer well cells.
- `PRINT_INPUT` keyword to indicate that the list of multi-aquifer well information will be written to the listing file immediately after it is read.
- `PRINT_HEAD` keyword to indicate that the list of multi-aquifer well heads will be printed to the listing file for every stress period in which "HEAD PRINT" is specified in Output Control. If there is no Output Control option and PRINT_HEAD is specified, then heads are printed for the last time step of each stress period.
- `PRINT_FLOWS` keyword to indicate that the list of multi-aquifer well flow rates will be printed to the listing file for every stress period time step in which "BUDGET PRINT" is specified in Output Control. If there is no Output Control option and "PRINT_FLOWS" is specified, then flow rates are printed for the last time step of each stress period.
- `SAVE_FLOWS` keyword to indicate that multi-aquifer well flow terms will be written to the file specified with "BUDGET FILEOUT" in Output Control.
- `HEAD` keyword to specify that record corresponds to head.
- `headfile` name of the binary output file to write head information.
- `BUDGET` keyword to specify that record corresponds to the budget.
- `FILEOUT` keyword to specify that an output filename is expected next.
- `budgetfile` name of the binary output file to write budget information.
- `NO_WELL_STORAGE` keyword that deactivates inclusion of well storage contributions to the multi-aquifer well package continuity equation.
- `FLOW_CORRECTION` keyword that activates flow corrections in cases where the head in a multi-aquifer well is below the bottom of the screen for a connection or the head in a convertible cell connected to a multi-aquifer well is below the cell bottom. When flow corrections are activated, unit head gradients are used to calculate the flow between a multi-aquifer well and a connected GWF cell. By default, flow corrections are not made.
- `FLOWING_WELLS` keyword that activates the flowing wells option for the multi-aquifer well package.
- `shutdown_theta` value that defines the weight applied to discharge rate for wells that limit the water level in a discharging well (defined using the HEAD_LIMIT keyword in the stress period data). SHUTDOWN_THETA is used to control discharge rate oscillations when the flow rate from the aquifer is less than the specified flow rate from the aquifer to the well. Values range between 0.0 and 1.0, and larger values increase the weight (decrease under-relaxation) applied to the well discharge rate. The HEAD_LIMIT option has been included to facilitate backward compatibility with previous versions of MODFLOW but use of the RATE_SCALING option instead of the HEAD_LIMIT option is recommended. By default, SHUTDOWN_THETA is 0.7.
- `shutdown_kappa` value that defines the weight applied to discharge rate for wells that limit the water level in a discharging well (defined using the HEAD_LIMIT keyword in the stress period data). SHUTDOWN_KAPPA is used to control discharge rate oscillations when the flow rate from the aquifer is less than the specified flow

rate from the aquifer to the well. Values range between 0.0 and 1.0, and larger values increase the weight applied to the well discharge rate. The HEAD_LIMIT option has been included to facilitate backward compatibility with previous versions of MODFLOW but use of the RATE_SCALING option instead of the HEAD_LIMIT option is recommended. By default, SHUTDOWN_KAPPA is 0.0001.

- `TS6` keyword to specify that record corresponds to a time-series file.
- `FILEIN` keyword to specify that an input filename is expected next.
- `ts6_filename` defines a time-series file defining time series that can be used to assign time-varying values. See the "Time-Variable Input" section for instructions on using the time-series capability.
- `OBS6` keyword to specify that record corresponds to an observations file.
- `obs6_filename` name of input file to define observations for the MAW package. See the "Observation utility" section for instructions for preparing observation input files. Tables ref{table:gwf-obstypetable} and ref{table:gwt-obstypetable} lists observation type(s) supported by the MAW package.
- `MOVER` keyword to indicate that this instance of the MAW Package can be used with the Water Mover (MVR) Package. When the MOVER option is specified, additional memory is allocated within the package to store the available, provided, and received water.

Block: DIMENSIONS

- `nmawwells` integer value specifying the number of multi-aquifer wells that will be simulated for all stress periods.

Block: PACKAGEDATA

- `wellno` integer value that defines the well number associated with the specified PACKAGEDATA data on the line. WELLNO must be greater than zero and less than or equal to NMAWWELLS. Multi-aquifer well information must be specified for every multi-aquifer well or the program will terminate with an error. The program will also terminate with an error if information for a multi-aquifer well is specified more than once.
- `radius` radius for the multi-aquifer well. The program will terminate with an error if the radius is less than or equal to zero.
- `bottom` bottom elevation of the multi-aquifer well. If CONDEQN is SPECIFIED, THIEM, SKIN, or COMPOSITE, BOTTOM is set to the cell bottom in the lowermost GWF cell connection in cases where the specified well bottom is above the bottom of this GWF cell. If CONDEQN is MEAN, BOTTOM is set to the lowermost GWF cell connection screen bottom in cases where the specified well bottom is above this value. The bottom elevation defines the lowest well head that will be simulated when the NEWTON UNDER_RELAXATION option is specified in the GWF model name file. The bottom elevation is also used to calculate volumetric storage in the well.
- `strt` starting head for the multi-aquifer well. The program will terminate with an error if the starting head is less than the specified well bottom.
- `condeqn` character string that defines the conductance equation that is used to calculate the saturated conductance for the multi-aquifer well. Possible multi-aquifer well CONDEQN strings include: SPECIFIED–character keyword to indicate the multi-aquifer well saturated conductance will be specified. THIEM–character keyword to indicate the multi-aquifer well saturated conductance will be calculated using the Thiem equation, which considers the cell top and bottom, aquifer hydraulic conductivity, and effective cell and well radius. SKIN–character keyword to indicate that the multi-aquifer well saturated conductance will be calculated using the cell top and bottom, aquifer and screen hydraulic conductivity, and well and skin radius. CUMULATIVE–character keyword to indicate that the multi-aquifer well saturated conductance will be calculated using a combination of the Thiem and SKIN equations. MEAN–character keyword to indicate the multi-aquifer well saturated conductance

will be calculated using the aquifer and screen top and bottom, aquifer and screen hydraulic conductivity, and well and skin radius. The CUMULATIVE conductance equation is identical to the SKIN LOSSTYPE in the Multi-Node Well (MNW2) package for MODFLOW-2005. The program will terminate with an error condition if CONDEQN is SKIN or CUMULATIVE and the calculated saturated conductance is less than zero; if an error condition occurs, it is suggested that the THEIM or MEAN conductance equations be used for these multi-aquifer wells.

- `ngwfnodes` integer value that defines the number of GWF nodes connected to this (WELLNO) multi-aquifer well. NGWFNODES must be greater than zero.
- `aux` represents the values of the auxiliary variables for each multi-aquifer well. The values of auxiliary variables must be present for each multi-aquifer well. The values must be specified in the order of the auxiliary variables specified in the OPTIONS block. If the package supports time series and the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `boundname` name of the multi-aquifer well cell. BOUNDNAME is an ASCII character variable that can contain as many as 40 characters. If BOUNDNAME contains spaces in it, then the entire name must be enclosed within single quotes.

Block: CONNECTIONDATA

- `wellno` integer value that defines the well number associated with the specified CONNECTIONDATA data on the line. WELLNO must be greater than zero and less than or equal to NMAWWELLS. Multi-aquifer well connection information must be specified for every multi-aquifer well connection to the GWF model (NGWFNODES) or the program will terminate with an error. The program will also terminate with an error if connection information for a multi-aquifer well connection to the GWF model is specified more than once.
- `icon` integer value that defines the GWF connection number for this multi-aquifer well connection entry. ICONN must be greater than zero and less than or equal to NGWFNODES for multi-aquifer well WELLNO.
- `cellid` is the cell identifier, and depends on the type of grid that is used for the simulation. For a structured grid that uses the DIS input file, CELLID is the layer, row, and column. For a grid that uses the DISV input file, CELLID is the layer and CELL2D number. If the model uses the unstructured discretization (DISU) input file, CELLID is the node number for the cell. One or more screened intervals can be connected to the same CELLID if CONDEQN for a well is MEAN. The program will terminate with an error if MAW wells using SPECIFIED, THIEM, SKIN, or CUMULATIVE conductance equations have more than one connection to the same CELLID.
- `scrn_top` value that defines the top elevation of the screen for the multi-aquifer well connection. If CONDEQN is SPECIFIED, THIEM, SKIN, or COMPOSITE, SCRN_TOP can be any value and is set to the top of the cell. If CONDEQN is MEAN, SCRN_TOP is set to the multi-aquifer well connection cell top if the specified value is greater than the cell top. The program will terminate with an error if the screen top is less than the screen bottom.
- `scrn_bot` value that defines the bottom elevation of the screen for the multi-aquifer well connection. If CONDEQN is SPECIFIED, THIEM, SKIN, or COMPOSITE, SCRN_BOT can be any value is set to the bottom of the cell. If CONDEQN is MEAN, SCRN_BOT is set to the multi-aquifer well connection cell bottom if the specified value is less than the cell bottom. The program will terminate with an error if the screen bottom is greater than the screen top.
- `hk_skin` value that defines the skin (filter pack) hydraulic conductivity (if CONDEQN for the multi-aquifer well is SKIN, CUMULATIVE, or MEAN) or conductance (if CONDEQN for the multi-aquifer well is SPECIFIED) for each GWF node connected to the multi-aquifer well (NGWFNODES). If CONDEQN is SPECIFIED, HK_SKIN must be greater than or equal to zero. HK_SKIN can be any value if CONDEQN is THIEM. Otherwise, HK_SKIN must be greater than zero. If CONDEQN is SKIN, the contrast between the cell transmissivity (the product of geometric mean horizontal hydraulic conductivity and the cell thickness) and the well transmissivity (the product of HK_SKIN and the screen thicknesses) must be greater than one in node CELLID or the

program will terminate with an error condition; if an error condition occurs, it is suggested that the HK_SKIN be reduced to a value less than K11 and K22 in node CELLID or the THEIM or MEAN conductance equations be used for these multi-aquifer wells.

- `radius_skin` real value that defines the skin radius (filter pack radius) for the multi-aquifer well. RADIUS_SKIN can be any value if CONDEQN is SPECIFIED or THIEM. If CONDEQN is SKIN, CUMULATIVE, or MEAN, the program will terminate with an error if RADIUS_SKIN is less than or equal to the RADIUS for the multi-aquifer well.

Block: PERIOD

- `iper` integer value specifying the starting stress period number for which the data specified in the PERIOD block apply. IPER must be less than or equal to NPER in the TDIS Package and greater than zero. The IPER value assigned to a stress period block must be greater than the IPER value assigned for the previous PERIOD block. The information specified in the PERIOD block will continue to apply for all subsequent stress periods, unless the program encounters another PERIOD block.
- `wellno` integer value that defines the well number associated with the specified PERIOD data on the line. WELLNO must be greater than zero and less than or equal to NMAWWELLS.
- `mawsetting` line of information that is parsed into a keyword and values. Keyword values that can be used to start the MAWSETTING string include: STATUS, FLOWING_WELL, RATE, WELL_HEAD, HEAD_LIMIT, SHUT_OFF, RATE_SCALING, and AUXILIARY.

```
STATUS <status>
FLOWING_WELL <fwelev> <fwcond> <fwrlen>
RATE <rate>
WELL_HEAD <well_head>
HEAD_LIMIT <head_limit>
SHUT_OFF <minrate> <maxrate>
RATE_SCALING <pump_elevation> <scaling_length>
AUXILIARY <auxname> <auxval>
```

- `status` keyword option to define well status. STATUS can be ACTIVE, INACTIVE, or CONSTANT. By default, STATUS is ACTIVE.
- `FLOWING_WELL` keyword to indicate the well is a flowing well. The FLOWING_WELL option can be used to simulate flowing wells when the simulated well head exceeds the specified drainage elevation.
- `fwelev` elevation used to determine whether or not the well is flowing.
- `fwcond` conductance used to calculate the discharge of a free flowing well. Flow occurs when the head in the well is above the well top elevation (FWELEV).
- `fwrlen` length used to reduce the conductance of the flowing well. When the head in the well drops below the well top plus the reduction length, then the conductance is reduced. This reduction length can be used to improve the stability of simulations with flowing wells so that there is not an abrupt change in flowing well rates.
- `rate` is the volumetric pumping rate for the multi-aquifer well. A positive value indicates recharge and a negative value indicates discharge (pumping). RATE only applies to active (IBOUND > 0) multi-aquifer wells. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value. By default, the RATE for each multi-aquifer well is zero.
- `well_head` is the head in the multi-aquifer well. WELL_HEAD is only applied to constant head (STATUS is CONSTANT) and inactive (STATUS is INACTIVE) multi-aquifer wells. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series

by entering the time-series name in place of a numeric value. The program will terminate with an error if WELL_HEAD is less than the bottom of the well.

- `head_limit` is the limiting water level (head) in the well, which is the minimum of the well RATE or the well inflow rate from the aquifer. HEAD_LIMIT can be applied to extraction wells (RATE < 0) or injection wells (RATE > 0). HEAD_LIMIT can be deactivated by specifying the text string "OFF". The HEAD_LIMIT option is based on the HEAD_LIMIT functionality available in the MNW2 package for MODFLOW-2005. The HEAD_LIMIT option has been included to facilitate backward compatibility with previous versions of MODFLOW but use of the RATE_SCALING option instead of the HEAD_LIMIT option is recommended. By default, HEAD_LIMIT is "OFF".
- `SHUT_OFF` keyword for activating well shut off capability. Subsequent values define the minimum and maximum pumping rate that a well must exceed to shutoff or reactivate a well, respectively, during a stress period. SHUT_OFF is only applied to injection wells (RATE<0) and if HEAD_LIMIT is specified (not set to "OFF"). If HEAD_LIMIT is specified, SHUT_OFF can be deactivated by specifying a minimum value equal to zero. The SHUT_OFF option is based on the SHUT_OFF functionality available in the MNW2 package for MODFLOW-2005. The SHUT_OFF option has been included to facilitate backward compatibility with previous versions of MODFLOW but use of the RATE_SCALING option instead of the SHUT_OFF option is recommended. By default, SHUT_OFF is not used.
- `minrate` is the minimum rate that a well must exceed to shutoff a well during a stress period. The well will shut down during a time step if the flow rate to the well from the aquifer is less than MINRATE. If a well is shut down during a time step, reactivation of the well cannot occur until the next time step to reduce oscillations. MINRATE must be less than maxrate.
- `maxrate` is the maximum rate that a well must exceed to reactivate a well during a stress period. The well will reactivate during a timestep if the well was shutdown during the previous time step and the flow rate to the well from the aquifer exceeds maxrate. Reactivation of the well cannot occur until the next time step if a well is shutdown to reduce oscillations. maxrate must be greater than MINRATE.
- `RATE_SCALING` activate rate scaling. If RATE_SCALING is specified, both PUMP_ELEVATION and SCALING_LENGTH must be specified. RATE_SCALING cannot be used with HEAD_LIMIT. RATE_SCALING can be used for extraction or injection wells. For extraction wells, the extraction rate will start to decrease once the head in the well lowers to a level equal to the pump elevation plus the scaling length. If the head in the well drops below the pump elevation, then the extraction rate is calculated to be zero. For an injection well, the injection rate will begin to decrease once the head in the well rises above the specified pump elevation. If the head in the well rises above the pump elevation plus the scaling length, then the injection rate will be set to zero.
- `pump_elevation` is the elevation of the multi-aquifer well pump (PUMP_ELEVATION). PUMP_ELEVATION should not be less than the bottom elevation (BOTTOM) of the multi-aquifer well.
- `scaling_length` height above the pump elevation (SCALING_LENGTH). If the simulated well head is below this elevation (pump elevation plus the scaling length), then the pumping rate is reduced.
- `AUXILIARY` keyword for specifying auxiliary variable.
- `auxname` name for the auxiliary variable to be assigned AUXVAL. AUXNAME must match one of the auxiliary variable names defined in the OPTIONS block. If AUXNAME does not match one of the auxiliary variable names defined in the OPTIONS block the data are ignored.
- `auxval` value for the auxiliary variable. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.

Example Input File

Example 1

```
  begin options
    print_input
    print_head
    print_flows
    boundnames
    head fileout maw-1.head.bin
    budget fileout maw-1.cbc
  end options

  begin dimensions
    nmawwells 2
  end dimensions

  begin packagedata
  #  wellno radius bottom strt condeqn ngwnodes name
          1   0.15 -100.0 9.14   thiem        2 pwell
          2   0.25 -100.0 9.14   thiem        1 iwell
  end packagedata

  begin connectiondata
  # wellno conn l  r  c  stop sbot  k  rskin
         1    1 1 51 51     0    0  0      0
         1    2 2 51 51     0    0  0      0
         2    1 2  2  2     0    0  0      0
  end connectiondata

  begin period 1
    1 rate_scaling -90. 5.
    1 rate -1767.
    2 status inactive
  end period

  begin period 100
    2 status active
    2 rate 529.
    1 rate -2767.
  end period
```

Example 2

```
  begin options
    print_input
    print_head
    print_flows
    boundnames
  end options

  begin dimensions
    nmawwells 2
  end dimensions

  begin packagedata
  #  wellno radius bottom strt   condeqn ngwnodes name
          1   0.15 -100.0 9.14      mean        2 pwell
```

(continues on next page)

(continued from previous page)

```
        2   0.25 -100.0 9.14      mean        1 iwell
end packagedata

begin connectiondata
# wellno conn l  r  c  stop  sbot  k  rskin
       1    1 1 51 51    0. -100. 361.  .25
       1    2 2 51 51    0. -100. 361.  .25
       2    1 2  2  2  -50. -100. 361   .50
end connectiondata

begin period 1
  1 rate_scaling -90. 5.
  1 rate -1767.
  2 status inactive
end period

begin period 100
  2 status active
  2 rate 529.
  1 rate -2767.
end period
```

Example 3

```
begin options
  print_input
  print_head
  print_flows
  boundnames
  flowing_wells
end options

begin dimensions
  nmawwells 1
end dimensions

begin packagedata
#  wellno radius bottom strt   condeqn ngwnodes name
        1   0.15 -514.9 9.14 specified        2 ntwell
end packagedata

begin connectiondata
# wellno conn l  r  c  stop   sbot         k  rskin
       1    1 1 51 51   -50 -514.9  111.3763      0
       1    2 2 51 51   -50 -514.9  445.9849      0
end connectiondata

begin period 1
  1 rate 0
  1 flowing_well 0. 7500. 0.5
end period
```

Available Observation Types

Example Observation Input File

```
BEGIN OPTIONS
  DIGITS 10
  PRINT_INPUT
END OPTIONS

BEGIN CONTINUOUS FILEOUT my_model.maw.csv
  m1head    head  1
  m1rate01  maw   1 1
  m1rate02  maw   1 2
  m1rate    maw   well-1
  m2rate01  maw   well-2
END CONTINUOUS
```

1.3.16 GWF-MVR

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [PRINT_INPUT]
  [PRINT_FLOWS]
  [MODELNAMES]
  [BUDGET FILEOUT <budgetfile>]
END OPTIONS
```

```
BEGIN DIMENSIONS
  MAXMVR <maxmvr>
  MAXPACKAGES <maxpackages>
END DIMENSIONS
```

```
BEGIN PACKAGES
  [<mname>] <pname>
  [<mname>] <pname>
  ...
END PACKAGES
```

FOR ANY STRESS PERIOD

```
BEGIN PERIOD <iper>
  [<mname1>] <pname1> <id1> [<mname2>] <pname2> <id2> <mvrtype> <value>
  [<mname1>] <pname1> <id1> [<mname2>] <pname2> <id2> <mvrtype> <value>
  ...
END PERIOD
```

Explanation of Variables

Block: OPTIONS

- PRINT_INPUT keyword to indicate that the list of MVR information will be written to the listing file immediately after it is read.
- PRINT_FLOWS keyword to indicate that the list of MVR flow rates will be printed to the listing file for every stress period time step in which "BUDGET PRINT" is specified in Output Control. If there is no Output Control option and "PRINT_FLOWS" is specified, then flow rates are printed for the last time step of each stress period.
- MODELNAMES keyword to indicate that all package names will be preceded by the model name for the package. Model names are required when the Mover Package is used with a GWF-GWF Exchange. The MODELNAME keyword should not be used for a Mover Package that is for a single GWF Model.
- BUDGET keyword to specify that record corresponds to the budget.
- FILEOUT keyword to specify that an output filename is expected next.
- budgetfile name of the output file to write budget information.

Block: DIMENSIONS

- maxmvr integer value specifying the maximum number of water mover entries that will specified for any stress period.
- maxpackages integer value specifying the number of unique packages that are included in this water mover input file.

Block: PACKAGES

- mname name of model containing the package. Model names are assigned by the user in the simulation name file.
- pname is the name of a package that may be included in a subsequent stress period block. The package name is assigned in the name file for the GWF Model. Package names are optionally provided in the name file. If they are not provided by the user, then packages are assigned a default value, which is the package acronym followed by a hyphen and the package number. For example, the first Drain Package is named DRN-1. The second Drain Package is named DRN-2, and so forth.

Block: PERIOD

- iper integer value specifying the starting stress period number for which the data specified in the PERIOD block apply. IPER must be less than or equal to NPER in the TDIS Package and greater than zero. The IPER value assigned to a stress period block must be greater than the IPER value assigned for the previous PERIOD block. The information specified in the PERIOD block will continue to apply for all subsequent stress periods, unless the program encounters another PERIOD block.
- mname1 name of model containing the package, PNAME1.
- pname1 is the package name for the provider. The package PNAME1 must be designated to provide water through the MVR Package by specifying the keyword "MOVER" in its OPTIONS block.
- id1 is the identifier for the provider. For the standard boundary packages, the provider identifier is the number of the boundary as it is listed in the package input file. (Note that the order of these boundaries may change by stress period, which must be accounted for in the Mover Package.) So the first well has an identifier of one.

The second is two, and so forth. For the advanced packages, the identifier is the reach number (SFR Package), well number (MAW Package), or UZF cell number. For the Lake Package, ID1 is the lake outlet number. Thus, outflows from a single lake can be routed to different streams, for example.

- `mname2` name of model containing the package, PNAME2.
- `pname2` is the package name for the receiver. The package PNAME2 must be designated to receive water from the MVR Package by specifying the keyword "MOVER" in its OPTIONS block.
- `id2` is the identifier for the receiver. The receiver identifier is the reach number (SFR Package), Lake number (LAK Package), well number (MAW Package), or UZF cell number.
- `mvrtype` is the character string signifying the method for determining how much water will be moved. Supported values are "FACTOR" "EXCESS" "THRESHOLD" and "UPTO". These four options determine how the receiver flow rate, Q_R, is calculated. These options mirror the options defined for the cprior variable in the SFR package, with the term "FACTOR" being functionally equivalent to the "FRACTION" option for cprior.
- `value` is the value to be used in the equation for calculating the amount of water to move. For the "FACTOR" option, VALUE is the alpha factor. For the remaining options, VALUE is the specified flow rate, Q_S.

Example Input File

```
BEGIN OPTIONS
  PRINT_INPUT
  PRINT_FLOWS
END OPTIONS

BEGIN DIMENSIONS
  MAXMVR 16
  MAXPACKAGES 5
END DIMENSIONS

BEGIN PACKAGES
  MAW-1
  MAW-2
  SFR-1
  LAK-1
  UZF-1
END PACKAGES

BEGIN PERIOD 1
# ***PROVIDER***   ***RECEIVER***  ***FLOW INFO**
#  PAK1  PAK1RCH    PAK2  PAK2RCH  TYPE     VALUE
  MAW-1         1   MAW-2        21  EXCESS    5.00
  MAW-1        11   SFR-1        77  FACTOR    0.25
  MAW-1        21   UZF-1        93  FACTOR    0.01
  MAW-1        21   LAK-1         3  FACTOR    1.00

  SFR-1      1021   MAW-1        21  THRESHOLD 10.0
  SFR-1       441   SFR-1        77  FACTOR    0.10
  SFR-1        56   UZF-1        93  FACTOR    0.10
  SFR-1      4587   LAK-1         3  FACTOR    1.00

  UZF-1         4   MAW-1        11  FACTOR    1.00
  UZF-1         5   SFR-1        22  FACTOR    1.00
  UZF-1         6   UZF-1        45  FACTOR    1.00
  UZF-1         7   LAK-1         3  FACTOR    1.00
```

(continues on next page)

(continued from previous page)

```
  LAK-1        1   MAW-1       11  EXCESS    1000.
  LAK-1        2   SFR-1       22  UPTO      2000.
  LAK-1        3   UZF-1       45  UPTO      3000.
  LAK-1        4   LAK-1        3  UPTO      3000.
END PERIOD 1
```

1.3.17 GWF-NAM

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [LIST <list>]
  [PRINT_INPUT]
  [PRINT_FLOWS]
  [SAVE_FLOWS]
  [NEWTON [UNDER_RELAXATION]]
END OPTIONS
```

```
BEGIN PACKAGES
  <ftype> <fname> [<pname>]
  <ftype> <fname> [<pname>]
  ...
END PACKAGES
```

Explanation of Variables

Block: OPTIONS

- `list` is name of the listing file to create for this GWF model. If not specified, then the name of the list file will be the basename of the GWF model name file and the '.lst' extension. For example, if the GWF name file is called "my.model.nam" then the list file will be called "my.model.lst".
- `PRINT_INPUT` keyword to indicate that the list of all model stress package information will be written to the listing file immediately after it is read.
- `PRINT_FLOWS` keyword to indicate that the list of all model package flow rates will be printed to the listing file for every stress period time step in which "BUDGET PRINT" is specified in Output Control. If there is no Output Control option and "PRINT_FLOWS" is specified, then flow rates are printed for the last time step of each stress period.
- `SAVE_FLOWS` keyword to indicate that all model package flow terms will be written to the file specified with "BUDGET FILEOUT" in Output Control.
- `NEWTON` keyword that activates the Newton-Raphson formulation for groundwater flow between connected, convertible groundwater cells and stress packages that support calculation of Newton-Raphson terms for groundwater exchanges. Cells will not dry when this option is used. By default, the Newton-Raphson formulation is not applied.
- `UNDER_RELAXATION` keyword that indicates whether the groundwater head in a cell will be under-relaxed when water levels fall below the bottom of the model below any given cell. By default, Newton-Raphson UNDER_RELAXATION is not applied.

Block: PACKAGES

- `ftype` is the file type, which must be one of the following character values shown in table ref{table:ftype}. Ftype may be entered in any combination of uppercase and lowercase.
- `fname` is the name of the file containing the package input. The path to the file should be included if the file is not located in the folder where the program was run.
- `pname` is the user-defined name for the package. PNAME is restricted to 16 characters. No spaces are allowed in PNAME. PNAME character values are read and stored by the program for stress packages only. These names may be useful for labeling purposes when multiple stress packages of the same type are located within a single GWF Model. If PNAME is specified for a stress package, then PNAME will be used in the flow budget table in the listing file; it will also be used for the text entry in the cell-by-cell budget file. PNAME is case insensitive and is stored in all upper case letters.

Example Input File

```
# This block is optional
BEGIN OPTIONS
  PRINT_INPUT
  PRINT_FLOWS
  SAVE_FLOWS
END OPTIONS

# List of packages.  List can be listed in any order.
BEGIN PACKAGES
  IC6           bcf2ss.ic
  NPF6          bcf2ss.npf
  WEL6          bcf2ss.wel  WEL-COUNTY
  RIV6          bcf2ss.riv
  RCH6          bcf2ss.rch
  OC6           bcf2ss.oc
  DIS6          bcf2ss.dis
END PACKAGES
```

1.3.18 GWF-NPF

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [SAVE_FLOWS]
  [ALTERNATIVE_CELL_AVERAGING <alternative_cell_averaging>]
  [THICKSTRT]
  [VARIABLECV [DEWATERED]]
  [PERCHED]
  [REWET WETFCT <wetfct> IWETIT <iwetit> IHDWET <ihdwet>]
  [XT3D [RHS]]
  [SAVE_SPECIFIC_DISCHARGE]
  [SAVE_SATURATION]
  [K22OVERK]
  [K33OVERK]
END OPTIONS
```

```
BEGIN GRIDDATA
  ICELLTYPE [LAYERED]
        <icelltype(nodes)> -- READARRAY
  K [LAYERED]
        <k(nodes)> -- READARRAY
  [K22 [LAYERED]
        <k22(nodes)> -- READARRAY]
  [K33 [LAYERED]
        <k33(nodes)> -- READARRAY]
  [ANGLE1 [LAYERED]
        <angle1(nodes)> -- READARRAY]
  [ANGLE2 [LAYERED]
        <angle2(nodes)> -- READARRAY]
  [ANGLE3 [LAYERED]
        <angle3(nodes)> -- READARRAY]
  [WETDRY [LAYERED]
        <wetdry(nodes)> -- READARRAY]
END GRIDDATA
```

Explanation of Variables

Block: OPTIONS

- SAVE_FLOWS keyword to indicate that budget flow terms will be written to the file specified with "BUDGET SAVE FILE" in Output Control.
- alternative_cell_averaging is a text keyword to indicate that an alternative method will be used for calculating the conductance for horizontal cell connections. The text value for ALTERNATIVE_CELL_AVERAGING can be "LOGARITHMIC", "AMT-LMK", or "AMT-HMK". "AMT-LMK" signifies that the conductance will be calculated using arithmetic-mean thickness and logarithmic-mean hydraulic conductivity. "AMT-HMK" signifies that the conductance will be calculated using arithmetic-mean thickness and harmonic-mean hydraulic conductivity. If the user does not specify a value for ALTERNATIVE_CELL_AVERAGING, then the harmonic-mean method will be used. This option cannot be used if the XT3D option is invoked.
- THICKSTRT indicates that cells having a negative ICELLTYPE are confined, and their cell thickness for conductance calculations will be computed as STRT-BOT rather than TOP-BOT.
- VARIABLECV keyword to indicate that the vertical conductance will be calculated using the saturated thickness and properties of the overlying cell and the thickness and properties of the underlying cell. If the DEWATERED keyword is also specified, then the vertical conductance is calculated using only the saturated thickness and properties of the overlying cell if the head in the underlying cell is below its top. If these keywords are not specified, then the default condition is to calculate the vertical conductance at the start of the simulation using the initial head and the cell properties. The vertical conductance remains constant for the entire simulation.
- DEWATERED If the DEWATERED keyword is specified, then the vertical conductance is calculated using only the saturated thickness and properties of the overlying cell if the head in the underlying cell is below its top.
- PERCHED keyword to indicate that when a cell is overlying a dewatered convertible cell, the head difference used in Darcy's Law is equal to the head in the overlying cell minus the bottom elevation of the overlying cell. If not specified, then the default is to use the head difference between the two cells.
- REWET activates model rewetting. Rewetting is off by default.
- wetfct is a keyword and factor that is included in the calculation of the head that is initially established at a cell when that cell is converted from dry to wet.

- iwetit is a keyword and iteration interval for attempting to wet cells. Wetting is attempted every IWETIT iteration. This applies to outer iterations and not inner iterations. If IWETIT is specified as zero or less, then the value is changed to 1.
- ihdwet is a keyword and integer flag that determines which equation is used to define the initial head at cells that become wet. If IHDWET is 0, h = BOT + WETFCT (hm - BOT). If IHDWET is not 0, h = BOT + WETFCT (THRESH).
- XT3D keyword indicating that the XT3D formulation will be used. If the RHS keyword is also included, then the XT3D additional terms will be added to the right-hand side. If the RHS keyword is excluded, then the XT3D terms will be put into the coefficient matrix. Use of XT3D will substantially increase the computational effort, but will result in improved accuracy for anisotropic conductivity fields and for unstructured grids in which the CVFD requirement is violated. XT3D requires additional information about the shapes of grid cells. If XT3D is active and the DISU Package is used, then the user will need to provide in the DISU Package the angldegx array in the CONNECTIONDATA block and the VERTICES and CELL2D blocks.
- RHS If the RHS keyword is also included, then the XT3D additional terms will be added to the right-hand side. If the RHS keyword is excluded, then the XT3D terms will be put into the coefficient matrix.
- SAVE_SPECIFIC_DISCHARGE keyword to indicate that x, y, and z components of specific discharge will be calculated at cell centers and written to the budget file, which is specified with "BUDGET SAVE FILE" in Output Control. If this option is activated, then additional information may be required in the discretization packages and the GWF Exchange package (if GWF models are coupled). Specifically, ANGLDEGX must be specified in the CONNECTIONDATA block of the DISU Package; ANGLDEGX must also be specified for the GWF Exchange as an auxiliary variable.
- SAVE_SATURATION keyword to indicate that cell saturation will be written to the budget file, which is specified with "BUDGET SAVE FILE" in Output Control. Saturation will be saved to the budget file as an auxiliary variable saved with the DATA-SAT text label. Saturation is a cell variable that ranges from zero to one and can be used by post processing programs to determine how much of a cell volume is saturated. If ICELLTYPE is 0, then saturation is always one.
- K22OVERK keyword to indicate that specified K22 is a ratio of K22 divided by K. If this option is specified, then the K22 array entered in the NPF Package will be multiplied by K after being read.
- K33OVERK keyword to indicate that specified K33 is a ratio of K33 divided by K. If this option is specified, then the K33 array entered in the NPF Package will be multiplied by K after being read.

Block: GRIDDATA

- icelltype flag for each cell that specifies how saturated thickness is treated. 0 means saturated thickness is held constant; >0 means saturated thickness varies with computed head when head is below the cell top; <0 means saturated thickness varies with computed head unless the THICKSTRT option is in effect. When THICKSTRT is in effect, a negative value of icelltype indicates that saturated thickness will be computed as STRT-BOT and held constant.
- k is the hydraulic conductivity. For the common case in which the user would like to specify the horizontal hydraulic conductivity and the vertical hydraulic conductivity, then K should be assigned as the horizontal hydraulic conductivity, K33 should be assigned as the vertical hydraulic conductivity, and texttt{K22} and the three rotation angles should not be specified. When more sophisticated anisotropy is required, then K corresponds to the K11 hydraulic conductivity axis. All included cells (IDOMAIN > 0) must have a K value greater than zero.
- k22 is the hydraulic conductivity of the second ellipsoid axis (or the ratio of K22/K if the K22OVERK option is specified); for an unrotated case this is the hydraulic conductivity in the y direction. If K22 is not included in the GRIDDATA block, then K22 is set equal to K. For a regular MODFLOW grid (DIS Package is used) in which no rotation angles are specified, K22 is the hydraulic conductivity along columns in the y direction. For

an unstructured DISU grid, the user must assign principal x and y axes and provide the angle for each cell face relative to the assigned x direction. All included cells (IDOMAIN > 0) must have a K22 value greater than zero.

- `k33` is the hydraulic conductivity of the third ellipsoid axis (or the ratio of K33/K if the K33OVERK option is specified); for an unrotated case, this is the vertical hydraulic conductivity. When anisotropy is applied, K33 corresponds to the K33 tensor component. All included cells (IDOMAIN > 0) must have a K33 value greater than zero.
- `angle1` is a rotation angle of the hydraulic conductivity tensor in degrees. The angle represents the first of three sequential rotations of the hydraulic conductivity ellipsoid. With the K11, K22, and K33 axes of the ellipsoid initially aligned with the x, y, and z coordinate axes, respectively, ANGLE1 rotates the ellipsoid about its K33 axis (within the x - y plane). A positive value represents counter-clockwise rotation when viewed from any point on the positive K33 axis, looking toward the center of the ellipsoid. A value of zero indicates that the K11 axis lies within the x - z plane. If ANGLE1 is not specified, default values of zero are assigned to ANGLE1, ANGLE2, and ANGLE3, in which case the K11, K22, and K33 axes are aligned with the x, y, and z axes, respectively.
- `angle2` is a rotation angle of the hydraulic conductivity tensor in degrees. The angle represents the second of three sequential rotations of the hydraulic conductivity ellipsoid. Following the rotation by ANGLE1 described above, ANGLE2 rotates the ellipsoid about its K22 axis (out of the x - y plane). An array can be specified for ANGLE2 only if ANGLE1 is also specified. A positive value of ANGLE2 represents clockwise rotation when viewed from any point on the positive K22 axis, looking toward the center of the ellipsoid. A value of zero indicates that the K11 axis lies within the x - y plane. If ANGLE2 is not specified, default values of zero are assigned to ANGLE2 and ANGLE3; connections that are not user-designated as vertical are assumed to be strictly horizontal (that is, to have no z component to their orientation); and connection lengths are based on horizontal distances.
- `angle3` is a rotation angle of the hydraulic conductivity tensor in degrees. The angle represents the third of three sequential rotations of the hydraulic conductivity ellipsoid. Following the rotations by ANGLE1 and ANGLE2 described above, ANGLE3 rotates the ellipsoid about its K11 axis. An array can be specified for ANGLE3 only if ANGLE1 and ANGLE2 are also specified. An array must be specified for ANGLE3 if ANGLE2 is specified. A positive value of ANGLE3 represents clockwise rotation when viewed from any point on the positive K11 axis, looking toward the center of the ellipsoid. A value of zero indicates that the K22 axis lies within the x - y plane.
- `wetdry` is a combination of the wetting threshold and a flag to indicate which neighboring cells can cause a cell to become wet. If WETDRY < 0, only a cell below a dry cell can cause the cell to become wet. If WETDRY > 0, the cell below a dry cell and horizontally adjacent cells can cause a cell to become wet. If WETDRY is 0, the cell cannot be wetted. The absolute value of WETDRY is the wetting threshold. When the sum of BOT and the absolute value of WETDRY at a dry cell is equaled or exceeded by the head at an adjacent cell, the cell is wetted. WETDRY must be specified if "REWET" is specified in the OPTIONS block. If "REWET" is not specified in the options block, then WETDRY can be entered, and memory will be allocated for it, even though it is not used.

Example Input File

```
BEGIN OPTIONS
  SAVE_FLOWS
END OPTIONS

BEGIN GRIDDATA
  #
  #icelltype(nodes) is 0:confined, 1:convertible
  ICELLTYPE
    constant 0
```

(continues on next page)

(continued from previous page)

```
    #
    # horizontal hydraulic conductivity
    K
      constant 1.0
    #
    # vertical hydraulic conductivity
    K33
      constant 0.1
  END GRIDDATA
```

1.3.19 GWF-OC

Structure of Blocks

FOR EACH SIMULATION

```
  BEGIN OPTIONS
    [BUDGET FILEOUT <budgetfile>]
    [HEAD FILEOUT <headfile>]
    [HEAD PRINT_FORMAT COLUMNS <columns> WIDTH <width> DIGITS <digits> <format>]
  END OPTIONS
```

FOR ANY STRESS PERIOD

```
  BEGIN PERIOD <iper>
    [SAVE <rtype> <ocsetting>]
    [PRINT <rtype> <ocsetting>]
  END PERIOD
```

Explanation of Variables

Block: OPTIONS

- `BUDGET` keyword to specify that record corresponds to the budget.
- `FILEOUT` keyword to specify that an output filename is expected next.
- `budgetfile` name of the output file to write budget information.
- `HEAD` keyword to specify that record corresponds to head.
- `headfile` name of the output file to write head information.
- `PRINT_FORMAT` keyword to specify format for printing to the listing file.
- `columns` number of columns for writing data.
- `width` width for writing each number.
- `digits` number of digits to use for writing a number.
- `format` write format can be EXPONENTIAL, FIXED, GENERAL, or SCIENTIFIC.

Block: PERIOD

- `iper` integer value specifying the starting stress period number for which the data specified in the PERIOD block apply. IPER must be less than or equal to NPER in the TDIS Package and greater than zero. The IPER value assigned to a stress period block must be greater than the IPER value assigned for the previous PERIOD block. The information specified in the PERIOD block will continue to apply for all subsequent stress periods, unless the program encounters another PERIOD block.
- `SAVE` keyword to indicate that information will be saved this stress period.
- `PRINT` keyword to indicate that information will be printed this stress period.
- `rtype` type of information to save or print. Can be BUDGET or HEAD.
- `ocsetting` specifies the steps for which the data will be saved.

```
ALL
FIRST
LAST
FREQUENCY <frequency>
STEPS <steps(<nstp)>
```

- `ALL` keyword to indicate save for all time steps in period.
- `FIRST` keyword to indicate save for first step in period. This keyword may be used in conjunction with other keywords to print or save results for multiple time steps.
- `LAST` keyword to indicate save for last step in period. This keyword may be used in conjunction with other keywords to print or save results for multiple time steps.
- `frequency` save at the specified time step frequency. This keyword may be used in conjunction with other keywords to print or save results for multiple time steps.
- `steps` save for each step specified in STEPS. This keyword may be used in conjunction with other keywords to print or save results for multiple time steps.

Example Input File

```
BEGIN OPTIONS
  HEAD FILEOUT AdvGW_tidal.hds
  BUDGET FILEOUT AdvGW_tidal.cbc
  HEAD PRINT_FORMAT COLUMNS 100 WIDTH 15 DIGITS 4 GENERAL
END OPTIONS

BEGIN PERIOD 1
  PRINT HEAD FIRST
  PRINT HEAD LAST
  PRINT BUDGET LAST
  SAVE HEAD ALL
  SAVE BUDGET ALL
END PERIOD

# No output for stress periods 2 through 24
BEGIN PERIOD 2
END PERIOD

BEGIN PERIOD 25
  PRINT HEAD STEPS 6 12 23
```

(continues on next page)

(continued from previous page)

```
  SAVE BUDGET FIRST
  SAVE BUDGET LAST
  SAVE BUDGET FREQUENCY 5
END PERIOD
```

1.3.20 GWF-RCH

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [FIXED_CELL]
  [AUXILIARY <auxiliary(naux)>]
  [AUXMULTNAME <auxmultname>]
  [BOUNDNAMES]
  [PRINT_INPUT]
  [PRINT_FLOWS]
  [SAVE_FLOWS]
  [TS6 FILEIN <ts6_filename>]
  [OBS6 FILEIN <obs6_filename>]
END OPTIONS
```

```
BEGIN DIMENSIONS
  MAXBOUND <maxbound>
END DIMENSIONS
```

FOR ANY STRESS PERIOD

```
BEGIN PERIOD <iper>
  <cellid(ncelldim)> <recharge> [<aux(naux)>] [<boundname>]
  <cellid(ncelldim)> <recharge> [<aux(naux)>] [<boundname>]
  ...
END PERIOD
```

Explanation of Variables

Block: OPTIONS

- `FIXED_CELL` indicates that recharge will not be reassigned to a cell underlying the cell specified in the list if the specified cell is inactive.
- `auxiliary` defines an array of one or more auxiliary variable names. There is no limit on the number of auxiliary variables that can be provided on this line; however, lists of information provided in subsequent blocks must have a column of data for each auxiliary variable name defined here. The number of auxiliary variables detected on this line determines the value for naux. Comments cannot be provided anywhere on this line as they will be interpreted as auxiliary variable names. Auxiliary variables may not be used by the package, but they will be available for use by other parts of the program. The program will terminate with an error if auxiliary variables are specified on more than one line in the options block.
- `auxmultname` name of auxiliary variable to be used as multiplier of recharge.
- `BOUNDNAMES` keyword to indicate that boundary names may be provided with the list of recharge cells.

- PRINT_INPUT keyword to indicate that the list of recharge information will be written to the listing file immediately after it is read.
- PRINT_FLOWS keyword to indicate that the list of recharge flow rates will be printed to the listing file for every stress period time step in which "BUDGET PRINT" is specified in Output Control. If there is no Output Control option and "PRINT_FLOWS" is specified, then flow rates are printed for the last time step of each stress period.
- SAVE_FLOWS keyword to indicate that recharge flow terms will be written to the file specified with "BUDGET FILEOUT" in Output Control.
- TS6 keyword to specify that record corresponds to a time-series file.
- FILEIN keyword to specify that an input filename is expected next.
- ts6_filename defines a time-series file defining time series that can be used to assign time-varying values. See the "Time-Variable Input" section for instructions on using the time-series capability.
- OBS6 keyword to specify that record corresponds to an observations file.
- obs6_filename name of input file to define observations for the Recharge package. See the "Observation utility" section for instructions for preparing observation input files. Tables ref{table:gwf-obstypetable} and ref{table:gwt-obstypetable} lists observation type(s) supported by the Recharge package.

Block: DIMENSIONS

- maxbound integer value specifying the maximum number of recharge cells cells that will be specified for use during any stress period.

Block: PERIOD

- iper integer value specifying the starting stress period number for which the data specified in the PERIOD block apply. IPER must be less than or equal to NPER in the TDIS Package and greater than zero. The IPER value assigned to a stress period block must be greater than the IPER value assigned for the previous PERIOD block. The information specified in the PERIOD block will continue to apply for all subsequent stress periods, unless the program encounters another PERIOD block.
- cellid is the cell identifier, and depends on the type of grid that is used for the simulation. For a structured grid that uses the DIS input file, CELLID is the layer, row, and column. For a grid that uses the DISV input file, CELLID is the layer and CELL2D number. If the model uses the unstructured discretization (DISU) input file, CELLID is the node number for the cell.
- recharge is the recharge flux rate (LT-1). This rate is multiplied inside the program by the surface area of the cell to calculate the volumetric recharge rate. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- aux represents the values of the auxiliary variables for each recharge. The values of auxiliary variables must be present for each recharge. The values must be specified in the order of the auxiliary variables specified in the OPTIONS block. If the package supports time series and the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- boundname name of the recharge cell. BOUNDNAME is an ASCII character variable that can contain as many as 40 characters. If BOUNDNAME contains spaces in it, then the entire name must be enclosed within single quotes.

Example Input File

Example 1

```
BEGIN OPTIONS
  AUXILIARY var1 var2 mult
  PRINT_INPUT
  PRINT_FLOWS
  SAVE_FLOWS
  BOUNDNAMES
  TS6 FILEIN recharge_rates.ts
  # Note: Time-series file recharge_rates.ts defines time series rch_1
  AUXMULTNAME mult
END OPTIONS

BEGIN DIMENSIONS
  MAXBOUND  10
END DIMENSIONS

BEGIN PERIOD 1
  RECHARGE
  # Lay  Row  Col  Rate   Var1  Var2   mult  BoundName
     1    1    1   rch_1   1.0   2.0     1.0       Rch-1-1
     1    1    2   rch_1   1.1   2.1     1.0       Rch-1-2
     1    1    3   rch_1   1.2   2.2     0.5
     1    2    1   rch_1   1.3   2.3     1.0       Rch-2-1
     1    2    2   rch_1   1.4   2.4     1.0       Rch-2-2
     1    2    3   rch_1   1.5   2.5     1.0
     1    2    4   rch_1   1.6   2.6     0.5
     1    3    1   rch_1   1.7   2.7     1.0
     1    3    2   rch_1   1.8   2.8     1.0
     1    3    3   rch_1   1.9   2.9     1.0
END PERIOD
```

Example 2

```
BEGIN OPTIONS
  AUXILIARY var1 var2 mymult
  READASARRAYS
  PRINT_INPUT
  PRINT_FLOWS
  SAVE_FLOWS
  AUXMULTNAME mymult
END OPTIONS

BEGIN PERIOD 1

  # For this model, the absence of an IRCH array causes
  # recharge to apply to model layer 1. To make recharge
  # apply to layer 2 instead, the following lines
  # (uncommented) could be used:
  # IRCH
  #   constant 2

  # recharge rate
  RECHARGE
    constant 0.0040
```

(continues on next page)

(continued from previous page)

```
    # auxiliary variable (var1) array
    var1
      constant 100.

    # auxiliary variable (var2) array
    var2
      constant 0.

    # auxiliary variable (mymult) array
    # Because ``AUXMULTNAME mymult'' was specified in the
    # options block, the MYMULT array will be used to multiply
    # the values in the RECHARGE array
    MYMULT
      INTERNAL FACTOR  1.0
        0.5  1.0  1.0  0.5
        1.0  1.0  1.0  1.0
        0.5  1.0  1.0  0.5

  END PERIOD
```

Available Observation Types

Example Observation Input File

```
  BEGIN OPTIONS
    PRINT_INPUT
  END OPTIONS

  BEGIN CONTINUOUS FILEOUT my_model.rch.csv
    rch1-1    RCH   Rch-1-1
    rch1-2    RCH   Rch-1-2
    rch2-1    RCH   Rch-2-1
    rch2-2    RCH   Rch-2-2
    rch2-3    RCH   1  2  3
  END CONTINUOUS
```

1.3.21 GWF-RCHA

Structure of Blocks

FOR EACH SIMULATION

```
  BEGIN OPTIONS
    READASARRAYS
    [FIXED_CELL]
    [AUXILIARY <auxiliary(naux)>]
    [AUXMULTNAME <auxmultname>]
    [PRINT_INPUT]
    [PRINT_FLOWS]
    [SAVE_FLOWS]
    [TAS6 FILEIN <tas6_filename>]
    [OBS6 FILEIN <obs6_filename>]
  END OPTIONS
```

FOR ANY STRESS PERIOD

```
BEGIN PERIOD <iper>
  [IRCH
        <irch(ncol*nrow; ncpl)> -- READARRAY]
  RECHARGE
        <recharge(ncol*nrow; ncpl)> -- READARRAY
  [AUX
        <aux(ncol*nrow; ncpl)> -- READARRAY]
END PERIOD
```

Explanation of Variables

Block: OPTIONS

- READASARRAYS indicates that array-based input will be used for the Recharge Package. This keyword must be specified to use array-based input.
- FIXED_CELL indicates that recharge will not be reassigned to a cell underlying the cell specified in the list if the specified cell is inactive.
- auxiliary defines an array of one or more auxiliary variable names. There is no limit on the number of auxiliary variables that can be provided on this line; however, lists of information provided in subsequent blocks must have a column of data for each auxiliary variable name defined here. The number of auxiliary variables detected on this line determines the value for naux. Comments cannot be provided anywhere on this line as they will be interpreted as auxiliary variable names. Auxiliary variables may not be used by the package, but they will be available for use by other parts of the program. The program will terminate with an error if auxiliary variables are specified on more than one line in the options block.
- auxmultname name of auxiliary variable to be used as multiplier of recharge.
- PRINT_INPUT keyword to indicate that the list of recharge information will be written to the listing file immediately after it is read.
- PRINT_FLOWS keyword to indicate that the list of recharge flow rates will be printed to the listing file for every stress period time step in which "BUDGET PRINT" is specified in Output Control. If there is no Output Control option and "PRINT_FLOWS" is specified, then flow rates are printed for the last time step of each stress period.
- SAVE_FLOWS keyword to indicate that recharge flow terms will be written to the file specified with "BUDGET FILEOUT" in Output Control.
- TAS6 keyword to specify that record corresponds to a time-array-series file.
- FILEIN keyword to specify that an input filename is expected next.
- tas6_filename defines a time-array-series file defining a time-array series that can be used to assign time-varying values. See the Time-Variable Input section for instructions on using the time-array series capability.
- OBS6 keyword to specify that record corresponds to an observations file.
- obs6_filename name of input file to define observations for the Recharge package. See the "Observation utility" section for instructions for preparing observation input files. Tables ref{table:gwf-obstypetable} and ref{table:gwt-obstypetable} lists observation type(s) supported by the Recharge package.

Block: PERIOD

- `iper` integer value specifying the starting stress period number for which the data specified in the PERIOD block apply. IPER must be less than or equal to NPER in the TDIS Package and greater than zero. The IPER value assigned to a stress period block must be greater than the IPER value assigned for the previous PERIOD block. The information specified in the PERIOD block will continue to apply for all subsequent stress periods, unless the program encounters another PERIOD block.
- `irch` IRCH is the layer number that defines the layer in each vertical column where recharge is applied. If IRCH is omitted, recharge by default is applied to cells in layer 1. IRCH can only be used if READASARRAYS is specified in the OPTIONS block. If IRCH is specified, it must be specified as the first variable in the PERIOD block or MODFLOW will terminate with an error.
- `recharge` is the recharge flux rate (LT-1). This rate is multiplied inside the program by the surface area of the cell to calculate the volumetric recharge rate. The recharge array may be defined by a time-array series (see the "Using Time-Array Series in a Package" section).
- `aux` is an array of values for auxiliary variable aux(iaux), where iaux is a value from 1 to naux, and aux(iaux) must be listed as part of the auxiliary variables. A separate array can be specified for each auxiliary variable. If an array is not specified for an auxiliary variable, then a value of zero is assigned. If the value specified here for the auxiliary variable is the same as auxmultname, then the recharge array will be multiplied by this array.

Example Input File

```
BEGIN OPTIONS
  AUXILIARY var1 var2 mymult
  READASARRAYS
  PRINT_INPUT
  PRINT_FLOWS
  SAVE_FLOWS
  AUXMULTNAME mymult
END OPTIONS

BEGIN PERIOD 1

  # For this model, the absence of an IRCH array causes
  # recharge to apply to model layer 1. To make recharge
  # apply to layer 2 instead, the following lines
  # (uncommented) could be used:
  # IRCH
  #   constant 2

  # recharge rate
  RECHARGE
    constant 0.0040

  # auxiliary variable (var1) array
  var1
    constant 100.

  # auxiliary variable (var2) array
  var2
    constant 0.

  # auxiliary variable (mymult) array
  # Because ``AUXMULTNAME mymult'' was specified in the
```

(continues on next page)

(continued from previous page)

```
    # options block, the MYMULT array will be used to multiply
    # the values in the RECHARGE array
    MYMULT
      INTERNAL FACTOR  1.0
        0.5  1.0  1.0  0.5
        1.0  1.0  1.0  1.0
        0.5  1.0  1.0  0.5

  END PERIOD
```

1.3.22 GWF-RIV

Structure of Blocks

FOR EACH SIMULATION

```
  BEGIN OPTIONS
    [AUXILIARY <auxiliary(naux)>]
    [AUXMULTNAME <auxmultname>]
    [BOUNDNAMES]
    [PRINT_INPUT]
    [PRINT_FLOWS]
    [SAVE_FLOWS]
    [TS6 FILEIN <ts6_filename>]
    [OBS6 FILEIN <obs6_filename>]
    [MOVER]
  END OPTIONS
```

```
  BEGIN DIMENSIONS
    MAXBOUND <maxbound>
  END DIMENSIONS
```

FOR ANY STRESS PERIOD

```
  BEGIN PERIOD <iper>
    <cellid(ncelldim)> <stage> <cond> <rbot> [<aux(naux)>] [<boundname>]
    <cellid(ncelldim)> <stage> <cond> <rbot> [<aux(naux)>] [<boundname>]
    ...
  END PERIOD
```

Explanation of Variables

Block: OPTIONS

- `auxiliary` defines an array of one or more auxiliary variable names. There is no limit on the number of auxiliary variables that can be provided on this line; however, lists of information provided in subsequent blocks must have a column of data for each auxiliary variable name defined here. The number of auxiliary variables detected on this line determines the value for naux. Comments cannot be provided anywhere on this line as they will be interpreted as auxiliary variable names. Auxiliary variables may not be used by the package, but they will be available for use by other parts of the program. The program will terminate with an error if auxiliary variables are specified on more than one line in the options block.
- `auxmultname` name of auxiliary variable to be used as multiplier of riverbed conductance.

- BOUNDNAMES keyword to indicate that boundary names may be provided with the list of river cells.
- PRINT_INPUT keyword to indicate that the list of river information will be written to the listing file immediately after it is read.
- PRINT_FLOWS keyword to indicate that the list of river flow rates will be printed to the listing file for every stress period time step in which "BUDGET PRINT" is specified in Output Control. If there is no Output Control option and "PRINT_FLOWS" is specified, then flow rates are printed for the last time step of each stress period.
- SAVE_FLOWS keyword to indicate that river flow terms will be written to the file specified with "BUDGET FILEOUT" in Output Control.
- TS6 keyword to specify that record corresponds to a time-series file.
- FILEIN keyword to specify that an input filename is expected next.
- ts6_filename defines a time-series file defining time series that can be used to assign time-varying values. See the "Time-Variable Input" section for instructions on using the time-series capability.
- OBS6 keyword to specify that record corresponds to an observations file.
- obs6_filename name of input file to define observations for the River package. See the "Observation utility" section for instructions for preparing observation input files. Tables ref{table:gwf-obstypetable} and ref{table:gwt-obstypetable} lists observation type(s) supported by the River package.
- MOVER keyword to indicate that this instance of the River Package can be used with the Water Mover (MVR) Package. When the MOVER option is specified, additional memory is allocated within the package to store the available, provided, and received water.

Block: DIMENSIONS

- maxbound integer value specifying the maximum number of rivers cells that will be specified for use during any stress period.

Block: PERIOD

- iper integer value specifying the starting stress period number for which the data specified in the PERIOD block apply. IPER must be less than or equal to NPER in the TDIS Package and greater than zero. The IPER value assigned to a stress period block must be greater than the IPER value assigned for the previous PERIOD block. The information specified in the PERIOD block will continue to apply for all subsequent stress periods, unless the program encounters another PERIOD block.
- cellid is the cell identifier, and depends on the type of grid that is used for the simulation. For a structured grid that uses the DIS input file, CELLID is the layer, row, and column. For a grid that uses the DISV input file, CELLID is the layer and CELL2D number. If the model uses the unstructured discretization (DISU) input file, CELLID is the node number for the cell.
- stage is the head in the river. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- cond is the riverbed hydraulic conductance. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- rbot is the elevation of the bottom of the riverbed. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.

- `aux` represents the values of the auxiliary variables for each river. The values of auxiliary variables must be present for each river. The values must be specified in the order of the auxiliary variables specified in the OPTIONS block. If the package supports time series and the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `boundname` name of the river cell. BOUNDNAME is an ASCII character variable that can contain as many as 40 characters. If BOUNDNAME contains spaces in it, then the entire name must be enclosed within single quotes.

Example Input File

```
BEGIN OPTIONS
  PRINT_INPUT
  PRINT_FLOWS
  SAVE_FLOWS
  BOUNDNAMES
  TS6 FILEIN river_stages.ts
END OPTIONS

begin dimensions
  MAXBOUND 20
end dimensions

BEGIN PERIOD 1
# layer   row    col    stage           cond    rbot  BoundName
      1     3     1   river_stage_1     1001.   35.9
      1     4     2   river_stage_1     1002.   35.8
      1     5     3   river_stage_1     1003.   35.7
      1     5     4   river_stage_1     1004.   35.6
      1     6     5   river_stage_1     1005.   35.5
      1     6     6   river_stage_1     1006.   35.4  riv1_c6
      1     6     7   river_stage_1     1007.   35.3  riv1_c7
      1     5     8   river_stage_1     1008.   35.2
      1     5     9   river_stage_1     1009.   35.1
      1     5    10   river_stage_1     1010.   35.0
      1    10     1   river_stage_2     1001.   36.9  riv2_upper
      1     9     2   river_stage_2     1002.   36.8  riv2_upper
      1     8     3   river_stage_2     1003.   36.7  riv2_upper
      1     7     4   river_stage_2     1004.   36.6
      1     7     5   river_stage_2     1005.   36.5
      1     6     6   river_stage_2     1006.   36.4  riv2_c6
      1     6     7   river_stage_2     1007.   36.3  riv2_c7
      1     7     8   river_stage_2     1008.   36.2
      1     7     9   river_stage_2     1009.   36.1
      1     7    10   river_stage_2     1010.   36.0
END PERIOD
```

Available Observation Types

Example Observation Input File

```
BEGIN OPTIONS
  DIGITS 7
  PRINT_INPUT
END OPTIONS

BEGIN CONTINUOUS FILEOUT my_model.riv.csv
# obsname        type  ID
  rv1-5-4        RIV   1    5    4
  rv1-6-5        RIV   1    6    5
  rv1-c7         RIV   riv1_c7     # flow at boundary "riv1_c7"
  rv2-7-4        RIV   1    7    4
  rv2-8-5        RIV   1    7    5
  rv2-9-6        RIV   1    6    6
END CONTINUOUS

BEGIN CONTINUOUS FILEOUT my_model.riv.flows.csv
# obsname         type  ID
  rv1-3-1         RIV   1    3    1
  rv1-4-2         RIV   1    4    2
  rv1-5-3         RIV   1    5    3
  rv1-c6          RIV   riv1_c6
  rv2-upper       RIV   riv2_upper
END CONTINUOUS
```

1.3.23 GWF-SFR

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [AUXILIARY <auxiliary(naux)>]
  [BOUNDNAMES]
  [PRINT_INPUT]
  [PRINT_STAGE]
  [PRINT_FLOWS]
  [SAVE_FLOWS]
  [STAGE FILEOUT <stagefile>]
  [BUDGET FILEOUT <budgetfile>]
  [PACKAGE_CONVERGENCE FILEOUT <package_convergence_filename>]
  [TS6 FILEIN <ts6_filename>]
  [OBS6 FILEIN <obs6_filename>]
  [MOVER]
  [MAXIMUM_PICARD_ITERATIONS <maximum_picard_iterations>]
  [MAXIMUM_ITERATIONS <maximum_iterations>]
  [MAXIMUM_DEPTH_CHANGE <maximum_depth_change>]
  [UNIT_CONVERSION <unit_conversion>]
END OPTIONS
```

```
  BEGIN DIMENSIONS
    NREACHES <nreaches>
  END DIMENSIONS
```

```
  BEGIN PACKAGEDATA
    <rno> <cellid(ncelldim)> <rlen> <rwid> <rgrd> <rtp> <rbth> <rhk> <man> <ncon>
↪<ustrf> <ndv> [<aux(naux)>] [<boundname>]
    <rno> <cellid(ncelldim)> <rlen> <rwid> <rgrd> <rtp> <rbth> <rhk> <man> <ncon>
↪<ustrf> <ndv> [<aux(naux)>] [<boundname>]
    ...
  END PACKAGEDATA
```

```
  BEGIN CONNECTIONDATA
    <rno> <ic(ncon(rno))>
    <rno> <ic(ncon(rno))>
    ...
  END CONNECTIONDATA
```

```
  BEGIN DIVERSIONS
    <rno> <idv> <iconr> <cprior>
    <rno> <idv> <iconr> <cprior>
    ...
  END DIVERSIONS
```

FOR ANY STRESS PERIOD

```
  BEGIN PERIOD <iper>
    <rno> <sfrsetting>
    <rno> <sfrsetting>
    ...
  END PERIOD
```

Explanation of Variables

Block: OPTIONS

- `auxiliary` defines an array of one or more auxiliary variable names. There is no limit on the number of auxiliary variables that can be provided on this line; however, lists of information provided in subsequent blocks must have a column of data for each auxiliary variable name defined here. The number of auxiliary variables detected on this line determines the value for naux. Comments cannot be provided anywhere on this line as they will be interpreted as auxiliary variable names. Auxiliary variables may not be used by the package, but they will be available for use by other parts of the program. The program will terminate with an error if auxiliary variables are specified on more than one line in the options block.
- `BOUNDNAMES` keyword to indicate that boundary names may be provided with the list of stream reach cells.
- `PRINT_INPUT` keyword to indicate that the list of stream reach information will be written to the listing file immediately after it is read.
- `PRINT_STAGE` keyword to indicate that the list of stream reach stages will be printed to the listing file for every stress period in which "HEAD PRINT" is specified in Output Control. If there is no Output Control option and PRINT_STAGE is specified, then stages are printed for the last time step of each stress period.
- `PRINT_FLOWS` keyword to indicate that the list of stream reach flow rates will be printed to the listing file for every stress period time step in which "BUDGET PRINT" is specified in Output Control. If there is no Output

Control option and "PRINT_FLOWS" is specified, then flow rates are printed for the last time step of each stress period.

- SAVE_FLOWS keyword to indicate that stream reach flow terms will be written to the file specified with "BUDGET FILEOUT" in Output Control.
- STAGE keyword to specify that record corresponds to stage.
- stagefile name of the binary output file to write stage information.
- BUDGET keyword to specify that record corresponds to the budget.
- FILEOUT keyword to specify that an output filename is expected next.
- budgetfile name of the binary output file to write budget information.
- PACKAGE_CONVERGENCE keyword to specify that record corresponds to the package convergence comma spaced values file.
- package_convergence_filename name of the comma spaced values output file to write package convergence information.
- TS6 keyword to specify that record corresponds to a time-series file.
- FILEIN keyword to specify that an input filename is expected next.
- ts6_filename defines a time-series file defining time series that can be used to assign time-varying values. See the "Time-Variable Input" section for instructions on using the time-series capability.
- OBS6 keyword to specify that record corresponds to an observations file.
- obs6_filename name of input file to define observations for the SFR package. See the "Observation utility" section for instructions for preparing observation input files. Tables ref{table:gwf-obstypetable} and ref{table:gwt-obstypetable} lists observation type(s) supported by the SFR package.
- MOVER keyword to indicate that this instance of the SFR Package can be used with the Water Mover (MVR) Package. When the MOVER option is specified, additional memory is allocated within the package to store the available, provided, and received water.
- maximum_picard_iterations value that defines the maximum number of Streamflow Routing picard iterations allowed when solving for reach stages and flows as part of the GWF formulate step. Picard iterations are used to minimize differences in SFR package results between subsequent GWF picard (non-linear) iterations as a result of non-optimal reach numbering. If reaches are numbered in order, from upstream to downstream, MAXIMUM_PICARD_ITERATIONS can be set to 1 to reduce model run time. By default, MAXIMUM_PICARD_ITERATIONS is equal to 100.
- maximum_iterations value that defines the maximum number of Streamflow Routing Newton-Raphson iterations allowed for a reach. By default, MAXIMUM_ITERATIONS is equal to 100.
- maximum_depth_change value that defines the depth closure tolerance. By default, DMAXCHG is equal to 1 x 10-5.
- unit_conversion value (or conversion factor) that is used in calculating stream depth for stream reach. A constant of 1.486 is used for flow units of cubic feet per second, and a constant of 1.0 is used for units of cubic meters per second. The constant must be multiplied by 86,400 when using time units of days in the simulation.

Block: DIMENSIONS

- `nreaches` integer value specifying the number of stream reaches. There must be NREACHES entries in the PACKAGEDATA block.

Block: PACKAGEDATA

- `rno` integer value that defines the reach number associated with the specified PACKAGEDATA data on the line. RNO must be greater than zero and less than or equal to NREACHES. Reach information must be specified for every reach or the program will terminate with an error. The program will also terminate with an error if information for a reach is specified more than once.
- `cellid` The keyword "NONE" must be specified for reaches that are not connected to an underlying GWF cell. The keyword "NONE" is used for reaches that are in cells that have IDOMAIN values less than one or are in areas not covered by the GWF model grid. Reach-aquifer flow is not calculated if the keyword "NONE" is specified.
- `rlen` real value that defines the reach length. RLEN must be greater than zero.
- `rwid` real value that defines the reach width. RWID must be greater than zero.
- `rgrd` real value that defines the stream gradient (slope) across the reach. RGRD must be greater than zero.
- `rtp` real value that defines the top elevation of the reach streambed.
- `rbth` real value that defines the thickness of the reach streambed. RBTH can be any value if CELLID is "NONE". Otherwise, RBTH must be greater than zero.
- `rhk` real value that defines the hydraulic conductivity of the reach streambed. RHK can be any positive value if CELLID is "NONE". Otherwise, RHK must be greater than zero.
- `man` real or character value that defines the Manning's roughness coefficient for the reach. MAN must be greater than zero. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `ncon` integer value that defines the number of reaches connected to the reach. If a value of zero is specified for NCON an entry for RNO is still required in the subsequent CONNECTIONDATA block.
- `ustrf` real value that defines the fraction of upstream flow from each upstream reach that is applied as upstream inflow to the reach. The sum of all USTRF values for all reaches connected to the same upstream reach must be equal to one and USTRF must be greater than or equal to zero. If the Options block includes a TIMESERIES-FILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `ndv` integer value that defines the number of downstream diversions for the reach.
- `aux` represents the values of the auxiliary variables for each stream reach. The values of auxiliary variables must be present for each stream reach. The values must be specified in the order of the auxiliary variables specified in the OPTIONS block. If the package supports time series and the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `boundname` name of the stream reach cell. BOUNDNAME is an ASCII character variable that can contain as many as 40 characters. If BOUNDNAME contains spaces in it, then the entire name must be enclosed within single quotes.

Block: CONNECTIONDATA

- `rno` integer value that defines the reach number associated with the specified CONNECTIONDATA data on the line. RNO must be greater than zero and less than or equal to NREACHES. Reach connection information must be specified for every reach or the program will terminate with an error. The program will also terminate with an error if connection information for a reach is specified more than once.
- `ic` integer value that defines the reach number of the reach connected to the current reach and whether it is connected to the upstream or downstream end of the reach. Negative IC numbers indicate connected reaches are connected to the downstream end of the current reach. Positive IC numbers indicate connected reaches are connected to the upstream end of the current reach. The absolute value of IC must be greater than zero and less than or equal to NREACHES.

Block: DIVERSIONS

- `rno` integer value that defines the reach number associated with the specified DIVERSIONS data on the line. RNO must be greater than zero and less than or equal to NREACHES. Reach diversion information must be specified for every reach with a NDV value greater than 0 or the program will terminate with an error. The program will also terminate with an error if diversion information for a given reach diversion is specified more than once.
- `idv` integer value that defines the downstream diversion number for the diversion for reach RNO. IDV must be greater than zero and less than or equal to NDV for reach RNO.
- `iconr` integer value that defines the downstream reach that will receive the diverted water. IDV must be greater than zero and less than or equal to NREACHES. Furthermore, reach ICONR must be a downstream connection for reach RNO.
- `cprior` character string value that defines the the prioritization system for the diversion, such as when insufficient water is available to meet all diversion stipulations, and is used in conjunction with the value of FLOW value specified in the STRESS_PERIOD_DATA section. Available diversion options include: (1) CPRIOR = "FRACTION", then the amount of the diversion is computed as a fraction of the streamflow leaving reach RNO (Q_{DS}); in this case, 0.0 DIVFLOW 1.0. (2) CPRIOR = "EXCESS", a diversion is made only if Q_{DS} for reach RNO exceeds the value of DIVFLOW. If this occurs, then the quantity of water diverted is the excess flow (Q_{DS} - DIVFLOW) and Q_{DS} from reach RNO is set equal to DIVFLOW. This represents a flood-control type of diversion, as described by Danskin and Hanson (2002). (3) CPRIOR = "THRESHOLD", then if Q_{DS} in reach RNO is less than the specified diversion flow (DIVFLOW), no water is diverted from reach RNO. If Q_{DS} in reach RNO is greater than or equal to (DIVFLOW), (DIVFLOW) is diverted and Q_{DS} is set to the remainder (Q_{DS} - DIVFLOW)). This approach assumes that once flow in the stream is sufficiently low, diversions from the stream cease, and is the "priority" algorithm that originally was programmed into the STR1 Package (Prudic, 1989). (4) CPRIOR = "UPTO" – if Q_{DS} in reach RNO is greater than or equal to the specified diversion flow (DIVFLOW), Q_{DS} is reduced by DIVFLOW. If Q_{DS} in reach RNO is less than (DIVFLOW), DIVFLOW is set to Q_{DS} and there will be no flow available for reaches connected to downstream end of reach RNO.

Block: PERIOD

- `iper` integer value specifying the starting stress period number for which the data specified in the PERIOD block apply. IPER must be less than or equal to NPER in the TDIS Package and greater than zero. The IPER value assigned to a stress period block must be greater than the IPER value assigned for the previous PERIOD block. The information specified in the PERIOD block will continue to apply for all subsequent stress periods, unless the program encounters another PERIOD block.
- `rno` integer value that defines the reach number associated with the specified PERIOD data on the line. RNO must be greater than zero and less than or equal to NREACHES.
- `sfrsetting` line of information that is parsed into a keyword and values. Keyword values that can be used to start the SFRSETTING string include: STATUS, MANNING, STAGE, INFLOW, RAINFALL, EVAPORATION, RUNOFF, DIVERSION, UPSTREAM_FRACTION, and AUXILIARY.

```
STATUS <status>
MANNING <manning>
STAGE <stage>
INFLOW <inflow>
RAINFALL <rainfall>
EVAPORATION <evaporation>
RUNOFF <runoff>
DIVERSION <idv> <divflow>
UPSTREAM_FRACTION <upstream_fraction>
AUXILIARY <auxname> <auxval>
```

- `status` keyword option to define stream reach status. STATUS can be ACTIVE, INACTIVE, or SIMPLE. The SIMPLE STATUS option simulates streamflow using a user-specified stage for a reach or a stage set to the top of the reach (depth = 0). In cases where the simulated leakage calculated using the specified stage exceeds the sum of inflows to the reach, the stage is set to the top of the reach and leakage is set equal to the sum of inflows. Upstream fractions should be changed using the UPSTREAM_FRACTION SFRSETTING if the status for one or more reaches is changed to ACTIVE or INACTIVE. For example, if one of two downstream connections for a reach is inactivated, the upstream fraction for the active and inactive downstream reach should be changed to 1.0 and 0.0, respectively, to ensure that the active reach receives all of the downstream outflow from the upstream reach. By default, STATUS is ACTIVE.
- `manning` real or character value that defines the Manning's roughness coefficient for the reach. MANNING must be greater than zero. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `stage` real or character value that defines the stage for the reach. The specified STAGE is only applied if the reach uses the simple routing option. If STAGE is not specified for reaches that use the simple routing option, the specified stage is set to the top of the reach. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `inflow` real or character value that defines the volumetric inflow rate for the streamflow routing reach. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value. By default, inflow rates are zero for each reach.
- `rainfall` real or character value that defines the volumetric rate per unit area of water added by precipitation directly on the streamflow routing reach. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value. By default, rainfall rates are zero for each reach.
- `evaporation` real or character value that defines the volumetric rate per unit area of water subtracted by

evaporation from the streamflow routing reach. A positive evaporation rate should be provided. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value. If the volumetric evaporation rate for a reach exceeds the sources of water to the reach (upstream and specified inflows, rainfall, and runoff but excluding groundwater leakage into the reach) the volumetric evaporation rate is limited to the sources of water to the reach. By default, evaporation rates are zero for each reach.

- `runoff` real or character value that defines the volumetric rate of diffuse overland runoff that enters the streamflow routing reach. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value. If the volumetric runoff rate for a reach is negative and exceeds inflows to the reach (upstream and specified inflows, and rainfall but excluding groundwater leakage into the reach) the volumetric runoff rate is limited to inflows to the reach and the volumetric evaporation rate for the reach is set to zero. By default, runoff rates are zero for each reach.
- `DIVERSION` keyword to indicate diversion record.
- `idv` an integer value specifying which diversion of reach RNO that DIVFLOW is being specified for. Must be less or equal to ndv for the current reach (RNO).
- `divflow` real or character value that defines the volumetric diversion (DIVFLOW) rate for the streamflow routing reach. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `upstream_fraction` real value that defines the fraction of upstream flow (USTRF) from each upstream reach that is applied as upstream inflow to the reach. The sum of all USTRF values for all reaches connected to the same upstream reach must be equal to one.
- `AUXILIARY` keyword for specifying auxiliary variable.
- `auxname` name for the auxiliary variable to be assigned AUXVAL. AUXNAME must match one of the auxiliary variable names defined in the OPTIONS block. If AUXNAME does not match one of the auxiliary variable names defined in the OPTIONS block the data are ignored.
- `auxval` value for the auxiliary variable. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.

Example Input File

```
  BEGIN OPTIONS
    UNIT_CONVERSION 1.486
    BOUNDNAMES
    PRINT_STAGE
    PRINT_FLOWS
    STAGE FILEOUT sfr-1.stage.bin
    BUDGET FILEOUT sfr-1.cbc
  END OPTIONS

  #dimension block is required
  BEGIN DIMENSIONS
    NREACHES 37
  END DIMENSIONS

  BEGIN PACKAGEDATA
  #rno k  i j rlen rwid        rgrd      rtp    rbth     rhk    man   ncon ustrf  ndv ␣
↪boundname
      1 1  1 1 4500.  12    8.67E-04 1093.048    3.0 0.00003  0.03      1   1.0    0 ␣
↪ reach1
```

(continues on next page)

(continued from previous page)

```
      2 1  2 2 7000.  12    8.67E-04 1088.059    3.0 0.00003  0.03     2   1.0    0 ␣
↪  reach2
      3 1  3 3 6000.  12    8.67E-04 1082.419    3.0 0.00003  0.03     2   1.0    0 ␣
↪  reach3
      4 1  3 4 5550.  12    8.67E-04 1077.408    3.0 0.00003  0.03     3   1.0    1 ␣
↪  reach4
      5 1  4 5 6500.  12    9.43E-04 1071.934    3.0 0.00003  0.03     2   1.0    0
      6 1  5 6 5000.  12    9.43E-04 1066.509    3.0 0.00003  0.03     2   1.0    0
      7 1  6 6 5000.  12    9.43E-04 1061.792    3.0 0.00003  0.03     2   1.0    0
      8 1  7 6 5000.  12    9.43E-04 1057.075    3.0 0.00003  0.03     2   1.0    0
      9 1  8 6 5000.  12    9.43E-04 1052.359    3.0 0.00003  0.03     2   1.0    0
     10 1  3 5 5000.  10    5.45E-04 1073.636    2.0 0.00003  0.03     2   0.0    0 ␣
↪  canal
     11 1  3 6 5000.  10    5.45E-04 1070.909    2.0 0.00003  0.03     2   1.0    0 ␣
↪  canal
     12 1  3 7 4500.  10    5.45E-04 1068.318    2.0 0.00003  0.03     2   1.0    0 ␣
↪  canal
     13 1  4 8 6000.  10    5.45E-04 1065.455    2.0 0.00003  0.03     2   1.0    0 ␣
↪  canal
     14 1  5 8 5000.  10    5.45E-04 1062.455    2.0 0.00003  0.03     2   1.0    0 ␣
↪  canal
     15 1  6 8 2000.  10    5.45E-04 1060.545    2.0 0.00003  0.03     2   1.0    0 ␣
↪  canal
     16 1  510 2500.  10    1.81E-03 1077.727    3.0 0.00003  0.03     1   1.0    0
     17 1  5 9 5000.  10    1.81E-03 1070.909    3.0 0.00003  0.03     2   1.0    0
     18 1  6 8 3500.  10    1.81E-03 1063.182    3.0 0.00003  0.03     2   1.0    0
     19 1  6 8 4000.  15    1.00E-03 1058.000    3.0 0.00003  0.03     3   1.0    0
     20 1  7 7 5000.  15    1.00E-03 1053.500    3.0 0.00003  0.03     2   1.0    0
     21 1  8 7 3500.  15    1.00E-03 1049.250    3.0 0.00003  0.03     2   1.0    0
     22 1  8 6 2500.  15    1.00E-03 1046.250    3.0 0.00003  0.03     2   1.0    0
     23 1  9 6 5000.  12    9.09E-04 1042.727    3.0 0.00003  0.03     3   1.0    0
     24 1 10 7 5000.  12    9.09E-04 1038.182    3.0 0.00003  0.03     2   1.0    0
     25 1 11 7 5000.  12    9.09E-04 1033.636    3.0 0.00003  0.03     2   1.0    0
     26 1 12 7 5000.  12    9.09E-04 1029.091    3.0 0.00003  0.03     2   1.0    0
     27 1 13 7 2000.  12    9.09E-04 1025.909    3.0 0.00003  0.03     2   1.0    0
     28 1 14 9 5000.  55    9.67E-04 1037.581    3.0 0.00006  0.025    1   1.0    0
     29 1 13 8 5500.  55    9.67E-04 1032.500    3.0 0.00006  0.025    2   1.0    0
     30 1 13 7 5000.  55    9.67E-04 1027.419    3.0 0.00006  0.025    2   1.0    0
     31 1 13 6 5000.  40    1.25E-03 1021.875    3.0 0.00006  0.025    3   1.0    0
     32 1 13 5 5000.  40    1.25E-03 1015.625    3.0 0.00006  0.025    2   1.0    0
     33 1 13 4 5000.  40    1.25E-03 1009.375    3.0 0.00006  0.025    2   1.0    0
     34 1 13 3 5000.  40    1.25E-03 1003.125    3.0 0.00006  0.025    2   1.0    0
     35 1 13 2 5000.  40    1.25E-03 996.8750    3.0 0.00006  0.025    2   1.0    0
     36 1 13 1 3000.  40    1.25E-03 991.8750    3.0 0.00006  0.025    2   1.0    0
     37 none   5000.  40    1.25E-03 985.6250    3.0 0.00006  0.025    1   1.0    0
  END PACKAGEDATA

  BEGIN CONNECTIONDATA
  #rno ic1 ic2 ic3
     1  -2
     2   1  -3
     3   2  -4
     4   3  -5 -10
     5   4  -6
     6   5  -7
     7   6  -8
     8   7  -9
```

(continues on next page)

(continued from previous page)

```
      9   8 -23
     10   4 -11
     11  10 -12
     12  11 -13
     13  12 -14
     14  13 -15
     15  14 -19
     16 -17
     17  16 -18
     18  17 -19
     19  15  18 -20
     20  19 -21
     21  20 -22
     22  21 -23
     23   9  22 -24
     24  23 -25
     25  24 -26
     26  25 -27
     27  26 -31
     28 -29
     29  28 -30
     30  29 -31
     31  27  30 -32
     32  31 -33
     33  32 -34
     34  33 -35
     35  34 -36
     36  35 -37
     37  36
  END CONNECTIONDATA

  BEGIN DIVERSIONS
  #  rno idv iconr cprior
       4   1    10 UPTO
  END DIVERSIONS

  BEGIN PERIOD 1
  # rno sfrsetting
      1 inflow 25.
     16 inflow 10.
     28 inflow 150.
      4 diversion 1 10.
     10 status simple
     11 status simple
     12 status simple
     13 status simple
     14 status simple
     15 status simple
     10 stage 1075.5454
     11 stage 1072.6363
     12 stage 1069.8727
     13 stage 1066.8181
     14 stage 1063.6181
     15 stage 1061.5818
  END PERIOD
```

Available Observation Types

Example Observation Input File

```
  BEGIN OPTIONS
    DIGITS 8
    PRINT_INPUT
  END OPTIONS

  BEGIN CONTINUOUS FILEOUT my_model.sfr.csv
  # obsname     obstype          id
    gage1stage  STAGE            reach4
    gage2stage  STAGE            7
    gage2inflow INFLOW           7
    gage2disch  DOWNSTREAM-FLOW  7
    gage3stage  STAGE            14
  END CONTINUOUS

  BEGIN CONTINUOUS FILEOUT my_model.sfr.leakage.csv
  # obsname    obstype           id
    leak1      SFR               reach1
    leak10     SFR               10
    leak11     SFR               11
    leak12     SFR               12
    leak13     SFR               13
    leak14     SFR               14
    leak15     SFR               15
    leakcanal  SFR               canal  #Sum of flows between canal reaches and␣
↪groundwater
  END CONTINUOUS
```

1.3.24 GWF-STO

Structure of Blocks

FOR EACH SIMULATION

```
  BEGIN OPTIONS
    [SAVE_FLOWS]
    [STORAGECOEFFICIENT]
  END OPTIONS
```

```
  BEGIN GRIDDATA
    ICONVERT [LAYERED]
          <iconvert(nodes)> -- READARRAY
    SS [LAYERED]
          <ss(nodes)> -- READARRAY
    SY [LAYERED]
          <sy(nodes)> -- READARRAY
  END GRIDDATA
```

FOR ANY STRESS PERIOD

```
  BEGIN PERIOD <iper>
    [STEADY-STATE]
```

(continues on next page)

(continued from previous page)

```
    [TRANSIENT]
  END PERIOD
```

Explanation of Variables

Block: OPTIONS

- SAVE_FLOWS keyword to indicate that cell-by-cell flow terms will be written to the file specified with "BUDGET SAVE FILE" in Output Control.
- STORAGECOEFFICIENT keyword to indicate that the SS array is read as storage coefficient rather than specific storage.

Block: GRIDDATA

- iconvert is a flag for each cell that specifies whether or not a cell is convertible for the storage calculation. 0 indicates confined storage is used. >0 indicates confined storage is used when head is above cell top and a mixed formulation of unconfined and confined storage is used when head is below cell top.
- ss is specific storage (or the storage coefficient if STORAGECOEFFICIENT is specified as an option). Specific storage values must be greater than or equal to 0. If the CSUB Package is included in the GWF model, specific storage must be zero for every cell.
- sy is specific yield. Specific yield values must be greater than or equal to 0. Specific yield does not have to be specified if there are no convertible cells (ICONVERT=0 in every cell).

Block: PERIOD

- iper integer value specifying the starting stress period number for which the data specified in the PERIOD block apply. IPER must be less than or equal to NPER in the TDIS Package and greater than zero. The IPER value assigned to a stress period block must be greater than the IPER value assigned for the previous PERIOD block. The information specified in the PERIOD block will continue to apply for all subsequent stress periods, unless the program encounters another PERIOD block.
- STEADY-STATE keyword to indicate that stress period IPER is steady-state. Steady-state conditions will apply until the TRANSIENT keyword is specified in a subsequent BEGIN PERIOD block. If the CSUB Package is included in the GWF model, only the first and last stress period can be steady-state.
- TRANSIENT keyword to indicate that stress period IPER is transient. Transient conditions will apply until the STEADY-STATE keyword is specified in a subsequent BEGIN PERIOD block.

Example Input File

```
  BEGIN OPTIONS
    SAVE_FLOWS
  END OPTIONS

  BEGIN GRIDDATA
    #cell storage conversion 0:confined, 1:convertible
    ICONVERT
      constant 1
```

(continues on next page)

(continued from previous page)

```
  #specific storage (for all model cells)
  SS
    constant 1.e-5
  #specific yield (specified by layer because of LAYERED keyword)
  SY LAYERED
    constant 0.2
    constant 0.15
    constant 0.15
END GRIDDATA

BEGIN PERIOD 1
  STEADY-STATE
END PERIOD

BEGIN PERIOD 2
  TRANSIENT
END PERIOD

#stress period 3 will be transient because
#a BEGIN PERIOD block is not provided.

BEGIN PERIOD 4
  STEADY-STATE
END PERIOD
```

1.3.25 GWF-UZF

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [AUXILIARY <auxiliary(naux)>]
  [AUXMULTNAME <auxmultname>]
  [BOUNDNAMES]
  [PRINT_INPUT]
  [PRINT_FLOWS]
  [SAVE_FLOWS]
  [BUDGET FILEOUT <budgetfile>]
  [PACKAGE_CONVERGENCE FILEOUT <package_convergence_filename>]
  [TS6 FILEIN <ts6_filename>]
  [OBS6 FILEIN <obs6_filename>]
  [MOVER]
  [SIMULATE_ET]
  [LINEAR_GWET]
  [SQUARE_GWET]
  [SIMULATE_GWSEEP]
  [UNSAT_ETWC]
  [UNSAT_ETAE]
END OPTIONS
```

```
BEGIN DIMENSIONS
  NUZFCELLS <nuzfcells>
  NTRAILWAVES <ntrailwaves>
```

(continues on next page)

(continued from previous page)

```
  NWAVESETS <nwavesets>
END DIMENSIONS
```

```
BEGIN PACKAGEDATA
  <iuzno> <cellid(ncelldim)> <landflag> <ivertcon> <surfdep> <vks> <thtr> <thts>
↪<thti> <eps> [<boundname>]
  <iuzno> <cellid(ncelldim)> <landflag> <ivertcon> <surfdep> <vks> <thtr> <thts>
↪<thti> <eps> [<boundname>]
  ...
END PACKAGEDATA
```

FOR ANY STRESS PERIOD

```
BEGIN PERIOD <iper>
  <iuzno> <finf> <pet> <extdp> <extwc> <ha> <hroot> <rootact> [<aux(naux)>]
  <iuzno> <finf> <pet> <extdp> <extwc> <ha> <hroot> <rootact> [<aux(naux)>]
  ...
END PERIOD
```

Explanation of Variables

Block: OPTIONS

- `auxiliary` defines an array of one or more auxiliary variable names. There is no limit on the number of auxiliary variables that can be provided on this line; however, lists of information provided in subsequent blocks must have a column of data for each auxiliary variable name defined here. The number of auxiliary variables detected on this line determines the value for naux. Comments cannot be provided anywhere on this line as they will be interpreted as auxiliary variable names. Auxiliary variables may not be used by the package, but they will be available for use by other parts of the program. The program will terminate with an error if auxiliary variables are specified on more than one line in the options block.
- `auxmultname` name of auxiliary variable to be used as multiplier of GWF cell area used by UZF cell.
- `BOUNDNAMES` keyword to indicate that boundary names may be provided with the list of UZF cells.
- `PRINT_INPUT` keyword to indicate that the list of UZF information will be written to the listing file immediately after it is read.
- `PRINT_FLOWS` keyword to indicate that the list of UZF flow rates will be printed to the listing file for every stress period time step in which "BUDGET PRINT" is specified in Output Control. If there is no Output Control option and "PRINT_FLOWS" is specified, then flow rates are printed for the last time step of each stress period.
- `SAVE_FLOWS` keyword to indicate that UZF flow terms will be written to the file specified with "BUDGET FILEOUT" in Output Control.
- `BUDGET` keyword to specify that record corresponds to the budget.
- `FILEOUT` keyword to specify that an output filename is expected next.
- `budgetfile` name of the binary output file to write budget information.
- `PACKAGE_CONVERGENCE` keyword to specify that record corresponds to the package convergence comma spaced values file.
- `package_convergence_filename` name of the comma spaced values output file to write package convergence information.
- `TS6` keyword to specify that record corresponds to a time-series file.

- FILEIN keyword to specify that an input filename is expected next.
- ts6_filename defines a time-series file defining time series that can be used to assign time-varying values. See the "Time-Variable Input" section for instructions on using the time-series capability.
- OBS6 keyword to specify that record corresponds to an observations file.
- obs6_filename name of input file to define observations for the UZF package. See the "Observation utility" section for instructions for preparing observation input files. Tables ref{table:gwf-obstypetable} and ref{table:gwt-obstypetable} lists observation type(s) supported by the UZF package.
- MOVER keyword to indicate that this instance of the UZF Package can be used with the Water Mover (MVR) Package. When the MOVER option is specified, additional memory is allocated within the package to store the available, provided, and received water.
- SIMULATE_ET keyword specifying that ET in the unsaturated (UZF) and saturated zones (GWF) will be simulated. ET can be simulated in the UZF cell and not the GWF cell by omitting keywords LINEAR_GWET and SQUARE_GWET.
- LINEAR_GWET keyword specifying that groundwater ET will be simulated using the original ET formulation of MODFLOW-2005.
- SQUARE_GWET keyword specifying that groundwater ET will be simulated by assuming a constant ET rate for groundwater levels between land surface (TOP) and land surface minus the ET extinction depth (TOP-EXTDP). Groundwater ET is smoothly reduced from the PET rate to zero over a nominal interval at TOP-EXTDP.
- SIMULATE_GWSEEP keyword specifying that groundwater discharge (GWSEEP) to land surface will be simulated. Groundwater discharge is nonzero when groundwater head is greater than land surface.
- UNSAT_ETWC keyword specifying that ET in the unsaturated zone will be simulated as a function of the specified PET rate while the water content (THETA) is greater than the ET extinction water content (EXTWC).
- UNSAT_ETAE keyword specifying that ET in the unsaturated zone will be simulated using a capillary pressure based formulation. Capillary pressure is calculated using the Brooks-Corey retention function.

Block: DIMENSIONS

- nuzfcells is the number of UZF cells. More than one UZF cell can be assigned to a GWF cell; however, only one GWF cell can be assigned to a single UZF cell. If more than one UZF cell is assigned to a GWF cell, then an auxiliary variable should be used to reduce the surface area of the UZF cell with the AUXMULTNAME option.
- ntrailwaves is the number of trailing waves. A recommended value of 7 can be used for NTRAILWAVES. This value can be increased to lower mass balance error in the unsaturated zone.
- nwavesets is the number of wave sets. A recommended value of 40 can be used for NWAVESETS. This value can be increased if more waves are required to resolve variations in water content within the unsaturated zone.

Block: PACKAGEDATA

- `iuzno` integer value that defines the UZF cell number associated with the specified PACKAGEDATA data on the line. IUZNO must be greater than zero and less than or equal to NUZFCELLS. UZF information must be specified for every UZF cell or the program will terminate with an error. The program will also terminate with an error if information for a UZF cell is specified more than once.
- `cellid` is the cell identifier, and depends on the type of grid that is used for the simulation. For a structured grid that uses the DIS input file, CELLID is the layer, row, and column. For a grid that uses the DISV input file, CELLID is the layer and CELL2D number. If the model uses the unstructured discretization (DISU) input file, CELLID is the node number for the cell.
- `landflag` integer value set to one for land surface cells indicating that boundary conditions can be applied and data can be specified in the PERIOD block. A value of 0 specifies a non-land surface cell.
- `ivertcon` integer value set to specify underlying UZF cell that receives water flowing to bottom of cell. If unsaturated zone flow reaches the water table before the cell bottom, then water is added to the GWF cell instead of flowing to the underlying UZF cell. A value of 0 indicates the UZF cell is not connected to an underlying UZF cell.
- `surfdep` is the surface depression depth of the UZF cell.
- `vks` is the vertical saturated hydraulic conductivity of the UZF cell.
- `thtr` is the residual (irreducible) water content of the UZF cell.
- `thts` is the saturated water content of the UZF cell.
- `thti` is the initial water content of the UZF cell.
- `eps` is the epsilon exponent of the UZF cell.
- `boundname` name of the UZF cell cell. BOUNDNAME is an ASCII character variable that can contain as many as 40 characters. If BOUNDNAME contains spaces in it, then the entire name must be enclosed within single quotes.

Block: PERIOD

- `iper` integer value specifying the starting stress period number for which the data specified in the PERIOD block apply. IPER must be less than or equal to NPER in the TDIS Package and greater than zero. The IPER value assigned to a stress period block must be greater than the IPER value assigned for the previous PERIOD block. The information specified in the PERIOD block will continue to apply for all subsequent stress periods, unless the program encounters another PERIOD block.
- `iuzno` integer value that defines the UZF cell number associated with the specified PERIOD data on the line.
- `finf` real or character value that defines the applied infiltration rate of the UZF cell (LT-1). If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `pet` real or character value that defines the potential evapotranspiration rate of the UZF cell and specified GWF cell. Evapotranspiration is first removed from the unsaturated zone and any remaining potential evapotranspiration is applied to the saturated zone. If IVERTCON is greater than zero then residual potential evapotranspiration not satisfied in the UZF cell is applied to the underlying UZF and GWF cells. PET is always specified, but is only used if SIMULATE_ET is specified in the OPTIONS block. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.

- `extdp` real or character value that defines the evapotranspiration extinction depth of the UZF cell. If IVERTCON is greater than zero and EXTDP extends below the GWF cell bottom then remaining potential evapotranspiration is applied to the underlying UZF and GWF cells. EXTDP is always specified, but is only used if SIMULATE_ET is specified in the OPTIONS block. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `extwc` real or character value that defines the evapotranspiration extinction water content of the UZF cell. EXTWC is always specified, but is only used if SIMULATE_ET and UNSAT_ETWC are specified in the OPTIONS block. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `ha` real or character value that defines the air entry potential (head) of the UZF cell. HA is always specified, but is only used if SIMULATE_ET and UNSAT_ETAE are specified in the OPTIONS block. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `hroot` real or character value that defines the root potential (head) of the UZF cell. HROOT is always specified, but is only used if SIMULATE_ET and UNSAT_ETAE are specified in the OPTIONS block. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `rootact` real or character value that defines the root activity function of the UZF cell. ROOTACT is the length of roots in a given volume of soil divided by that volume. Values range from 0 to about 3 cm-2, depending on the plant community and its stage of development. ROOTACT is always specified, but is only used if SIMULATE_ET and UNSAT_ETAE are specified in the OPTIONS block. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `aux` represents the values of the auxiliary variables for each UZF. The values of auxiliary variables must be present for each UZF. The values must be specified in the order of the auxiliary variables specified in the OPTIONS block. If the package supports time series and the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.

Example Input File

```
BEGIN OPTIONS
  OBS6 UZF.obs
  SIMULATE_ET
  UNSAT_ETWC
  LINEAR_GWET
END OPTIONS

BEGIN DIMENSIONS
  NUZFCELLS    10
  NTRAILWAVES  7
  NWAVESETS    40
END DIMENSIONS

BEGIN PACKAGEDATA
  1  1 1 1 1 1.0 1.0 0.05 0.35 0.1 4.0
  2  1 1 2 1 1.0 1.0 0.05 0.35 0.1 4.0
  3  1 1 3 1 1.0 1.0 0.05 0.35 0.1 4.0
  4  1 1 4 1 1.0 1.0 0.05 0.35 0.1 4.0
  5  1 1 5 1 1.0 1.0 0.05 0.35 0.1 4.0
```

(continues on next page)

(continued from previous page)

```
  6  1 1 6 1 1.0 1.0 0.05 0.35 0.1 4.0
  7  1 1 7 1 1.0 1.0 0.05 0.35 0.1 4.0
  8  1 1 8 1 1.0 1.0 0.05 0.35 0.1 4.0
  9  1 1 9 1 1.0 1.0 0.05 0.35 0.1 4.0
  10 1 1 10 1 1.0 1.0 0.05 0.35 0.1 4.0
END PACKAGEDATA

BEGIN PERIOD 1
  2 0.00005 0.00002 2.0 0.10
  3 0.00008 0.00002 2.0 0.10
  4 0.00009 0.00002 2.0 0.10
  5 0.0001  0.00002 2.0 0.10
  6 0.0001 0.00002 2.0 0.10
  7 0.00009 0.00002 2.0 0.10
  8 0.00008 0.00002 2.0 0.10
  9 0.00005 0.00002 2.0 0.10
END PERIOD

BEGIN PERIOD 2
  2 0.00009 0.00003 2.0 0.10
  3 0.0001  0.00003 2.0 0.10
  4 0.0001  0.00003 2.0 0.10
  5 0.00015 0.00003 2.0 0.10
  6 0.00015 0.00003 2.0 0.10
  7 0.0001  0.00003 2.0 0.10
  8 0.0001  0.00003 2.0 0.10
  9 0.00009 0.00003 2.0 0.10
END PERIOD
```

Available Observation Types

Example Observation Input File

```
BEGIN CONTINUOUS FILEOUT  my_model.obs.uzf.csv
  id26_infil    infiltration  26
  id126_infil   infiltration  126
  id26_dpth=20  water-content 26 20.0
  id126_dpth=51 water-content 126 1.0    #depth is below celtop
  id126_rch     uzf-gwrch     126
END CONTINUOUS

BEGIN CONTINUOUS FILEOUT my_model.uzf.budget.uzf.csv
  sinf         infiltration     uzfcells
  frommvr      from-mvr         uzfcells
  rejinf       rej-inf          uzfcells
  rejinftomvr  rej-inf-to-mvr   uzfcells
  uzet         uzet             uzfcells
  storage      storage          uzfcells
  net-inf      net-infiltration uzfcells
END CONTINUOUS

BEGIN CONTINUOUS FILEOUT  my_model.uzf.budget.gwf.csv
  gwrch      uzf-gwrch        uzfcells
  gwd        uzf-gwd          uzfcells
  gwdtomvr   uzf-gwd-to-mvr   uzfcells
```

(continues on next page)

(continued from previous page)

```
  gwet       uzf-gwet         uzfcells
END CONTINUOUS
```

1.3.26 GWF-WEL

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [AUXILIARY <auxiliary(naux)>]
  [AUXMULTNAME <auxmultname>]
  [BOUNDNAMES]
  [PRINT_INPUT]
  [PRINT_FLOWS]
  [SAVE_FLOWS]
  [AUTO_FLOW_REDUCE <auto_flow_reduce>]
  [TS6 FILEIN <ts6_filename>]
  [OBS6 FILEIN <obs6_filename>]
  [MOVER]
END OPTIONS
```

```
BEGIN DIMENSIONS
  MAXBOUND <maxbound>
END DIMENSIONS
```

FOR ANY STRESS PERIOD

```
BEGIN PERIOD <iper>
  <cellid(ncelldim)> <q> [<aux(naux)>] [<boundname>]
  <cellid(ncelldim)> <q> [<aux(naux)>] [<boundname>]
  ...
END PERIOD
```

Explanation of Variables

Block: OPTIONS

- `auxiliary` defines an array of one or more auxiliary variable names. There is no limit on the number of auxiliary variables that can be provided on this line; however, lists of information provided in subsequent blocks must have a column of data for each auxiliary variable name defined here. The number of auxiliary variables detected on this line determines the value for naux. Comments cannot be provided anywhere on this line as they will be interpreted as auxiliary variable names. Auxiliary variables may not be used by the package, but they will be available for use by other parts of the program. The program will terminate with an error if auxiliary variables are specified on more than one line in the options block.
- `auxmultname` name of auxiliary variable to be used as multiplier of well flow rate.
- `BOUNDNAMES` keyword to indicate that boundary names may be provided with the list of well cells.

- PRINT_INPUT keyword to indicate that the list of well information will be written to the listing file immediately after it is read.
- PRINT_FLOWS keyword to indicate that the list of well flow rates will be printed to the listing file for every stress period time step in which "BUDGET PRINT" is specified in Output Control. If there is no Output Control option and "PRINT_FLOWS" is specified, then flow rates are printed for the last time step of each stress period.
- SAVE_FLOWS keyword to indicate that well flow terms will be written to the file specified with "BUDGET FILEOUT" in Output Control.
- auto_flow_reduce keyword and real value that defines the fraction of the cell thickness used as an interval for smoothly adjusting negative pumping rates to 0 in cells with head values less than or equal to the bottom of the cell. Negative pumping rates are adjusted to 0 or a smaller negative value when the head in the cell is equal to or less than the calculated interval above the cell bottom. AUTO_FLOW_REDUCE is set to 0.1 if the specified value is less than or equal to zero. By default, negative pumping rates are not reduced during a simulation.
- TS6 keyword to specify that record corresponds to a time-series file.
- FILEIN keyword to specify that an input filename is expected next.
- ts6_filename defines a time-series file defining time series that can be used to assign time-varying values. See the "Time-Variable Input" section for instructions on using the time-series capability.
- OBS6 keyword to specify that record corresponds to an observations file.
- obs6_filename name of input file to define observations for the Well package. See the "Observation utility" section for instructions for preparing observation input files. Tables ref{table:gwf-obstypetable} and ref{table:gwt-obstypetable} lists observation type(s) supported by the Well package.
- MOVER keyword to indicate that this instance of the Well Package can be used with the Water Mover (MVR) Package. When the MOVER option is specified, additional memory is allocated within the package to store the available, provided, and received water.

Block: DIMENSIONS

- maxbound integer value specifying the maximum number of wells cells that will be specified for use during any stress period.

Block: PERIOD

- iper integer value specifying the starting stress period number for which the data specified in the PERIOD block apply. IPER must be less than or equal to NPER in the TDIS Package and greater than zero. The IPER value assigned to a stress period block must be greater than the IPER value assigned for the previous PERIOD block. The information specified in the PERIOD block will continue to apply for all subsequent stress periods, unless the program encounters another PERIOD block.
- cellid is the cell identifier, and depends on the type of grid that is used for the simulation. For a structured grid that uses the DIS input file, CELLID is the layer, row, and column. For a grid that uses the DISV input file, CELLID is the layer and CELL2D number. If the model uses the unstructured discretization (DISU) input file, CELLID is the node number for the cell.
- q is the volumetric well rate. A positive value indicates recharge (injection) and a negative value indicates discharge (extraction). If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- aux represents the values of the auxiliary variables for each well. The values of auxiliary variables must be present for each well. The values must be specified in the order of the auxiliary variables specified in the OPTIONS block. If the package supports time series and the Options block includes a TIMESERIESFILE entry

(see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.

- `boundname` name of the well cell. BOUNDNAME is an ASCII character variable that can contain as many as 40 characters. If BOUNDNAME contains spaces in it, then the entire name must be enclosed within single quotes.

Example Input File

```
#The OPTIONS block is optional
BEGIN OPTIONS
  AUXILIARY depth screen_length
  BOUNDNAMES
  PRINT_INPUT
  PRINT_FLOWS
  SAVE_FLOWS
END OPTIONS

#The DIMENSIONS block is required
BEGIN DIMENSIONS
  MAXBOUND 5
END DIMENSIONS

#The following block of wells will be activated for stress periods
#2 and 3.  No wells are present in stress period 1 due to an
#absence of a block for that period.
BEGIN PERIOD 2
    #layer  row  col      Q    depth  screen_length  boundname

          #wells 1 and 2
         7  102   17 -19000    275.9           17.6       CW_1
         9  192   44 -13000    280.0           24.0       CW_2

          #wells 3 through 5
         9  109   67 -24000    295.1           12.1       CW_3
        10   43   17 -12000    301.3            9.6       CW_4
        11   12   17 -17000    315.0           18.6       CW_5

END PERIOD

#Turn off all wells for stress period 4
BEGIN PERIOD 4
  #An empty block indicates that there are no wells.
END PERIOD

#For stress period 5, turn on wells 1 and 4,
#and add three wells that are grouped in a well field
BEGIN PERIOD 5
    #layer  row  col      Q    depth  screen_length  boundname
         7  102   17 -19000    275.9           17.6       CW_1
        10   43   17 -12000    301.3            9.6       CW_4

    #wells in well field
         5   27   50 -11000    190.0           20.0  well_field
         5   27   51 -10000    185.0           20.0  well_field
         5   28   50 -12000    187.3           15.0  well_field
END PERIOD
```

(continues on next page)

(continued from previous page)

```
#Use a list of wells in ASCII file wells_sp6.txt for stress period 6.
#Use these wells until the end of the simulation.
BEGIN PERIOD 6
  OPEN/CLOSE wells_sp6.txt
END PERIOD
```

Available Observation Types

Example Observation Input File

```
BEGIN OPTIONS
  DIGITS 7
  PRINT_INPUT
END OPTIONS

BEGIN CONTINUOUS FILEOUT my_model.wel.obs.csv
# obsname           obstype  ID
  wel-7-102-17      WEL      7  102  17
  wel-7-102-17      WEL      CW_1
  well-field        WEL      well_field
END CONTINUOUS
```

1.4 Groundwater Transport

1.4.1 GWT-ADV

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [SCHEME <scheme>]
END OPTIONS
```

Explanation of Variables

Block: OPTIONS

- `scheme` scheme used to solve the advection term. Can be upstream, central, or TVD. If not specified, upstream weighting is the default weighting scheme.

Example Input File

```
BEGIN OPTIONS
  SCHEME UPSTREAM
END OPTIONS
```

1.4.2 GWT-CNC

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [AUXILIARY <auxiliary(naux)>]
  [AUXMULTNAME <auxmultname>]
  [BOUNDNAMES]
  [PRINT_INPUT]
  [PRINT_FLOWS]
  [SAVE_FLOWS]
  [TS6 FILEIN <ts6_filename>]
  [OBS6 FILEIN <obs6_filename>]
END OPTIONS
```

```
BEGIN DIMENSIONS
  MAXBOUND <maxbound>
END DIMENSIONS
```

FOR ANY STRESS PERIOD

```
BEGIN PERIOD <iper>
  <cellid(ncelldim)> <conc> [<aux(naux)>] [<boundname>]
  <cellid(ncelldim)> <conc> [<aux(naux)>] [<boundname>]
  ...
END PERIOD
```

Explanation of Variables

Block: OPTIONS

- `auxiliary` defines an array of one or more auxiliary variable names. There is no limit on the number of auxiliary variables that can be provided on this line; however, lists of information provided in subsequent blocks must have a column of data for each auxiliary variable name defined here. The number of auxiliary variables detected on this line determines the value for naux. Comments cannot be provided anywhere on this line as they will be interpreted as auxiliary variable names. Auxiliary variables may not be used by the package, but they will be available for use by other parts of the program. The program will terminate with an error if auxiliary variables are specified on more than one line in the options block.
- `auxmultname` name of auxiliary variable to be used as multiplier of concentration value.
- `BOUNDNAMES` keyword to indicate that boundary names may be provided with the list of constant concentration cells.
- `PRINT_INPUT` keyword to indicate that the list of constant concentration information will be written to the listing file immediately after it is read.

- PRINT_FLOWS keyword to indicate that the list of constant concentration flow rates will be printed to the listing file for every stress period time step in which "BUDGET PRINT" is specified in Output Control. If there is no Output Control option and "PRINT_FLOWS" is specified, then flow rates are printed for the last time step of each stress period.
- SAVE_FLOWS keyword to indicate that constant concentration flow terms will be written to the file specified with "BUDGET FILEOUT" in Output Control.
- TS6 keyword to specify that record corresponds to a time-series file.
- FILEIN keyword to specify that an input filename is expected next.
- ts6_filename defines a time-series file defining time series that can be used to assign time-varying values. See the "Time-Variable Input" section for instructions on using the time-series capability.
- OBS6 keyword to specify that record corresponds to an observations file.
- obs6_filename name of input file to define observations for the Constant Concentration package. See the "Observation utility" section for instructions for preparing observation input files. Tables ref{table:gwf-obstypetable} and ref{table:gwt-obstypetable} lists observation type(s) supported by the Constant Concentration package.

Block: DIMENSIONS

- maxbound integer value specifying the maximum number of constant concentrations cells that will be specified for use during any stress period.

Block: PERIOD

- iper integer value specifying the starting stress period number for which the data specified in the PERIOD block apply. IPER must be less than or equal to NPER in the TDIS Package and greater than zero. The IPER value assigned to a stress period block must be greater than the IPER value assigned for the previous PERIOD block. The information specified in the PERIOD block will continue to apply for all subsequent stress periods, unless the program encounters another PERIOD block.
- cellid is the cell identifier, and depends on the type of grid that is used for the simulation. For a structured grid that uses the DIS input file, CELLID is the layer, row, and column. For a grid that uses the DISV input file, CELLID is the layer and CELL2D number. If the model uses the unstructured discretization (DISU) input file, CELLID is the node number for the cell.
- conc is the constant concentration value. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- aux represents the values of the auxiliary variables for each constant concentration. The values of auxiliary variables must be present for each constant concentration. The values must be specified in the order of the auxiliary variables specified in the OPTIONS block. If the package supports time series and the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- boundname name of the constant concentration cell. BOUNDNAME is an ASCII character variable that can contain as many as 40 characters. If BOUNDNAME contains spaces in it, then the entire name must be enclosed within single quotes.

Example Input File

```
BEGIN OPTIONS
  PRINT_FLOWS
  PRINT_INPUT
  SAVE_FLOWS
END OPTIONS

BEGIN DIMENSIONS
  MAXBOUND  1
END DIMENSIONS

BEGIN PERIOD 1
  1 1 1 1.0
END PERIOD
```

Available Observation Types

Example Observation Input File

```
BEGIN OPTIONS
  DIGITS 8
  PRINT_INPUT
END OPTIONS

BEGIN CONTINUOUS FILEOUT my_model.cnc01.csv
# obsname  obstype   ID
  cnc_2_1  CNC       1 1 2
  cnc_2_2  CNC       1 2 2
  cnc_2_3  CNC       1 3 2
  cnc_2_4  CNC       1 4 2
END SINGLE

BEGIN CONTINUOUS FILEOUT my_model.chd02.csv
# obsname     obstype   ID
  cnc_3_flow  CNC       CNC_1_3
END CONTINUOUS
```

1.4.3 GWT-DIS

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [LENGTH_UNITS <length_units>]
  [NOGRB]
  [XORIGIN <xorigin>]
  [YORIGIN <yorigin>]
  [ANGROT <angrot>]
END OPTIONS
```

```
BEGIN DIMENSIONS
  NLAY <nlay>
  NROW <nrow>
  NCOL <ncol>
END DIMENSIONS
```

```
BEGIN GRIDDATA
  DELR
        <delr(ncol)> -- READARRAY
  DELC
        <delc(nrow)> -- READARRAY
  TOP
        <top(ncol, nrow)> -- READARRAY
  BOTM [LAYERED]
        <botm(ncol, nrow, nlay)> -- READARRAY
  [IDOMAIN [LAYERED]
        <idomain(ncol, nrow, nlay)> -- READARRAY]
END GRIDDATA
```

Explanation of Variables

Block: OPTIONS

- `length_units` is the length units used for this model. Values can be "FEET", "METERS", or "CENTIMETERS". If not specified, the default is "UNKNOWN".
- `NOGRB` keyword to deactivate writing of the binary grid file.
- `xorigin` x-position of the lower-left corner of the model grid. A default value of zero is assigned if not specified. The value for XORIGIN does not affect the model simulation, but it is written to the binary grid file so that postprocessors can locate the grid in space.
- `yorigin` y-position of the lower-left corner of the model grid. If not specified, then a default value equal to zero is used. The value for YORIGIN does not affect the model simulation, but it is written to the binary grid file so that postprocessors can locate the grid in space.
- `angrot` counter-clockwise rotation angle (in degrees) of the lower-left corner of the model grid. If not specified, then a default value of 0.0 is assigned. The value for ANGROT does not affect the model simulation, but it is written to the binary grid file so that postprocessors can locate the grid in space.

Block: DIMENSIONS

- `nlay` is the number of layers in the model grid.
- `nrow` is the number of rows in the model grid.
- `ncol` is the number of columns in the model grid.

Block: GRIDDATA

- `delr` is the column spacing in the row direction.
- `delc` is the row spacing in the column direction.
- `top` is the top elevation for each cell in the top model layer.
- `botm` is the bottom elevation for each cell.
- `idomain` is an optional array that characterizes the existence status of a cell. If the IDOMAIN array is not specified, then all model cells exist within the solution. If the IDOMAIN value for a cell is 0, the cell does not exist in the simulation. Input and output values will be read and written for the cell, but internal to the program, the cell is excluded from the solution. If the IDOMAIN value for a cell is 1, the cell exists in the simulation. If the IDOMAIN value for a cell is -1, the cell does not exist in the simulation. Furthermore, the first existing cell above will be connected to the first existing cell below. This type of cell is referred to as a "vertical pass through" cell.

1.4.4 GWT-DISU

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [LENGTH_UNITS <length_units>]
  [NOGRB]
  [XORIGIN <xorigin>]
  [YORIGIN <yorigin>]
  [ANGROT <angrot>]
END OPTIONS
```

```
BEGIN DIMENSIONS
  NODES <nodes>
  NJA <nja>
  [NVERT <nvert>]
END DIMENSIONS
```

```
BEGIN GRIDDATA
  TOP
        <top(nodes)> -- READARRAY
  BOT
        <bot(nodes)> -- READARRAY
  AREA
        <area(nodes)> -- READARRAY
END GRIDDATA
```

```
BEGIN CONNECTIONDATA
  IAC
        <iac(nodes)> -- READARRAY
  JA
        <ja(nja)> -- READARRAY
  IHC
        <ihc(nja)> -- READARRAY
  CL12
```

(continues on next page)

(continued from previous page)

```
        <cl12(nja)> -- READARRAY
  HWVA
        <hwva(nja)> -- READARRAY
  [ANGLDEGX
        <angldegx(nja)> -- READARRAY]
END CONNECTIONDATA
```

```
BEGIN VERTICES
  <iv> <xv> <yv>
  <iv> <xv> <yv>
  ...
END VERTICES
```

```
BEGIN CELL2D
  <icell2d> <xc> <yc> <ncvert> <icvert(ncvert)>
  <icell2d> <xc> <yc> <ncvert> <icvert(ncvert)>
  ...
END CELL2D
```

Explanation of Variables

Block: OPTIONS

- `length_units` is the length units used for this model. Values can be "FEET", "METERS", or "CENTIMETERS". If not specified, the default is "UNKNOWN".
- `NOGRB` keyword to deactivate writing of the binary grid file.
- `xorigin` x-position of the origin used for model grid vertices. This value should be provided in a real-world coordinate system. A default value of zero is assigned if not specified. The value for XORIGIN does not affect the model simulation, but it is written to the binary grid file so that postprocessors can locate the grid in space.
- `yorigin` y-position of the origin used for model grid vertices. This value should be provided in a real-world coordinate system. If not specified, then a default value equal to zero is used. The value for YORIGIN does not affect the model simulation, but it is written to the binary grid file so that postprocessors can locate the grid in space.
- `angrot` counter-clockwise rotation angle (in degrees) of the model grid coordinate system relative to a real-world coordinate system. If not specified, then a default value of 0.0 is assigned. The value for ANGROT does not affect the model simulation, but it is written to the binary grid file so that postprocessors can locate the grid in space.

Block: DIMENSIONS

- `nodes` is the number of cells in the model grid.
- `nja` is the sum of the number of connections and NODES. When calculating the total number of connections, the connection between cell n and cell m is considered to be different from the connection between cell m and cell n. Thus, NJA is equal to the total number of connections, including n to m and m to n, and the total number of cells.
- `nvert` is the total number of (x, y) vertex pairs used to define the plan-view shape of each cell in the model grid. If NVERT is not specified or is specified as zero, then the VERTICES and CELL2D blocks below are not read. NVERT and the accompanying VERTICES and CELL2D blocks should be specified for most simulations. If

the XT3D or SAVE_SPECIFIC_DISCHARGE options are specified in the NPF Package, then this information is required.

Block: GRIDDATA

- `top` is the top elevation for each cell in the model grid.
- `bot` is the bottom elevation for each cell.
- `area` is the cell surface area (in plan view).

Block: CONNECTIONDATA

- `iac` is the number of connections (plus 1) for each cell. The sum of all the entries in IAC must be equal to NJA.
- `ja` is a list of cell number (n) followed by its connecting cell numbers (m) for each of the m cells connected to cell n. The number of values to provide for cell n is IAC(n). This list is sequentially provided for the first to the last cell. The first value in the list must be cell n itself, and the remaining cells must be listed in an increasing order (sorted from lowest number to highest). Note that the cell and its connections are only supplied for the GWF cells and their connections to the other GWF cells. Also note that the JA list input may be divided such that every node and its connectivity list can be on a separate line for ease in readability of the file. To further ease readability of the file, the node number of the cell whose connectivity is subsequently listed, may be expressed as a negative number, the sign of which is subsequently converted to positive by the code.
- `ihc` is an index array indicating the direction between node n and all of its m connections. If IHC = 0 then cell n and cell m are connected in the vertical direction. Cell n overlies cell m if the cell number for n is less than m; cell m overlies cell n if the cell number for m is less than n. If IHC = 1 then cell n and cell m are connected in the horizontal direction. If IHC = 2 then cell n and cell m are connected in the horizontal direction, and the connection is vertically staggered. A vertically staggered connection is one in which a cell is horizontally connected to more than one cell in a horizontal connection.
- `cl12` is the array containing connection lengths between the center of cell n and the shared face with each adjacent m cell.
- `hwva` is a symmetric array of size NJA. For horizontal connections, entries in HWVA are the horizontal width perpendicular to flow. For vertical connections, entries in HWVA are the vertical area for flow. Thus, values in the HWVA array contain dimensions of both length and area. Entries in the HWVA array have a one-to-one correspondence with the connections specified in the JA array. Likewise, there is a one-to-one correspondence between entries in the HWVA array and entries in the IHC array, which specifies the connection type (horizontal or vertical). Entries in the HWVA array must be symmetric; the program will terminate with an error if the value for HWVA for an n to m connection does not equal the value for HWVA for the corresponding n to m connection.
- `angldegx` is the angle (in degrees) between the horizontal x-axis and the outward normal to the face between a cell and its connecting cells. The angle varies between zero and 360.0 degrees, where zero degrees points in the positive x-axis direction, and 90 degrees points in the positive y-axis direction. ANGLDEGX is only needed if horizontal anisotropy is specified in the NPF Package, if the XT3D option is used in the NPF Package, or if the SAVE_SPECIFIC_DISCHARGE option is specifed in the NPF Package. ANGLDEGX does not need to be specified if these conditions are not met. ANGLDEGX is of size NJA; values specified for vertical connections and for the diagonal position are not used. Note that ANGLDEGX is read in degrees, which is different from MODFLOW-USG, which reads a similar variable (ANGLEX) in radians.

Block: VERTICES

- `iv` is the vertex number. Records in the VERTICES block must be listed in consecutive order from 1 to NVERT.
- `xv` is the x-coordinate for the vertex.
- `yv` is the y-coordinate for the vertex.

Block: CELL2D

- `icell2d` is the cell2d number. Records in the CELL2D block must be listed in consecutive order from 1 to NODES.
- `xc` is the x-coordinate for the cell center.
- `yc` is the y-coordinate for the cell center.
- `ncvert` is the number of vertices required to define the cell. There may be a different number of vertices for each cell.
- `icvert` is an array of integer values containing vertex numbers (in the VERTICES block) used to define the cell. Vertices must be listed in clockwise order.

1.4.5 GWT-DISV

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [LENGTH_UNITS <length_units>]
  [NOGRB]
  [XORIGIN <xorigin>]
  [YORIGIN <yorigin>]
  [ANGROT <angrot>]
END OPTIONS
```

```
BEGIN DIMENSIONS
  NLAY <nlay>
  NCPL <ncpl>
  NVERT <nvert>
END DIMENSIONS
```

```
BEGIN GRIDDATA
  TOP
        <top(ncpl)> -- READARRAY
  BOTM [LAYERED]
        <botm(nlay, ncpl)> -- READARRAY
  [IDOMAIN [LAYERED]
        <idomain(nlay, ncpl)> -- READARRAY]
END GRIDDATA
```

```
BEGIN VERTICES
  <iv> <xv> <yv>
  <iv> <xv> <yv>
```

(continues on next page)

(continued from previous page)

```
  ...
END VERTICES
```

```
BEGIN CELL2D
  <icell2d> <xc> <yc> <ncvert> <icvert(ncvert)>
  <icell2d> <xc> <yc> <ncvert> <icvert(ncvert)>
  ...
END CELL2D
```

Explanation of Variables

Block: OPTIONS

- `length_units` is the length units used for this model. Values can be "FEET", "METERS", or "CENTIMETERS". If not specified, the default is "UNKNOWN".
- `NOGRB` keyword to deactivate writing of the binary grid file.
- `xorigin` x-position of the origin used for model grid vertices. This value should be provided in a real-world coordinate system. A default value of zero is assigned if not specified. The value for XORIGIN does not affect the model simulation, but it is written to the binary grid file so that postprocessors can locate the grid in space.
- `yorigin` y-position of the origin used for model grid vertices. This value should be provided in a real-world coordinate system. If not specified, then a default value equal to zero is used. The value for YORIGIN does not affect the model simulation, but it is written to the binary grid file so that postprocessors can locate the grid in space.
- `angrot` counter-clockwise rotation angle (in degrees) of the model grid coordinate system relative to a real-world coordinate system. If not specified, then a default value of 0.0 is assigned. The value for ANGROT does not affect the model simulation, but it is written to the binary grid file so that postprocessors can locate the grid in space.

Block: DIMENSIONS

- `nlay` is the number of layers in the model grid.
- `ncpl` is the number of cells per layer. This is a constant value for the grid and it applies to all layers.
- `nvert` is the total number of (x, y) vertex pairs used to characterize the horizontal configuration of the model grid.

Block: GRIDDATA

- `top` is the top elevation for each cell in the top model layer.
- `botm` is the bottom elevation for each cell.
- `idomain` is an optional array that characterizes the existence status of a cell. If the IDOMAIN array is not specified, then all model cells exist within the solution. If the IDOMAIN value for a cell is 0, the cell does not exist in the simulation. Input and output values will be read and written for the cell, but internal to the program, the cell is excluded from the solution. If the IDOMAIN value for a cell is 1, the cell exists in the simulation. If the IDOMAIN value for a cell is -1, the cell does not exist in the simulation. Furthermore, the first existing cell above will be connected to the first existing cell below. This type of cell is referred to as a "vertical pass through" cell.

Block: VERTICES

- `iv` is the vertex number. Records in the VERTICES block must be listed in consecutive order from 1 to NVERT.
- `xv` is the x-coordinate for the vertex.
- `yv` is the y-coordinate for the vertex.

Block: CELL2D

- `icell2d` is the CELL2D number. Records in the CELL2D block must be listed in consecutive order from the first to the last.
- `xc` is the x-coordinate for the cell center.
- `yc` is the y-coordinate for the cell center.
- `ncvert` is the number of vertices required to define the cell. There may be a different number of vertices for each cell.
- `icvert` is an array of integer values containing vertex numbers (in the VERTICES block) used to define the cell. Vertices must be listed in clockwise order. Cells that are connected must share vertices.

1.4.6 GWT-DSP

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [XT3D_OFF]
  [XT3D_RHS]
END OPTIONS
```

```
BEGIN GRIDDATA
  [DIFFC [LAYERED]
        <diffc(nodes)> -- READARRAY]
  [ALH [LAYERED]
        <alh(nodes)> -- READARRAY]
  [ALV [LAYERED]
        <alv(nodes)> -- READARRAY]
  [ATH1 [LAYERED]
        <ath1(nodes)> -- READARRAY]
  [ATH2 [LAYERED]
        <ath2(nodes)> -- READARRAY]
  [ATV [LAYERED]
        <atv(nodes)> -- READARRAY]
END GRIDDATA
```

Explanation of Variables

Block: OPTIONS

- XT3D_OFF deactivate the xt3d method and use the faster and less accurate approximation. This option may provide a fast and accurate solution under some circumstances, such as when flow aligns with the model grid, there is no mechanical dispersion, or when the longitudinal and transverse dispersivities are equal. This option may also be used to assess the computational demand of the XT3D approach by noting the run time differences with and without this option on.
- XT3D_RHS add xt3d terms to right-hand side, when possible. This option uses less memory, but may require more iterations.

Block: GRIDDATA

- diffc effective molecular diffusion coefficient.
- alh longitudinal dispersivity in horizontal direction. If flow is strictly horizontal, then this is the longitudinal dispersivity that will be used. If flow is not strictly horizontal or strictly vertical, then the longitudinal dispersivity is a function of both ALH and ALV. If mechanical dispersion is represented (by specifying any dispersivity values) then this array is required.
- alv longitudinal dispersivity in vertical direction. If flow is strictly vertical, then this is the longitudinal dispsersivity value that will be used. If flow is not strictly horizontal or strictly vertical, then the longitudinal dispersivity is a function of both ALH and ALV. If this value is not specified and mechanical dispersion is represented, then this array is set equal to ALH.
- ath1 transverse dispersivity in horizontal direction. This is the transverse dispersivity value for the second ellipsoid axis. If flow is strictly horizontal and directed in the x direction (along a row for a regular grid), then this value controls spreading in the y direction. If mechanical dispersion is represented (by specifying any dispersivity values) then this array is required.
- ath2 transverse dispersivity in horizontal direction. This is the transverse dispersivity value for the third ellipsoid axis. If flow is strictly horizontal and directed in the x direction (along a row for a regular grid), then this value controls spreading in the z direction. If this value is not specified and mechanical dispersion is represented, then this array is set equal to ATH1.
- atv transverse dispersivity when flow is in vertical direction. If flow is strictly vertical and directed in the z direction, then this value controls spreading in the x and y directions. If this value is not specified and mechanical dispersion is represented, then this array is set equal to ATH2.

Example Input File

```
BEGIN OPTIONS
END OPTIONS

BEGIN GRIDDATA
  DIFFC
    CONSTANT  1.e-9
  ALH
    CONSTANT 1.
  ALV
    CONSTANT 1.
  ATH1
    CONSTANT 0.1
```

(continues on next page)

(continued from previous page)

```
    ATH2
      CONSTANT 0.1
    ATV
      CONSTANT 0.1
  END GRIDDATA
```

1.4.7 GWT-FMI

Structure of Blocks

FOR EACH SIMULATION

```
  BEGIN OPTIONS
    [FLOW_IMBALANCE_CORRECTION]
  END OPTIONS
```

```
  BEGIN PACKAGEDATA
    <flowtype> FILEIN <fname>
    <flowtype> FILEIN <fname>
    ...
  END PACKAGEDATA
```

Explanation of Variables

Block: OPTIONS

- `FLOW_IMBALANCE_CORRECTION` correct for an imbalance in flows by assuming that any residual flow error comes in or leaves at the concentration of the cell.

Block: PACKAGEDATA

- `flowtype` is the word GWFBUDGET, GWFHEAD, GWFMOVER or the name of an advanced GWF stress package. If GWFBUDGET is specified, then the corresponding file must be a budget file from a previous GWF Model run. If an advanced GWF stress package name appears then the corresponding file must be the budget file saved by a LAK, SFR, MAW or UZF Package.
- `FILEIN` keyword to specify that an input filename is expected next.
- `fname` is the name of the file containing flows. The path to the file should be included if the file is not located in the folder where the program was run.

Example Input File

```
BEGIN OPTIONS
  FLOW_IMBALANCE_CORRECTION
END OPTIONS

BEGIN PACKAGEDATA
  GWFBUDGET FILEIN ../flow/mygwfmodel.bud
  GWFHEAD FILEIN   ../flow/mygwfmodel.hds
  GWFMOVER FILEIN  ../flow/mygwfmodel.hds
  LAK-1 FILEIN     ../flow/mygwfmodel.lak.bud
  SFR-1 FILEIN     ../flow/mygwfmodel.sfr.bud
  MAW-1 FILEIN     ../flow/mygwfmodel.maw.bud
  UZF-1 FILEIN     ../flow/mygwfmodel.uzf.bud
  LAK-2 FILEIN     ../flow/mygwfmodel-2.lak.bud
END PACKAGEDATA
```

1.4.8 GWT-IC

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN GRIDDATA
  STRT [LAYERED]
        <strt(nodes)> -- READARRAY
END GRIDDATA
```

Explanation of Variables

Block: GRIDDATA

- `strt` is the initial (starting) concentration—that is, concentration at the beginning of the GWT Model simulation. STRT must be specified for all GWT Model simulations. One value is read for every model cell.

Example Input File

```
#The OPTIONS block is optional
BEGIN OPTIONS
END OPTIONS

#The GRIDDATA block is required
BEGIN GRIDDATA
  STRT LAYERED
    CONSTANT   0.0  Initial Concentration layer   1
    CONSTANT   0.0  Initial Concentration layer   2
END GRIDDATA
```

1.4.9 GWT-IST

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [SAVE_FLOWS]
  [SORBTION]
  [FIRST_ORDER_DECAY]
  [ZERO_ORDER_DECAY]
  [CIM FILEOUT <cimfile>]
  [CIM PRINT_FORMAT COLUMNS <columns> WIDTH <width> DIGITS <digits> <format>]
END OPTIONS
```

```
BEGIN GRIDDATA
  [CIM [LAYERED]
        <cim(nodes)> -- READARRAY]
  THETAIM [LAYERED]
        <thetaim(nodes)> -- READARRAY
  ZETAIM [LAYERED]
        <zetaim(nodes)> -- READARRAY
  [DECAY [LAYERED]
        <decay(nodes)> -- READARRAY]
  [DECAY_SORBED [LAYERED]
        <decay_sorbed(nodes)> -- READARRAY]
  BULK_DENSITY [LAYERED]
        <bulk_density(nodes)> -- READARRAY
  DISTCOEF [LAYERED]
        <distcoef(nodes)> -- READARRAY
END GRIDDATA
```

Explanation of Variables

Block: OPTIONS

- SAVE_FLOWS keyword to indicate that IST flow terms will be written to the file specified with "BUDGET FILEOUT" in Output Control.
- SORBTION is a text keyword to indicate that sorbtion will be activated. Use of this keyword requires that BULK_DENSITY and DISTCOEF are specified in the GRIDDATA block.
- FIRST_ORDER_DECAY is a text keyword to indicate that first-order decay will occur. Use of this keyword requires that DECAY and DECAY_SORBED (if sorbtion is active) are specified in the GRIDDATA block.
- ZERO_ORDER_DECAY is a text keyword to indicate that zero-order decay will occur. Use of this keyword requires that DECAY and DECAY_SORBED (if sorbtion is active) are specified in the GRIDDATA block.
- CIM keyword to specify that record corresponds to immobile concentration.
- FILEOUT keyword to specify that an output filename is expected next.
- cimfile name of the output file to write immobile concentrations.
- PRINT_FORMAT keyword to specify format for printing to the listing file.
- columns number of columns for writing data.
- width width for writing each number.

- digits number of digits to use for writing a number.
- format write format can be EXPONENTIAL, FIXED, GENERAL, or SCIENTIFIC.

Block: GRIDDATA

- cim initial concentration of the immobile domain in mass per length cubed. If CIM is not specified, then it is assumed to be zero.
- thetaim porosity of the immobile domain specified as the volume of immobile pore space per total volume (dimensionless).
- zetaim mass transfer rate coefficient between the mobile and immobile domains, in dimensions of per time.
- decay is the rate coefficient for first or zero-order decay for the aqueous phase of the immobile domain. A negative value indicates solute production. The dimensions of decay for first-order decay is one over time. The dimensions of decay for zero-order decay is mass per length cubed per time. decay will have no affect on simulation results unless either first- or zero-order decay is specified in the options block.
- decay_sorbed is the rate coefficient for first or zero-order decay for the sorbed phase of the immobile domain. A negative value indicates solute production. The dimensions of decay_sorbed for first-order decay is one over time. The dimensions of decay_sorbed for zero-order decay is mass of solute per mass of aquifer per time. If decay_sorbed is not specified and both decay and sorbtion are active, then the sorbed decay rate will be set equal to the aqueous decay rate. decay_sorbed will have no affect on simulation results unless the SORPTION keyword and either first- or zero-order decay are specified in the options block.
- bulk_density is the bulk density of the aquifer in mass per length cubed. bulk_density will have no affect on simulation results unless the SORBTION keyword is specified in the options block.
- distcoef is the distribution coefficient for the equilibrium-controlled linear sorption isotherm in dimensions of length cubed per mass. distcoef will have no affect on simulation results unless the SORBTION keyword is specified in the options block.

Example Input File

```
BEGIN OPTIONS
  SORBTION
  FIRST_ORDER_DECAY
  CIM FILEOUT gwtmodel.imd1.ucn
END OPTIONS

BEGIN GRIDDATA
  ZETAIM
    CONSTANT 0.01
  THETAIM
    CONSTANT 0.025
  BULK_DENSITY
    CONSTANT       0.25000000
  DISTCOEF
    CONSTANT       0.01000000
  DECAY
    CONSTANT       0.01000000
  DECAY_SORBED
    CONSTANT       0.01000000
END GRIDDATA
```

1.4.10 GWT-LKT

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [FLOW_PACKAGE_NAME <flow_package_name>]
  [AUXILIARY <auxiliary(naux)>]
  [FLOW_PACKAGE_AUXILIARY_NAME <flow_package_auxiliary_name>]
  [BOUNDNAMES]
  [PRINT_INPUT]
  [PRINT_CONCENTRATION]
  [PRINT_FLOWS]
  [SAVE_FLOWS]
  [CONCENTRATION FILEOUT <concfile>]
  [BUDGET FILEOUT <budgetfile>]
  [TS6 FILEIN <ts6_filename>]
  [OBS6 FILEIN <obs6_filename>]
END OPTIONS
```

```
BEGIN PACKAGEDATA
  <lakeno> <strt> [<aux(naux)>] [<boundname>]
  <lakeno> <strt> [<aux(naux)>] [<boundname>]
  ...
END PACKAGEDATA
```

FOR ANY STRESS PERIOD

```
BEGIN PERIOD <iper>
  <lakeno> <laksetting>
  <lakeno> <laksetting>
  ...
END PERIOD
```

Explanation of Variables

Block: OPTIONS

- `flow_package_name` keyword to specify the name of the corresponding flow package. If not specified, then the corresponding flow package must have the same name as this advanced transport package (the name associated with this package in the GWT name file).
- `auxiliary` defines an array of one or more auxiliary variable names. There is no limit on the number of auxiliary variables that can be provided on this line; however, lists of information provided in subsequent blocks must have a column of data for each auxiliary variable name defined here. The number of auxiliary variables detected on this line determines the value for naux. Comments cannot be provided anywhere on this line as they will be interpreted as auxiliary variable names. Auxiliary variables may not be used by the package, but they will be available for use by other parts of the program. The program will terminate with an error if auxiliary variables are specified on more than one line in the options block.
- `flow_package_auxiliary_name` keyword to specify the name of an auxiliary variable in the corresponding flow package. If specified, then the simulated concentrations from this advanced transport package will be copied into the auxiliary variable specified with this name. Note that the flow package must have an auxiliary

variable with this name or the program will terminate with an error. If the flows for this advanced transport package are read from a file, then this option will have no affect.

- BOUNDNAMES keyword to indicate that boundary names may be provided with the list of lake cells.
- PRINT_INPUT keyword to indicate that the list of lake information will be written to the listing file immediately after it is read.
- PRINT_CONCENTRATION keyword to indicate that the list of lake concentration will be printed to the listing file for every stress period in which "HEAD PRINT" is specified in Output Control. If there is no Output Control option and PRINT_CONCENTRATION is specified, then concentration are printed for the last time step of each stress period.
- PRINT_FLOWS keyword to indicate that the list of lake flow rates will be printed to the listing file for every stress period time step in which "BUDGET PRINT" is specified in Output Control. If there is no Output Control option and "PRINT_FLOWS" is specified, then flow rates are printed for the last time step of each stress period.
- SAVE_FLOWS keyword to indicate that lake flow terms will be written to the file specified with "BUDGET FILEOUT" in Output Control.
- CONCENTRATION keyword to specify that record corresponds to concentration.
- concfile name of the binary output file to write concentration information.
- BUDGET keyword to specify that record corresponds to the budget.
- FILEOUT keyword to specify that an output filename is expected next.
- budgetfile name of the binary output file to write budget information.
- TS6 keyword to specify that record corresponds to a time-series file.
- FILEIN keyword to specify that an input filename is expected next.
- ts6_filename defines a time-series file defining time series that can be used to assign time-varying values. See the "Time-Variable Input" section for instructions on using the time-series capability.
- OBS6 keyword to specify that record corresponds to an observations file.
- obs6_filename name of input file to define observations for the LKT package. See the "Observation utility" section for instructions for preparing observation input files. Tables ref{table:gwf-obstypetable} and ref{table:gwt-obstypetable} lists observation type(s) supported by the LKT package.

Block: PACKAGEDATA

- lakeno integer value that defines the lake number associated with the specified PACKAGEDATA data on the line. LAKENO must be greater than zero and less than or equal to NLAKES. Lake information must be specified for every lake or the program will terminate with an error. The program will also terminate with an error if information for a lake is specified more than once.
- strt real value that defines the starting concentration for the lake.
- aux represents the values of the auxiliary variables for each lake. The values of auxiliary variables must be present for each lake. The values must be specified in the order of the auxiliary variables specified in the OPTIONS block. If the package supports time series and the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- boundname name of the lake cell. BOUNDNAME is an ASCII character variable that can contain as many as 40 characters. If BOUNDNAME contains spaces in it, then the entire name must be enclosed within single quotes.

Block: PERIOD

- `iper` integer value specifying the starting stress period number for which the data specified in the PERIOD block apply. IPER must be less than or equal to NPER in the TDIS Package and greater than zero. The IPER value assigned to a stress period block must be greater than the IPER value assigned for the previous PERIOD block. The information specified in the PERIOD block will continue to apply for all subsequent stress periods, unless the program encounters another PERIOD block.
- `lakeno` integer value that defines the lake number associated with the specified PERIOD data on the line. LAKENO must be greater than zero and less than or equal to NLAKES.
- `laksetting` line of information that is parsed into a keyword and values. Keyword values that can be used to start the LAKSETTING string include: STATUS, CONCENTRATION, RAINFALL, EVAPORATION, RUNOFF, and AUXILIARY. These settings are used to assign the concentration of associated with the corresponding flow terms. Concentrations cannot be specified for all flow terms. For example, the Lake Package supports a "WITHDRAWAL" flow term. If this withdrawal term is active, then water will be withdrawn from the lake at the calculated concentration of the lake.

```
STATUS <status>
CONCENTRATION <concentration>
RAINFALL <rainfall>
EVAPORATION <evaporation>
RUNOFF <runoff>
EXT-INFLOW <ext-inflow>
AUXILIARY <auxname> <auxval>
```

- `status` keyword option to define lake status. STATUS can be ACTIVE, INACTIVE, or CONSTANT. By default, STATUS is ACTIVE, which means that concentration will be calculated for the lake. If a lake is inactive, then there will be no solute mass fluxes into or out of the lake and the inactive value will be written for the lake concentration. If a lake is constant, then the concentration for the lake will be fixed at the user specified value.
- `concentration` real or character value that defines the concentration for the lake. The specified CONCENTRATION is only applied if the lake is a constant concentration lake. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `rainfall` real or character value that defines the rainfall solute concentration (ML-3) for the lake. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `evaporation` real or character value that defines the concentration of evaporated water (ML-3) for the lake. If this concentration value is larger than the simulated concentration in the lake, then the evaporated water will be removed at the same concentration as the lake. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `runoff` real or character value that defines the concentration of runoff (ML-3) for the lake. Value must be greater than or equal to zero. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `ext-inflow` real or character value that defines the concentration of external inflow (ML-3) for the lake. Value must be greater than or equal to zero. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `AUXILIARY` keyword for specifying auxiliary variable.

- `auxname` name for the auxiliary variable to be assigned AUXVAL. AUXNAME must match one of the auxiliary variable names defined in the OPTIONS block. If AUXNAME does not match one of the auxiliary variable names defined in the OPTIONS block the data are ignored.
- `auxval` value for the auxiliary variable. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.

Example Input File

```
BEGIN OPTIONS
  AUXILIARY  aux1  aux2
  BOUNDNAMES
  PRINT_INPUT
  PRINT_CONCENTRATION
  PRINT_FLOWS
  SAVE_FLOWS
  CONCENTRATION  FILEOUT  gwt_lkt_02.lkt.bin
  BUDGET  FILEOUT  gwt_lkt_02.lkt.bud
  OBS6  FILEIN  gwt_lkt_02.lkt.obs
END OPTIONS

BEGIN PACKAGEDATA
# L             STRT             aux1             aux2    bname
  1       0.00000000      99.00000000     999.00000000  MYLAKE1
  2       0.00000000      99.00000000     999.00000000  MYLAKE2
  3       0.00000000      99.00000000     999.00000000  MYLAKE3
END PACKAGEDATA

BEGIN PERIOD  1
  1  STATUS  ACTIVE
  2  STATUS  ACTIVE
  3  STATUS  ACTIVE
END PERIOD  1
```

Available Observation Types

Example Observation Input File

```
BEGIN options
  DIGITS  7
  PRINT_INPUT
END options

BEGIN continuous  FILEOUT  gwt_lkt02.lkt.obs.csv
  lkt-1-conc  CONCENTRATION  1
  lkt-1-extinflow  EXT-INFLOW  1
  lkt-1-rain  RAINFALL  1
  lkt-1-roff  RUNOFF  1
  lkt-1-evap  EVAPORATION  1
  lkt-1-wdrl  WITHDRAWAL  1
  lkt-1-stor  STORAGE  1
  lkt-1-const  CONSTANT  1
  lkt-1-gwt1  LKT  1  1
```

(continues on next page)

(continued from previous page)

```
  lkt-1-gwt2  LKT  1  2
  lkt-2-gwt1  LKT  2  1
  lkt-1-mylake1  LKT  MYLAKE1
  lkt-1-fjf  FLOW-JA-FACE  1  2
  lkt-2-fjf  FLOW-JA-FACE  2  1
  lkt-3-fjf  FLOW-JA-FACE  2  3
  lkt-4-fjf  FLOW-JA-FACE  3  2
  lkt-5-fjf  FLOW-JA-FACE  MYLAKE1
  lkt-6-fjf  FLOW-JA-FACE  MYLAKE2
  lkt-7-fjf  FLOW-JA-FACE  MYLAKE3
END continuous
```

1.4.11 GWT-MST

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [SAVE_FLOWS]
  [FIRST_ORDER_DECAY]
  [ZERO_ORDER_DECAY]
  [SORBTION]
END OPTIONS
```

```
BEGIN GRIDDATA
  POROSITY [LAYERED]
        <porosity(nodes)> -- READARRAY
  [DECAY [LAYERED]
        <decay(nodes)> -- READARRAY]
  [DECAY_SORBED [LAYERED]
        <decay_sorbed(nodes)> -- READARRAY]
  [BULK_DENSITY [LAYERED]
        <bulk_density(nodes)> -- READARRAY]
  [DISTCOEF [LAYERED]
        <distcoef(nodes)> -- READARRAY]
END GRIDDATA
```

Explanation of Variables

Block: OPTIONS

- SAVE_FLOWS keyword to indicate that MST flow terms will be written to the file specified with "BUDGET FILEOUT" in Output Control.
- FIRST_ORDER_DECAY is a text keyword to indicate that first-order decay will occur. Use of this keyword requires that DECAY and DECAY_SORBED (if sorbtion is active) are specified in the GRIDDATA block.
- ZERO_ORDER_DECAY is a text keyword to indicate that zero-order decay will occur. Use of this keyword requires that DECAY and DECAY_SORBED (if sorbtion is active) are specified in the GRIDDATA block.
- SORBTION is a text keyword to indicate that sorbtion will be activated. Use of this keyword requires that BULK_DENSITY and DISTCOEF are specified in the GRIDDATA block.

Block: GRIDDATA

- `porosity` is the aquifer porosity.
- `decay` is the rate coefficient for first or zero-order decay for the aqueous phase of the mobile domain. A negative value indicates solute production. The dimensions of decay for first-order decay is one over time. The dimensions of decay for zero-order decay is mass per length cubed per time. decay will have no affect on simulation results unless either first- or zero-order decay is specified in the options block.
- `decay_sorbed` is the rate coefficient for first or zero-order decay for the sorbed phase of the mobile domain. A negative value indicates solute production. The dimensions of decay_sorbed for first-order decay is one over time. The dimensions of decay_sorbed for zero-order decay is mass of solute per mass of aquifer per time. If decay_sorbed is not specified and both decay and sorbtion are active, then the sorbed decay rate will be set equal to the aqueous decay rate. decay_sorbed will have no affect on simulation results unless the SORPTION keyword and either first- or zero-order decay are specified in the options block.
- `bulk_density` is the bulk density of the aquifer in mass per length cubed. bulk_density is not required unless the SORBTION keyword is specified.
- `distcoef` is the distribution coefficient for the equilibrium-controlled linear sorption isotherm in dimensions of length cubed per mass. distcoef is not required unless the SORBTION keyword is specified.

Example Input File

```
BEGIN OPTIONS
  SORBTION
  FIRST_ORDER_DECAY
END OPTIONS

BEGIN GRIDDATA
  POROSITY
    CONSTANT  0.1
  DECAY
    CONSTANT 0.001
  DECAY_SORBED
    CONSTANT 0.001
  BULK_DENSITY
    CONSTANT 1.
  DISTCOEF
    CONSTANT 0.01
END GRIDDATA
```

1.4.12 GWT-MVT

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [PRINT_INPUT]
  [PRINT_FLOWS]
  [SAVE_FLOWS]
  [BUDGET FILEOUT <budgetfile>]
END OPTIONS
```

Explanation of Variables

Block: OPTIONS

- PRINT_INPUT keyword to indicate that the list of mover information will be written to the listing file immediately after it is read.
- PRINT_FLOWS keyword to indicate that the list of lake flow rates will be printed to the listing file for every stress period time step in which "BUDGET PRINT" is specified in Output Control. If there is no Output Control option and "PRINT_FLOWS" is specified, then flow rates are printed for the last time step of each stress period.
- SAVE_FLOWS keyword to indicate that lake flow terms will be written to the file specified with "BUDGET FILEOUT" in Output Control.
- BUDGET keyword to specify that record corresponds to the budget.
- FILEOUT keyword to specify that an output filename is expected next.
- budgetfile name of the binary output file to write budget information.

Example Input File

```
BEGIN OPTIONS
  PRINT_INPUT
  PRINT_FLOWS
  SAVE_FLOWS
  BUDGET FILEOUT mygwtmodel.mvt.bud
END OPTIONS
```

1.4.13 GWT-MWT

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [FLOW_PACKAGE_NAME <flow_package_name>]
  [AUXILIARY <auxiliary(naux)>]
  [FLOW_PACKAGE_AUXILIARY_NAME <flow_package_auxiliary_name>]
  [BOUNDNAMES]
  [PRINT_INPUT]
  [PRINT_CONCENTRATION]
  [PRINT_FLOWS]
  [SAVE_FLOWS]
  [CONCENTRATION FILEOUT <concfile>]
  [BUDGET FILEOUT <budgetfile>]
  [TS6 FILEIN <ts6_filename>]
  [OBS6 FILEIN <obs6_filename>]
END OPTIONS
```

```
BEGIN PACKAGEDATA
  <mawno> <strt> [<aux(naux)>] [<boundname>]
  <mawno> <strt> [<aux(naux)>] [<boundname>]
  ...
END PACKAGEDATA
```

FOR ANY STRESS PERIOD

```
BEGIN PERIOD <iper>
  <mawno> <mwtsetting>
  <mawno> <mwtsetting>
  ...
END PERIOD
```

Explanation of Variables

Block: OPTIONS

- `flow_package_name` keyword to specify the name of the corresponding flow package. If not specified, then the corresponding flow package must have the same name as this advanced transport package (the name associated with this package in the GWT name file).
- `auxiliary` defines an array of one or more auxiliary variable names. There is no limit on the number of auxiliary variables that can be provided on this line; however, lists of information provided in subsequent blocks must have a column of data for each auxiliary variable name defined here. The number of auxiliary variables detected on this line determines the value for naux. Comments cannot be provided anywhere on this line as they will be interpreted as auxiliary variable names. Auxiliary variables may not be used by the package, but they will be available for use by other parts of the program. The program will terminate with an error if auxiliary variables are specified on more than one line in the options block.
- `flow_package_auxiliary_name` keyword to specify the name of an auxiliary variable in the corresponding flow package. If specified, then the simulated concentrations from this advanced transport package will be copied into the auxiliary variable specified with this name. Note that the flow package must have an auxiliary variable with this name or the program will terminate with an error. If the flows for this advanced transport package are read from a file, then this option will have no affect.
- `BOUNDNAMES` keyword to indicate that boundary names may be provided with the list of well cells.
- `PRINT_INPUT` keyword to indicate that the list of well information will be written to the listing file immediately after it is read.
- `PRINT_CONCENTRATION` keyword to indicate that the list of well concentration will be printed to the listing file for every stress period in which "HEAD PRINT" is specified in Output Control. If there is no Output Control option and PRINT_CONCENTRATION is specified, then concentration are printed for the last time step of each stress period.
- `PRINT_FLOWS` keyword to indicate that the list of well flow rates will be printed to the listing file for every stress period time step in which "BUDGET PRINT" is specified in Output Control. If there is no Output Control option and "PRINT_FLOWS" is specified, then flow rates are printed for the last time step of each stress period.
- `SAVE_FLOWS` keyword to indicate that well flow terms will be written to the file specified with "BUDGET FILEOUT" in Output Control.
- `CONCENTRATION` keyword to specify that record corresponds to concentration.
- `concfile` name of the binary output file to write concentration information.
- `BUDGET` keyword to specify that record corresponds to the budget.
- `FILEOUT` keyword to specify that an output filename is expected next.
- `budgetfile` name of the binary output file to write budget information.
- `TS6` keyword to specify that record corresponds to a time-series file.
- `FILEIN` keyword to specify that an input filename is expected next.

- ts6_filename defines a time-series file defining time series that can be used to assign time-varying values. See the "Time-Variable Input" section for instructions on using the time-series capability.
- OBS6 keyword to specify that record corresponds to an observations file.
- obs6_filename name of input file to define observations for the MWT package. See the "Observation utility" section for instructions for preparing observation input files. Tables ref{table:gwf-obstypetable} and ref{table:gwt-obstypetable} lists observation type(s) supported by the MWT package.

Block: PACKAGEDATA

- mawno integer value that defines the well number associated with the specified PACKAGEDATA data on the line. MAWNO must be greater than zero and less than or equal to NMAWWELLS. Well information must be specified for every well or the program will terminate with an error. The program will also terminate with an error if information for a well is specified more than once.
- strt real value that defines the starting concentration for the well.
- aux represents the values of the auxiliary variables for each well. The values of auxiliary variables must be present for each well. The values must be specified in the order of the auxiliary variables specified in the OPTIONS block. If the package supports time series and the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- boundname name of the well cell. BOUNDNAME is an ASCII character variable that can contain as many as 40 characters. If BOUNDNAME contains spaces in it, then the entire name must be enclosed within single quotes.

Block: PERIOD

- iper integer value specifying the starting stress period number for which the data specified in the PERIOD block apply. IPER must be less than or equal to NPER in the TDIS Package and greater than zero. The IPER value assigned to a stress period block must be greater than the IPER value assigned for the previous PERIOD block. The information specified in the PERIOD block will continue to apply for all subsequent stress periods, unless the program encounters another PERIOD block.
- mawno integer value that defines the well number associated with the specified PERIOD data on the line. MAWNO must be greater than zero and less than or equal to NMAWWELLS.
- mwtsetting line of information that is parsed into a keyword and values. Keyword values that can be used to start the MWTSETTING string include: STATUS, CONCENTRATION, RAINFALL, EVAPORATION, RUNOFF, and AUXILIARY. These settings are used to assign the concentration of associated with the corresponding flow terms. Concentrations cannot be specified for all flow terms. For example, the Multi-Aquifer Well Package supports a "WITHDRAWAL" flow term. If this withdrawal term is active, then water will be withdrawn from the well at the calculated concentration of the well.

```
STATUS <status>
CONCENTRATION <concentration>
RATE <rate>
AUXILIARY <auxname> <auxval>
```

- status keyword option to define well status. STATUS can be ACTIVE, INACTIVE, or CONSTANT. By default, STATUS is ACTIVE, which means that concentration will be calculated for the well. If a well is inactive, then there will be no solute mass fluxes into or out of the well and the inactive value will be written for the well concentration. If a well is constant, then the concentration for the well will be fixed at the user specified value.

- `concentration` real or character value that defines the concentration for the well. The specified CONCENTRATION is only applied if the well is a constant concentration well. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `rate` real or character value that defines the injection solute concentration (ML-3) for the well. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `AUXILIARY` keyword for specifying auxiliary variable.
- `auxname` name for the auxiliary variable to be assigned AUXVAL. AUXNAME must match one of the auxiliary variable names defined in the OPTIONS block. If AUXNAME does not match one of the auxiliary variable names defined in the OPTIONS block the data are ignored.
- `auxval` value for the auxiliary variable. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.

Example Input File

```
BEGIN OPTIONS
  AUXILIARY  aux1  aux2
  BOUNDNAMES
  PRINT_INPUT
  PRINT_CONCENTRATION
  PRINT_FLOWS
  SAVE_FLOWS
  CONCENTRATION  FILEOUT  gwt_mwt_02.mwt.bin
  BUDGET  FILEOUT  gwt_mwt_02.mwt.bud
  OBS6  FILEIN  gwt_mwt_02.mwt.obs
END OPTIONS

BEGIN PACKAGEDATA
# L             STRT             aux1             aux2    bname
  1       0.00000000      99.00000000     999.00000000  MYWELL1
  2       0.00000000      99.00000000     999.00000000  MYWELL2
  3       0.00000000      99.00000000     999.00000000  MYWELL3
END PACKAGEDATA

BEGIN PERIOD  1
  1  STATUS  ACTIVE
  2  STATUS  ACTIVE
  3  STATUS  ACTIVE
END PERIOD  1
```

Available Observation Types

Example Observation Input File

```
BEGIN options
  DIGITS  12
  PRINT_INPUT
END options
```

(continues on next page)

(continued from previous page)

```
  BEGIN continuous  FILEOUT  gwt_mwt_02.mwt.obs.csv
    mwt1mwt  MWT  1  1
    mwt2mwt  MWT  2  1
    mwt3mwt  MWT  3  1
    mwt4mwt  MWT  4  1
    mwt1conc  CONCENTRATION  1
    mwt2conc  CONCENTRATION  2
    mwt3conc  CONCENTRATION  3
    mwt4conc  CONCENTRATION  4
    mwt1stor  STORAGE  1
    mwt2stor  STORAGE  2
    mwt3stor  STORAGE  3
    mwt4stor  STORAGE  4
    mwt1cnst  CONSTANT  1
    mwt2cnst  CONSTANT  2
    mwt3cnst  CONSTANT  3
    mwt4cnst  CONSTANT  4
    mwt1fmvr  FROM-MVR  1
    mwt2fmvr  FROM-MVR  2
    mwt3fmvr  FROM-MVR  3
    mwt4fmvr  FROM-MVR  4
    mwt1rate  RATE  1
    mwt2rate  RATE  2
    mwt3rate  RATE  3
    mwt4rate  RATE  4
    mwt1rtmv  RATE-TO-MVR  1
    mwt2rtmv  RATE-TO-MVR  2
    mwt3rtmv  RATE-TO-MVR  3
    mwt4rtmv  RATE-TO-MVR  4
    mwt1fwrt  FW-RATE  1
    mwt2fwrt  FW-RATE  2
    mwt3fwrt  FW-RATE  3
    mwt4fwrt  FW-RATE  4
    mwt1frtm  FW-RATE-TO-MVR  1
    mwt2frtm  FW-RATE-TO-MVR  2
    mwt3frtm  FW-RATE-TO-MVR  3
    mwt4frtm  FW-RATE-TO-MVR  4
  END continuous  FILEOUT  gwt_mwt_02.mwt.obs.csv
```

1.4.14 GWT-NAM

Structure of Blocks

FOR EACH SIMULATION

```
  BEGIN OPTIONS
    [LIST <list>]
    [PRINT_INPUT]
    [PRINT_FLOWS]
    [SAVE_FLOWS]
  END OPTIONS
```

```
  BEGIN PACKAGES
    <ftype> <fname> [<pname>]
```

(continues on next page)

(continued from previous page)

```
  <ftype> <fname> [<pname>]
  ...
END PACKAGES
```

Explanation of Variables

Block: OPTIONS

- `list` is name of the listing file to create for this GWT model. If not specified, then the name of the list file will be the basename of the GWT model name file and the '.lst' extension. For example, if the GWT name file is called "my.model.nam" then the list file will be called "my.model.lst".
- `PRINT_INPUT` keyword to indicate that the list of all model stress package information will be written to the listing file immediately after it is read.
- `PRINT_FLOWS` keyword to indicate that the list of all model package flow rates will be printed to the listing file for every stress period time step in which "BUDGET PRINT" is specified in Output Control. If there is no Output Control option and "PRINT_FLOWS" is specified, then flow rates are printed for the last time step of each stress period.
- `SAVE_FLOWS` keyword to indicate that all model package flow terms will be written to the file specified with "BUDGET FILEOUT" in Output Control.

Block: PACKAGES

- `ftype` is the file type, which must be one of the following character values shown in table ref{table:ftype}. Ftype may be entered in any combination of uppercase and lowercase.
- `fname` is the name of the file containing the package input. The path to the file should be included if the file is not located in the folder where the program was run.
- `pname` is the user-defined name for the package. PNAME is restricted to 16 characters. No spaces are allowed in PNAME. PNAME character values are read and stored by the program for stress packages only. These names may be useful for labeling purposes when multiple stress packages of the same type are located within a single GWT Model. If PNAME is specified for a stress package, then PNAME will be used in the flow budget table in the listing file; it will also be used for the text entry in the cell-by-cell budget file. PNAME is case insensitive and is stored in all upper case letters.

Example Input File

```
# This block is optional
BEGIN OPTIONS
END OPTIONS

BEGIN PACKAGES
  DIS6       transport.dis
  IC6        transport.ic
  MST6       transport.mst
  ADV6       transport.adv
  DSP6       transport.dsp
  SSM6       transport.ssm
  CNC6       transport01.cnc LEFT
```

(continues on next page)

(continued from previous page)

```
      CNC6       transport02.cnc RIGHT
      SRC6       transport01.src LAY1
      SRC6       transport02.src LAY2
      SRC6       transport03.src LAY3
      IST6       transport01.ist CLAY
      IST6       transport02.ist SILT
      OC6        transport.oc
    END PACKAGES
```

1.4.15 GWT-OC

Structure of Blocks

FOR EACH SIMULATION

```
    BEGIN OPTIONS
      [BUDGET FILEOUT <budgetfile>]
      [CONCENTRATION FILEOUT <concentrationfile>]
      [CONCENTRATION PRINT_FORMAT COLUMNS <columns> WIDTH <width> DIGITS <digits>
↪<format>]
    END OPTIONS
```

FOR ANY STRESS PERIOD

```
    BEGIN PERIOD <iper>
      [SAVE <rtype> <ocsetting>]
      [PRINT <rtype> <ocsetting>]
    END PERIOD
```

Explanation of Variables

Block: OPTIONS

- `BUDGET` keyword to specify that record corresponds to the budget.
- `FILEOUT` keyword to specify that an output filename is expected next.
- `budgetfile` name of the output file to write budget information.
- `CONCENTRATION` keyword to specify that record corresponds to concentration.
- `concentrationfile` name of the output file to write conc information.
- `PRINT_FORMAT` keyword to specify format for printing to the listing file.
- `columns` number of columns for writing data.
- `width` width for writing each number.
- `digits` number of digits to use for writing a number.
- `format` write format can be EXPONENTIAL, FIXED, GENERAL, or SCIENTIFIC.

Block: PERIOD

- `iper` integer value specifying the starting stress period number for which the data specified in the PERIOD block apply. IPER must be less than or equal to NPER in the TDIS Package and greater than zero. The IPER value assigned to a stress period block must be greater than the IPER value assigned for the previous PERIOD block. The information specified in the PERIOD block will continue to apply for all subsequent stress periods, unless the program encounters another PERIOD block.
- `SAVE` keyword to indicate that information will be saved this stress period.
- `PRINT` keyword to indicate that information will be printed this stress period.
- `rtype` type of information to save or print. Can be BUDGET or CONCENTRATION.
- `ocsetting` specifies the steps for which the data will be saved.

```
ALL
FIRST
LAST
FREQUENCY <frequency>
STEPS <steps(<nstp)>
```

- `ALL` keyword to indicate save for all time steps in period.
- `FIRST` keyword to indicate save for first step in period. This keyword may be used in conjunction with other keywords to print or save results for multiple time steps.
- `LAST` keyword to indicate save for last step in period. This keyword may be used in conjunction with other keywords to print or save results for multiple time steps.
- `frequency` save at the specified time step frequency. This keyword may be used in conjunction with other keywords to print or save results for multiple time steps.
- `steps` save for each step specified in STEPS. This keyword may be used in conjunction with other keywords to print or save results for multiple time steps.

Example Input File

```
BEGIN OPTIONS
  CONCENTRATION FILEOUT  transport.ucn
  CONCENTRATION PRINT_FORMAT  COLUMNS 15  WIDTH 7  DIGITS 2  FIXED
END OPTIONS

BEGIN PERIOD 1
  PRINT BUDGET ALL
  SAVE CONCENTRATION ALL
  PRINT CONCENTRATION ALL
END PERIOD
```

1.4.16 GWT-SFT

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [FLOW_PACKAGE_NAME <flow_package_name>]
  [AUXILIARY <auxiliary(naux)>]
  [FLOW_PACKAGE_AUXILIARY_NAME <flow_package_auxiliary_name>]
  [BOUNDNAMES]
  [PRINT_INPUT]
  [PRINT_CONCENTRATION]
  [PRINT_FLOWS]
  [SAVE_FLOWS]
  [CONCENTRATION FILEOUT <concfile>]
  [BUDGET FILEOUT <budgetfile>]
  [TS6 FILEIN <ts6_filename>]
  [OBS6 FILEIN <obs6_filename>]
END OPTIONS
```

```
BEGIN PACKAGEDATA
  <rno> <strt> [<aux(naux)>] [<boundname>]
  <rno> <strt> [<aux(naux)>] [<boundname>]
  ...
END PACKAGEDATA
```

FOR ANY STRESS PERIOD

```
BEGIN PERIOD <iper>
  <rno> <reachsetting>
  <rno> <reachsetting>
  ...
END PERIOD
```

Explanation of Variables

Block: OPTIONS

- `flow_package_name` keyword to specify the name of the corresponding flow package. If not specified, then the corresponding flow package must have the same name as this advanced transport package (the name associated with this package in the GWT name file).
- `auxiliary` defines an array of one or more auxiliary variable names. There is no limit on the number of auxiliary variables that can be provided on this line; however, lists of information provided in subsequent blocks must have a column of data for each auxiliary variable name defined here. The number of auxiliary variables detected on this line determines the value for naux. Comments cannot be provided anywhere on this line as they will be interpreted as auxiliary variable names. Auxiliary variables may not be used by the package, but they will be available for use by other parts of the program. The program will terminate with an error if auxiliary variables are specified on more than one line in the options block.
- `flow_package_auxiliary_name` keyword to specify the name of an auxiliary variable in the corresponding flow package. If specified, then the simulated concentrations from this advanced transport package will be copied into the auxiliary variable specified with this name. Note that the flow package must have an auxiliary

variable with this name or the program will terminate with an error. If the flows for this advanced transport package are read from a file, then this option will have no affect.

- BOUNDNAMES keyword to indicate that boundary names may be provided with the list of reach cells.
- PRINT_INPUT keyword to indicate that the list of reach information will be written to the listing file immediately after it is read.
- PRINT_CONCENTRATION keyword to indicate that the list of reach stages will be printed to the listing file for every stress period in which "HEAD PRINT" is specified in Output Control. If there is no Output Control option and PRINT_STAGE is specified, then stages are printed for the last time step of each stress period.
- PRINT_FLOWS keyword to indicate that the list of reach flow rates will be printed to the listing file for every stress period time step in which "BUDGET PRINT" is specified in Output Control. If there is no Output Control option and "PRINT_FLOWS" is specified, then flow rates are printed for the last time step of each stress period.
- SAVE_FLOWS keyword to indicate that reach flow terms will be written to the file specified with "BUDGET FILEOUT" in Output Control.
- CONCENTRATION keyword to specify that record corresponds to concentration.
- concfile name of the binary output file to write concentration information.
- BUDGET keyword to specify that record corresponds to the budget.
- FILEOUT keyword to specify that an output filename is expected next.
- budgetfile name of the binary output file to write budget information.
- TS6 keyword to specify that record corresponds to a time-series file.
- FILEIN keyword to specify that an input filename is expected next.
- ts6_filename defines a time-series file defining time series that can be used to assign time-varying values. See the "Time-Variable Input" section for instructions on using the time-series capability.
- OBS6 keyword to specify that record corresponds to an observations file.
- obs6_filename name of input file to define observations for the SFT package. See the "Observation utility" section for instructions for preparing observation input files. Tables ref{table:gwf-obstypetable} and ref{table:gwt-obstypetable} lists observation type(s) supported by the SFT package.

Block: PACKAGEDATA

- rno integer value that defines the reach number associated with the specified PACKAGEDATA data on the line. RNO must be greater than zero and less than or equal to NREACHES. Reach information must be specified for every reach or the program will terminate with an error. The program will also terminate with an error if information for a reach is specified more than once.
- strt real value that defines the starting concentration for the reach.
- aux represents the values of the auxiliary variables for each reach. The values of auxiliary variables must be present for each reach. The values must be specified in the order of the auxiliary variables specified in the OPTIONS block. If the package supports time series and the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- boundname name of the reach cell. BOUNDNAME is an ASCII character variable that can contain as many as 40 characters. If BOUNDNAME contains spaces in it, then the entire name must be enclosed within single quotes.

Block: PERIOD

- iper integer value specifying the starting stress period number for which the data specified in the PERIOD block apply. IPER must be less than or equal to NPER in the TDIS Package and greater than zero. The IPER value assigned to a stress period block must be greater than the IPER value assigned for the previous PERIOD block. The information specified in the PERIOD block will continue to apply for all subsequent stress periods, unless the program encounters another PERIOD block.
- rno integer value that defines the reach number associated with the specified PERIOD data on the line. RNO must be greater than zero and less than or equal to NREACHES.
- reachsetting line of information that is parsed into a keyword and values. Keyword values that can be used to start the REACHSETTING string include: STATUS, CONCENTRATION, RAINFALL, EVAPORATION, RUNOFF, and AUXILIARY. These settings are used to assign the concentration of associated with the corresponding flow terms. Concentrations cannot be specified for all flow terms. For example, the Streamflow Package supports a "DIVERSION" flow term. Diversion water will be routed using the calculated concentration of the reach.

```
STATUS <status>
CONCENTRATION <concentration>
RAINFALL <rainfall>
EVAPORATION <evaporation>
RUNOFF <runoff>
INFLOW <inflow>
AUXILIARY <auxname> <auxval>
```

- status keyword option to define reach status. STATUS can be ACTIVE, INACTIVE, or CONSTANT. By default, STATUS is ACTIVE, which means that concentration will be calculated for the reach. If a reach is inactive, then there will be no solute mass fluxes into or out of the reach and the inactive value will be written for the reach concentration. If a reach is constant, then the concentration for the reach will be fixed at the user specified value.
- concentration real or character value that defines the concentration for the reach. The specified CONCENTRATION is only applied if the reach is a constant concentration reach. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- rainfall real or character value that defines the rainfall solute concentration (ML-3) for the reach. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- evaporation real or character value that defines the concentration of evaporated water (ML-3) for the reach. If this concentration value is larger than the simulated concentration in the reach, then the evaporated water will be removed at the same concentration as the reach. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- runoff real or character value that defines the concentration of runoff (ML-3) for the reach. Value must be greater than or equal to zero. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- inflow real or character value that defines the concentration of inflow (ML-3) for the reach. Value must be greater than or equal to zero. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- AUXILIARY keyword for specifying auxiliary variable.

- `auxname` name for the auxiliary variable to be assigned AUXVAL. AUXNAME must match one of the auxiliary variable names defined in the OPTIONS block. If AUXNAME does not match one of the auxiliary variable names defined in the OPTIONS block the data are ignored.
- `auxval` value for the auxiliary variable. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.

Example Input File

```
BEGIN OPTIONS
  AUXILIARY  aux1  aux2
  BOUNDNAMES
  PRINT_INPUT
  PRINT_CONCENTRATION
  PRINT_FLOWS
  SAVE_FLOWS
  CONCENTRATION  FILEOUT  gwt_sft_02.sft.bin
  BUDGET  FILEOUT  gwt_sft_02.sft.bud
  OBS6  FILEIN  gwt_sft_02.sft.obs
END OPTIONS

BEGIN PACKAGEDATA
# L             STRT             aux1             aux2    bname
  1       0.00000000      99.00000000     999.00000000   REACH1
  2       0.00000000      99.00000000     999.00000000   REACH2
  3       0.00000000      99.00000000     999.00000000   REACH3
END PACKAGEDATA

BEGIN PERIOD  1
  1  STATUS  ACTIVE
  2  STATUS  ACTIVE
  3  STATUS  ACTIVE
END PERIOD  1
```

Available Observation Types

Example Observation Input File

```
BEGIN options
  DIGITS  7
  PRINT_INPUT
END options

BEGIN continuous  FILEOUT  gwt_lkt02.lkt.obs.csv
  sft-1-conc  CONCENTRATION  1
  sft-1-extinflow  EXT-INFLOW  1
  sft-1-rain  RAINFALL  1
  sft-1-roff  RUNOFF  1
  sft-1-evap  EVAPORATION  1
  sft-1-stor  STORAGE  1
  sft-1-const  CONSTANT  1
  sft-1-gwt1  SFT  1  1
  sft-1-gwt2  SFT  1  2
```

(continues on next page)

(continued from previous page)

```
  sft-2-gwt1  SFT  2  1
  sft-1-mylake1  SFT  MYREACHES
  sft-1-fjf  FLOW-JA-FACE  1  2
  sft-2-fjf  FLOW-JA-FACE  2  1
  sft-3-fjf  FLOW-JA-FACE  2  3
  sft-4-fjf  FLOW-JA-FACE  3  2
  sft-5-fjf  FLOW-JA-FACE  MYREACH1
  sft-6-fjf  FLOW-JA-FACE  MYREACH2
  sft-7-fjf  FLOW-JA-FACE  MYREACH3
END continuous
```

1.4.17 GWT-SRC

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [AUXILIARY <auxiliary(naux)>]
  [AUXMULTNAME <auxmultname>]
  [BOUNDNAMES]
  [PRINT_INPUT]
  [PRINT_FLOWS]
  [SAVE_FLOWS]
  [TS6 FILEIN <ts6_filename>]
  [OBS6 FILEIN <obs6_filename>]
END OPTIONS
```

```
BEGIN DIMENSIONS
  MAXBOUND <maxbound>
END DIMENSIONS
```

FOR ANY STRESS PERIOD

```
BEGIN PERIOD <iper>
  <cellid(ncelldim)> <smassrate> [<aux(naux)>] [<boundname>]
  <cellid(ncelldim)> <smassrate> [<aux(naux)>] [<boundname>]
  ...
END PERIOD
```

Explanation of Variables

Block: OPTIONS

- `auxiliary` defines an array of one or more auxiliary variable names. There is no limit on the number of auxiliary variables that can be provided on this line; however, lists of information provided in subsequent blocks must have a column of data for each auxiliary variable name defined here. The number of auxiliary variables detected on this line determines the value for naux. Comments cannot be provided anywhere on this line as they will be interpreted as auxiliary variable names. Auxiliary variables may not be used by the package, but they will be available for use by other parts of the program. The program will terminate with an error if auxiliary variables are specified on more than one line in the options block.
- `auxmultname` name of auxiliary variable to be used as multiplier of mass loading rate.

- BOUNDNAMES keyword to indicate that boundary names may be provided with the list of mass source cells.
- PRINT_INPUT keyword to indicate that the list of mass source information will be written to the listing file immediately after it is read.
- PRINT_FLOWS keyword to indicate that the list of mass source flow rates will be printed to the listing file for every stress period time step in which "BUDGET PRINT" is specified in Output Control. If there is no Output Control option and "PRINT_FLOWS" is specified, then flow rates are printed for the last time step of each stress period.
- SAVE_FLOWS keyword to indicate that mass source flow terms will be written to the file specified with "BUDGET FILEOUT" in Output Control.
- TS6 keyword to specify that record corresponds to a time-series file.
- FILEIN keyword to specify that an input filename is expected next.
- ts6_filename defines a time-series file defining time series that can be used to assign time-varying values. See the "Time-Variable Input" section for instructions on using the time-series capability.
- OBS6 keyword to specify that record corresponds to an observations file.
- obs6_filename name of input file to define observations for the Mass Source package. See the "Observation utility" section for instructions for preparing observation input files. Tables ref{table:gwf-obstypetable} and ref{table:gwt-obstypetable} lists observation type(s) supported by the Mass Source package.

Block: DIMENSIONS

- maxbound integer value specifying the maximum number of sources cells that will be specified for use during any stress period.

Block: PERIOD

- iper integer value specifying the starting stress period number for which the data specified in the PERIOD block apply. IPER must be less than or equal to NPER in the TDIS Package and greater than zero. The IPER value assigned to a stress period block must be greater than the IPER value assigned for the previous PERIOD block. The information specified in the PERIOD block will continue to apply for all subsequent stress periods, unless the program encounters another PERIOD block.
- cellid is the cell identifier, and depends on the type of grid that is used for the simulation. For a structured grid that uses the DIS input file, CELLID is the layer, row, and column. For a grid that uses the DISV input file, CELLID is the layer and CELL2D number. If the model uses the unstructured discretization (DISU) input file, CELLID is the node number for the cell.
- smassrate is the mass source loading rate. A positive value indicates addition of solute mass and a negative value indicates removal of solute mass. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- aux represents the values of the auxiliary variables for each mass source. The values of auxiliary variables must be present for each mass source. The values must be specified in the order of the auxiliary variables specified in the OPTIONS block. If the package supports time series and the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- boundname name of the mass source cell. BOUNDNAME is an ASCII character variable that can contain as many as 40 characters. If BOUNDNAME contains spaces in it, then the entire name must be enclosed within single quotes.

Example Input File

```
BEGIN OPTIONS
  PRINT_FLOWS
  PRINT_INPUT
  SAVE_FLOWS
END OPTIONS

BEGIN DIMENSIONS
  MAXBOUND  1
END DIMENSIONS

BEGIN PERIOD 1
  1 1 1 1.0
END PERIOD
```

Available Observation Types

Example Observation Input File

```
BEGIN OPTIONS
  DIGITS 7
  PRINT_INPUT
END OPTIONS

BEGIN CONTINUOUS FILEOUT my_model.src.obs.csv
# obsname           obstype  ID
  src-7-102-17      SRC      7  102  17
  src-7-102-17      SRC      CW_1
  sources           SRC      sources
END CONTINUOUS
```

1.4.18 GWT-SSM

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [PRINT_FLOWS]
  [SAVE_FLOWS]
END OPTIONS
```

```
BEGIN SOURCES
  <pname> <srctype> <auxname>
  <pname> <srctype> <auxname>
  ...
END SOURCES
```

Explanation of Variables

Block: OPTIONS

- PRINT_FLOWS keyword to indicate that the list of SSM flow rates will be printed to the listing file for every stress period time step in which "BUDGET PRINT" is specified in Output Control. If there is no Output Control option and "PRINT_FLOWS" is specified, then flow rates are printed for the last time step of each stress period.
- SAVE_FLOWS keyword to indicate that SSM flow terms will be written to the file specified with "BUDGET FILEOUT" in Output Control.

Block: SOURCES

- pname name of the package for which an auxiliary variable contains a source concentration.
- srctype type of the source. Must be AUX.
- auxname name of the auxiliary variable in the package PNAME that contains the source concentration.

Example Input File

```
BEGIN SOURCES
# pname srctype        auxname
  WEL-1     AUX  CONCENTRATION
  LAK-1     AUX  CONCENTRATION
END SOURCES
```

1.4.19 GWT-UZT

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [FLOW_PACKAGE_NAME <flow_package_name>]
  [AUXILIARY <auxiliary(naux)>]
  [FLOW_PACKAGE_AUXILIARY_NAME <flow_package_auxiliary_name>]
  [BOUNDNAMES]
  [PRINT_INPUT]
  [PRINT_CONCENTRATION]
  [PRINT_FLOWS]
  [SAVE_FLOWS]
  [CONCENTRATION FILEOUT <concfile>]
  [BUDGET FILEOUT <budgetfile>]
  [TS6 FILEIN <ts6_filename>]
  [OBS6 FILEIN <obs6_filename>]
END OPTIONS
```

```
BEGIN PACKAGEDATA
  <uzfno> <strt> [<aux(naux)>] [<boundname>]
  <uzfno> <strt> [<aux(naux)>] [<boundname>]
  ...
END PACKAGEDATA
```

FOR ANY STRESS PERIOD

```
BEGIN PERIOD <iper>
  <uzfno> <uztsetting>
  <uzfno> <uztsetting>
  ...
END PERIOD
```

Explanation of Variables

Block: OPTIONS

- `flow_package_name` keyword to specify the name of the corresponding flow package. If not specified, then the corresponding flow package must have the same name as this advanced transport package (the name associated with this package in the GWT name file).
- `auxiliary` defines an array of one or more auxiliary variable names. There is no limit on the number of auxiliary variables that can be provided on this line; however, lists of information provided in subsequent blocks must have a column of data for each auxiliary variable name defined here. The number of auxiliary variables detected on this line determines the value for naux. Comments cannot be provided anywhere on this line as they will be interpreted as auxiliary variable names. Auxiliary variables may not be used by the package, but they will be available for use by other parts of the program. The program will terminate with an error if auxiliary variables are specified on more than one line in the options block.
- `flow_package_auxiliary_name` keyword to specify the name of an auxiliary variable in the corresponding flow package. If specified, then the simulated concentrations from this advanced transport package will be copied into the auxiliary variable specified with this name. Note that the flow package must have an auxiliary variable with this name or the program will terminate with an error. If the flows for this advanced transport package are read from a file, then this option will have no affect.
- `BOUNDNAMES` keyword to indicate that boundary names may be provided with the list of unsaturated zone flow cells.
- `PRINT_INPUT` keyword to indicate that the list of unsaturated zone flow information will be written to the listing file immediately after it is read.
- `PRINT_CONCENTRATION` keyword to indicate that the list of UZF cell concentration will be printed to the listing file for every stress period in which "HEAD PRINT" is specified in Output Control. If there is no Output Control option and PRINT_CONCENTRATION is specified, then concentration are printed for the last time step of each stress period.
- `PRINT_FLOWS` keyword to indicate that the list of unsaturated zone flow rates will be printed to the listing file for every stress period time step in which "BUDGET PRINT" is specified in Output Control. If there is no Output Control option and "PRINT_FLOWS" is specified, then flow rates are printed for the last time step of each stress period.
- `SAVE_FLOWS` keyword to indicate that unsaturated zone flow terms will be written to the file specified with "BUDGET FILEOUT" in Output Control.
- `CONCENTRATION` keyword to specify that record corresponds to concentration.
- `concfile` name of the binary output file to write concentration information.
- `BUDGET` keyword to specify that record corresponds to the budget.
- `FILEOUT` keyword to specify that an output filename is expected next.
- `budgetfile` name of the binary output file to write budget information.
- `TS6` keyword to specify that record corresponds to a time-series file.

- FILEIN keyword to specify that an input filename is expected next.
- ts6_filename defines a time-series file defining time series that can be used to assign time-varying values. See the "Time-Variable Input" section for instructions on using the time-series capability.
- OBS6 keyword to specify that record corresponds to an observations file.
- obs6_filename name of input file to define observations for the UZT package. See the "Observation utility" section for instructions for preparing observation input files. Tables ref{table:gwf-obstypetable} and ref{table:gwt-obstypetable} lists observation type(s) supported by the UZT package.

Block: PACKAGEDATA

- uzfno integer value that defines the UZF cell number associated with the specified PACKAGEDATA data on the line. UZFNO must be greater than zero and less than or equal to NUZFCELLS. Unsaturated zone flow information must be specified for every UZF cell or the program will terminate with an error. The program will also terminate with an error if information for a UZF cell is specified more than once.
- strt real value that defines the starting concentration for the unsaturated zone flow cell.
- aux represents the values of the auxiliary variables for each unsaturated zone flow. The values of auxiliary variables must be present for each unsaturated zone flow. The values must be specified in the order of the auxiliary variables specified in the OPTIONS block. If the package supports time series and the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- boundname name of the unsaturated zone flow cell. BOUNDNAME is an ASCII character variable that can contain as many as 40 characters. If BOUNDNAME contains spaces in it, then the entire name must be enclosed within single quotes.

Block: PERIOD

- iper integer value specifying the starting stress period number for which the data specified in the PERIOD block apply. IPER must be less than or equal to NPER in the TDIS Package and greater than zero. The IPER value assigned to a stress period block must be greater than the IPER value assigned for the previous PERIOD block. The information specified in the PERIOD block will continue to apply for all subsequent stress periods, unless the program encounters another PERIOD block.
- uzfno integer value that defines the UZF cell number associated with the specified PERIOD data on the line. UZFNO must be greater than zero and less than or equal to NUZFCELLS.
- uztsetting line of information that is parsed into a keyword and values. Keyword values that can be used to start the UZTSETTING string include: STATUS, CONCENTRATION, INFILTRATION, UZET, and AUXILIARY. These settings are used to assign the concentration of associated with the corresponding flow terms. Concentrations cannot be specified for all flow terms.

```
STATUS <status>
CONCENTRATION <concentration>
INFILTRATION <infiltration>
UZET <uzet>
AUXILIARY <auxname> <auxval>
```

- status keyword option to define UZF cell status. STATUS can be ACTIVE, INACTIVE, or CONSTANT. By default, STATUS is ACTIVE, which means that concentration will be calculated for the UZF cell. If a UZF cell is inactive, then there will be no solute mass fluxes into or out of the UZF cell and the inactive value will be written for the UZF cell concentration. If a UZF cell is constant, then the concentration for the UZF cell will be fixed at the user specified value.

- `concentration` real or character value that defines the concentration for the unsaturated zone flow cell. The specified CONCENTRATION is only applied if the unsaturated zone flow cell is a constant concentration cell. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `infiltration` real or character value that defines the infiltration solute concentration (ML-3) for the UZF cell. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `uzet` real or character value that defines the concentration of unsaturated zone evapotranspiration water (ML-3) for the UZF cell. If this concentration value is larger than the simulated concentration in the UZF cell, then the unsaturated zone ET water will be removed at the same concentration as the UZF cell. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.
- `AUXILIARY` keyword for specifying auxiliary variable.
- `auxname` name for the auxiliary variable to be assigned AUXVAL. AUXNAME must match one of the auxiliary variable names defined in the OPTIONS block. If AUXNAME does not match one of the auxiliary variable names defined in the OPTIONS block the data are ignored.
- `auxval` value for the auxiliary variable. If the Options block includes a TIMESERIESFILE entry (see the "Time-Variable Input" section), values can be obtained from a time series by entering the time-series name in place of a numeric value.

Example Input File

```
BEGIN OPTIONS
  AUXILIARY  aux1  aux2
  BOUNDNAMES
  PRINT_INPUT
  PRINT_CONCENTRATION
  PRINT_FLOWS
  SAVE_FLOWS
  CONCENTRATION  FILEOUT  gwt_02.uzt.bin
  BUDGET  FILEOUT  gwt_02.uzt.bud
  OBS6  FILEIN  gwt_02.uzt.obs
END OPTIONS

BEGIN PACKAGEDATA
# L             STRT             aux1             aux2    bname
  1       0.00000000      99.00000000     999.00000000  MYUZFCELL1
  2       0.00000000      99.00000000     999.00000000  MYUZFCELL2
  3       0.00000000      99.00000000     999.00000000  MYUZFCELL3
END PACKAGEDATA

BEGIN PERIOD  1
  1  STATUS  ACTIVE
  2  STATUS  ACTIVE
  3  STATUS  ACTIVE
END PERIOD  1
```

Available Observation Types

Example Observation Input File

```
BEGIN options
  DIGITS  7
  PRINT_INPUT
END options

BEGIN continuous  FILEOUT  gwt_02.uzt.obs.csv
  mwt-1-conc  CONCENTRATION  1
  mwt-1-stor  STORAGE  1
  mwt-1-gwt1  UZT  1  1
  mwt-1-gwt2  UZT  2  2
  mwt-2-gwt1  UZT  3  3
END continuous
```

1.5 Model Exchanges

1.5.1 EXG-GWFGWF

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [AUXILIARY <auxiliary(naux)>]
  [PRINT_INPUT]
  [PRINT_FLOWS]
  [SAVE_FLOWS]
  [CELL_AVERAGING <cell_averaging>]
  [VARIABLECV [DEWATERED]]
  [NEWTON]
  [GNC6 FILEIN <gnc6_filename>]
  [MVR6 FILEIN <mvr6_filename>]
  [OBS6 FILEIN <obs6_filename>]
END OPTIONS
```

```
BEGIN DIMENSIONS
  NEXG <nexg>
END DIMENSIONS
```

```
BEGIN EXCHANGEDATA
  <cellidm1> <cellidm2> <ihc> <cl1> <cl2> <hwva> [<aux(naux)>]
  <cellidm1> <cellidm2> <ihc> <cl1> <cl2> <hwva> [<aux(naux)>]
  ...
END EXCHANGEDATA
```

Explanation of Variables

Block: OPTIONS

- auxiliary an array of auxiliary variable names. There is no limit on the number of auxiliary variables that can be provided. Most auxiliary variables will not be used by the GWF-GWF Exchange, but they will be available for use by other parts of the program. If an auxiliary variable with the name "ANGLDEGX" is found, then this information will be used as the angle (provided in degrees) between the connection face normal and the x axis, where a value of zero indicates that a normal vector points directly along the positive x axis. The connection face normal is a normal vector on the cell face shared between the cell in model 1 and the cell in model 2 pointing away from the model 1 cell. Additional information on "ANGLDEGX" is provided in the description of the DISU Package. If an auxiliary variable with the name "CDIST" is found, then this information will be used as the straight-line connection distance, including the vertical component, between the two cell centers. Both ANGLDEGX and CDIST are required if specific discharge is calculated for either of the groundwater models.
- PRINT_INPUT keyword to indicate that the list of exchange entries will be echoed to the listing file immediately after it is read.
- PRINT_FLOWS keyword to indicate that the list of exchange flow rates will be printed to the listing file for every stress period in which "SAVE BUDGET" is specified in Output Control.
- SAVE_FLOWS keyword to indicate that cell-by-cell flow terms will be written to the budget file for each model provided that the Output Control for the models are set up with the "BUDGET SAVE FILE" option.
- cell_averaging is a keyword and text keyword to indicate the method that will be used for calculating the conductance for horizontal cell connections. The text value for CELL_AVERAGING can be "HARMONIC", "LOGARITHMIC", or "AMT-LMK", which means "arithmetic-mean thickness and logarithmic-mean hydraulic conductivity". If the user does not specify a value for CELL_AVERAGING, then the harmonic-mean method will be used.
- VARIABLECV keyword to indicate that the vertical conductance will be calculated using the saturated thickness and properties of the overlying cell and the thickness and properties of the underlying cell. If the DEWATERED keyword is also specified, then the vertical conductance is calculated using only the saturated thickness and properties of the overlying cell if the head in the underlying cell is below its top. If these keywords are not specified, then the default condition is to calculate the vertical conductance at the start of the simulation using the initial head and the cell properties. The vertical conductance remains constant for the entire simulation.
- DEWATERED If the DEWATERED keyword is specified, then the vertical conductance is calculated using only the saturated thickness and properties of the overlying cell if the head in the underlying cell is below its top.
- NEWTON keyword that activates the Newton-Raphson formulation for groundwater flow between connected, convertible groundwater cells. Cells will not dry when this option is used.
- FILEIN keyword to specify that an input filename is expected next.
- GNC6 keyword to specify that record corresponds to a ghost-node correction file.
- gnc6_filename is the file name for ghost node correction input file. Information for the ghost nodes are provided in the file provided with these keywords. The format for specifying the ghost nodes is the same as described for the GNC Package of the GWF Model. This includes specifying OPTIONS, DIMENSIONS, and GNCDATA blocks. The order of the ghost nodes must follow the same order as the order of the cells in the EXCHANGEDATA block. For the GNCDATA, noden and all of the nodej values are assumed to be located in model 1, and nodem is assumed to be in model 2.
- MVR6 keyword to specify that record corresponds to a mover file.
- mvr6_filename is the file name of the water mover input file to apply to this exchange. Information for the water mover are provided in the file provided with these keywords. The format for specifying the water mover information is the same as described for the Water Mover (MVR) Package of the GWF Model, with two

exceptions. First, in the PACKAGES block, the model name must be included as a separate string before each package. Second, the appropriate model name must be included before package name 1 and package name 2 in the BEGIN PERIOD block. This allows providers and receivers to be located in both models listed as part of this exchange.

- `OBS6` keyword to specify that record corresponds to an observations file.
- `obs6_filename` is the file name of the observations input file for this exchange. See the "Observation utility" section for instructions for preparing observation input files. Table ref{table:gwf-obstypetable} lists observation type(s) supported by the GWF-GWF package.

Block: DIMENSIONS

- `nexg` keyword and integer value specifying the number of GWF-GWF exchanges.

Block: EXCHANGEDATA

- `cellidm1` is the cellid of the cell in model 1 as specified in the simulation name file. For a structured grid that uses the DIS input file, CELLIDM1 is the layer, row, and column numbers of the cell. For a grid that uses the DISV input file, CELLIDM1 is the layer number and CELL2D number for the two cells. If the model uses the unstructured discretization (DISU) input file, then CELLIDM1 is the node number for the cell.
- `cellidm2` is the cellid of the cell in model 2 as specified in the simulation name file. For a structured grid that uses the DIS input file, CELLIDM2 is the layer, row, and column numbers of the cell. For a grid that uses the DISV input file, CELLIDM2 is the layer number and CELL2D number for the two cells. If the model uses the unstructured discretization (DISU) input file, then CELLIDM2 is the node number for the cell.
- `ihc` is an integer flag indicating the direction between node n and all of its m connections. If IHC = 0 then the connection is vertical. If IHC = 1 then the connection is horizontal. If IHC = 2 then the connection is horizontal for a vertically staggered grid.
- `cl1` is the distance between the center of cell 1 and the its shared face with cell 2.
- `cl2` is the distance between the center of cell 2 and the its shared face with cell 1.
- `hwva` is the horizontal width of the flow connection between cell 1 and cell 2 if IHC > 0, or it is the area perpendicular to flow of the vertical connection between cell 1 and cell 2 if IHC = 0.
- `aux` represents the values of the auxiliary variables for each GWFGWF Exchange. The values of auxiliary variables must be present for each exchange. The values must be specified in the order of the auxiliary variables specified in the OPTIONS block.

Example Input File

```
BEGIN OPTIONS
  PRINT_INPUT
  PRINT_FLOWS
  SAVE_FLOWS
  AUXILIARY testaux
  GNC6 FILEIN simulation.gnc
  MVR6 FILEIN simulation.mvr
END OPTIONS

BEGIN DIMENSIONS
  NEXG 36
```

(continues on next page)

(continued from previous page)

```
END DIMENSIONS

  # nodem1 nodem2  ihc  cl1   cl2    fahl  testaux
BEGIN EXCHANGEDATA
#
#       left side
        16      1    1  50. 16.67 33.33   100.99
        16     10    1  50. 16.67 33.33   100.99
        16     19    1  50. 16.67 33.33   100.99
        23     28    1  50. 16.67 33.33   100.99
        23     37    1  50. 16.67 33.33   100.99
        23     46    1  50. 16.67 33.33   100.99
        30     55    1  50. 16.67 33.33   100.99
        30     64    1  50. 16.67 33.33   100.99
        30     73    1  50. 16.67 33.33   100.99
#
#       right side
        20      9    1  50. 16.67 33.33   100.99
        20     18    1  50. 16.67 33.33   100.99
        20     27    1  50. 16.67 33.33   100.99
        27     36    1  50. 16.67 33.33   100.99
        27     45    1  50. 16.67 33.33   100.99
        27     54    1  50. 16.67 33.33   100.99
        34     63    1  50. 16.67 33.33   100.99
        34     72    1  50. 16.67 33.33   100.99
        34     81    1  50. 16.67 33.33   100.99
#
#       back
        10      1    1  50. 17.67 33.33   100.99
        10      2    1  50. 17.67 33.33   100.99
        10      3    1  50. 17.67 33.33   100.99
        11      4    1  50. 17.67 33.33   100.99
        11      5    1  50. 17.67 33.33   100.99
        11      6    1  50. 17.67 33.33   100.99
        12      7    1  50. 17.67 33.33   100.99
        12      8    1  50. 17.67 33.33   100.99
        12      9    1  50. 17.67 33.33   100.99
#
#       front
        38     73    1  50. 17.67 33.33   100.99
        38     74    1  50. 17.67 33.33   100.99
        38     75    1  50. 17.67 33.33   100.99
        39     76    1  50. 17.67 33.33   100.99
        39     77    1  50. 17.67 33.33   100.99
        39     78    1  50. 17.67 33.33   100.99
        40     79    1  50. 17.67 33.33   100.99
        40     80    1  50. 17.67 33.33   100.99
        40     81    1  50. 17.67 33.33   100.99
END EXCHANGEDATA
```

Example Observation Input File

```
BEGIN OPTIONS
  DIGITS 10
  PRINT_INPUT
END OPTIONS

# Block defining continuous observations
BEGIN CONTINUOUS FILEOUT simulation.obs.csv
# obsname        obstype
  exg1           flow-ja-face   1
END CONTINUOUS
```

1.5.2 EXG-GWFGWT

Structure of Blocks

FOR EACH SIMULATION

Explanation of Variables

1.6 Utilities

1.6.1 UTL-LAK-TAB

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN DIMENSIONS
  NROW <nrow>
  NCOL <ncol>
END DIMENSIONS
```

```
BEGIN TABLE
  <stage> <volume> <sarea> [<barea>]
  <stage> <volume> <sarea> [<barea>]
  ...
END TABLE
```

Explanation of Variables

Block: DIMENSIONS

- `nrow` integer value specifying the number of rows in the lake table. There must be NROW rows of data in the TABLE block.
- `ncol` integer value specifying the number of columns in the lake table. There must be NCOL columns of data in the TABLE block. For lakes with HORIZONTAL and/or VERTICAL CTYPE connections, NCOL must be equal to 3. For lakes with EMBEDDEDH or EMBEDDEDV CTYPE connections, NCOL must be equal to 4.

Block: TABLE

- `stage` real value that defines the stage corresponding to the remaining data on the line.
- `volume` real value that defines the lake volume corresponding to the stage specified on the line.
- `sarea` real value that defines the lake surface area corresponding to the stage specified on the line.
- `barea` real value that defines the lake-GWF exchange area corresponding to the stage specified on the line. BAREA is only specified if the CLAKTYPE for the lake is EMBEDDEDH or EMBEDDEDV.

Example Input File

```
begin dimensions
  nrow 11
  ncol  3
end dimensions

begin table
# stage    volume     sarea
      0        0.        0.
      1       0.5        1.
      2       1.0        2.
      3       2.0        2.
      4       3.0        2.
      5       4.0        2.
      6       5.0        2.
      7       6.0        2.
      8       7.0        2.
      9       8.0        2.
     10       9.0        2.
end table
```

1.6.2 UTL-OBS

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN OPTIONS
  [DIGITS <digits>]
  [PRINT_INPUT]
END OPTIONS
```

```
BEGIN CONTINUOUS FILEOUT <obs_output_file_name> [BINARY]
  <obsname> <obstype> <id> [<id2>]
  <obsname> <obstype> <id> [<id2>]
  ...
END CONTINUOUS
```

Explanation of Variables

Block: OPTIONS

- `digits` Keyword and an integer digits specifier used for conversion of simulated values to text on output. The default is 5 digits. When simulated values are written to a file specified as file type DATA in the Name File, the digits specifier controls the number of significant digits with which simulated values are written to the output file. The digits specifier has no effect on the number of significant digits with which the simulation time is written for continuous observations.
- `PRINT_INPUT` keyword to indicate that the list of observation information will be written to the listing file immediately after it is read.

Block: CONTINUOUS

- `FILEOUT` keyword to specify that an output filename is expected next.
- `obs_output_file_name` Name of a file to which simulated values corresponding to observations in the block are to be written. The file name can be an absolute or relative path name. A unique output file must be specified for each SINGLE or CONTINUOUS block. If the "BINARY" option is used, output is written in binary form. By convention, text output files have the extension "csv" (for "Comma-Separated Values") and binary output files have the extension "bsv" (for "Binary Simulated Values").
- `BINARY` an optional keyword used to indicate that the output file should be written in binary (unformatted) form.
- `obsname` string of 1 to 40 nonblank characters used to identify the observation. The identifier need not be unique; however, identification and post-processing of observations in the output files are facilitated if each observation is given a unique name.
- `obstype` a string of characters used to identify the observation type.
- `id` Text identifying cell where observation is located. For packages other than NPF, if boundary names are defined in the corresponding package input file, ID can be a boundary name. Otherwise ID is a cellid. If the model discretization is type DIS, cellid is three integers (layer, row, column). If the discretization is DISV, cellid is two integers (layer, cell number). If the discretization is DISU, cellid is one integer (node number).
- `id2` Text identifying cell adjacent to cell identified by ID. The form of ID2 is as described for ID. ID2 is used for intercell-flow observations of a GWF model, for three observation types of the LAK Package, for two observation types of the MAW Package, and one observation type of the UZF Package.

Example Observation Input File

Example 1

```
BEGIN OPTIONS
  DIGITS 10
  PRINT_INPUT
END OPTIONS

BEGIN CONTINUOUS FILEOUT my_model.gwf.head.csv
# obsname   obstype   ID
  L1        HEAD      1 51 51 # heads at lay 1 row 51 col 51
  L2        HEAD      2 51 51 # heads at lay 2 row 51 col 51
END CONTINUOUS
```

(continues on next page)

(continued from previous page)

```
  BEGIN CONTINUOUS FILEOUT my_model.gwf.ddn.csv
  # obsname   obstype   ID
    L1ddn     DRAWDOWN  1 51 51 # heads at lay 1 row 51 col 51
    L2ddn     DRAWDOWN  2 51 51 # heads at lay 2 row 51 col 51
  END CONTINUOUS

  BEGIN CONTINUOUS FILEOUT my_model.gwf.flow.csv
  # obsname    obstype       ID       ID1
    L1rfflow   FLOW-JA-FACE  1 51 51  1 51 52
    L2rfflow   FLOW-JA-FACE  2 51 51  2 51 52
    L1-L2flow  FLOW-JA-FACE  1 51 51  2 51 51
  END CONTINUOUS
```

Example Observation Input File

Example 2

```
  BEGIN OPTIONS
    DIGITS 10
    PRINT_INPUT
  END OPTIONS

  BEGIN CONTINUOUS FILEOUT my_model.gwt.conc.csv
  # obsname   obstype         ID
    L1        CONCENTRATION   1 51 51 # concs at lay 1 row 51 col 51
    L2        CONCENTRATION   2 51 51 # concs at lay 2 row 51 col 51
  END CONTINUOUS

  BEGIN CONTINUOUS FILEOUT my_model.gwt.mflow.csv
  # obsname    obstype       ID       ID1
    L1rfflow   FLOW-JA-FACE  1 51 51  1 51 52
    L2rfflow   FLOW-JA-FACE  2 51 51  2 51 52
    L1-L2flow  FLOW-JA-FACE  1 51 51  2 51 51
  END CONTINUOUS
```

1.6.3 UTL-TAS

Structure of Blocks

FOR EACH SIMULATION

```
  BEGIN ATTRIBUTES
    NAME <time_series_nameany1d>
    NAME
     <time_series_nameany1d>
    [METHOD <interpolation_method>]
    METHOD
     <interpolation_method>
    [SFAC <sfacvaltime_series_name>]
    SFAC
```

(continues on next page)

(continued from previous page)

```
    <sfacvaltime_series_name>
END ATTRIBUTES
```

```
BEGIN TIME <time_from_model_start>

        <tas_array(unknown)> -- READARRAY
END TIME
```

Explanation of Variables

Block: ATTRIBUTES

- NAME xxx
- time_series_name Name by which a package references a particular time-array series. The name must be unique among all time-array series used in a package.
- METHOD xxx
- interpolation_method Interpolation method, which is either STEPWISE or LINEAR.
- SFAC xxx
- sfacval Scale factor, which will multiply all array values in time series. SFAC is an optional attribute; if omitted, SFAC = 1.0.

Block: TIME

- time_from_model_start A numeric time relative to the start of the simulation, in the time unit used in the simulation. Times must be strictly increasing.
- tas_array An array of numeric, floating-point values, or a constant value, readable by the U2DREL array-reading utility.

1.6.4 UTL-TS

Structure of Blocks

FOR EACH SIMULATION

```
BEGIN ATTRIBUTES
  NAMES <time_series_namesany1d>
  NAMES
   <time_series_namesany1d>
  [METHODS <interpolation_methodtime_series_names>]
  METHODS
   <interpolation_methodtime_series_names>
  [METHOD <interpolation_method_single>]
  METHOD
   <interpolation_method_single>
  [SFACS <sfacval<time_series_name>]
  SFACS
   <sfacval<time_series_name>
```

(continues on next page)

(continued from previous page)

```
    [<sfacval<time_series_name>]

  END ATTRIBUTES
```

```
  BEGIN TIMESERIES
    <ts_time> <ts_arraytime_series_names>
    <ts_time> <ts_arraytime_series_names>
    ...
     <ts_time>
     <ts_arraytime_series_names>
  END TIMESERIES
```

Explanation of Variables

Block: ATTRIBUTES

- NAMES xxx
- time_series_names Name by which a package references a particular time-array series. The name must be unique among all time-array series used in a package.
- METHODS xxx
- interpolation_method Interpolation method, which is either STEPWISE or LINEAR.
- METHOD xxx
- interpolation_method_single Interpolation method, which is either STEPWISE or LINEAR.
- SFACS xxx
- sfacval Scale factor, which will multiply all array values in time series. SFAC is an optional attribute; if omitted, SFAC = 1.0.
- SFAC xxx

Block: TIMESERIES

- ts_time A numeric time relative to the start of the simulation, in the time unit used in the simulation. Times must be strictly increasing.
- ts_array A 2-D array of numeric, floating-point values, or a constant value, readable by the U2DREL array-reading utility.

CHAPTER

TWO

MODFLOW 6 SOURCE CODE

2.1 Class Hierarchy

2.2 File Hierarchy

2.3 Full API

2.3.1 Namespaces

Namespace bmif

Contents

Variables

Namespace KindModule

Namespace memoryhelpermodule

Contents

Functions

Variables

Namespace memorylistmodule

Contents

Functions

Namespace memorymanagermodule

Contents

Functions

- *Function memorymanagermodule::deallocate_str*
- *Function memorymanagermodule::deallocate_str1d*
- *Function memorymanagermodule::get_from_memorylist*
- *Function memorymanagermodule::get_isize*
- *Function memorymanagermodule::get_mem_elem_size*
- *Function memorymanagermodule::get_mem_rank*
- *Function memorymanagermodule::get_mem_shape*
- *Function memorymanagermodule::get_mem_type*
- *Function memorymanagermodule::mem_cleanup_table*
- *Function memorymanagermodule::mem_da*
- *Function memorymanagermodule::mem_detailed_table*
- *Function memorymanagermodule::mem_set_print_option*
- *Function memorymanagermodule::mem_summary_line*
- *Function memorymanagermodule::mem_summary_table*
- *Function memorymanagermodule::mem_summary_total*
- *Function memorymanagermodule::mem_unique_origins*
- *Function memorymanagermodule::mem_units*
- *Function memorymanagermodule::mem_write_usage*
- *Function memorymanagermodule::reallocate_dbl1d*
- *Function memorymanagermodule::reallocate_dbl2d*
- *Function memorymanagermodule::reallocate_int1d*
- *Function memorymanagermodule::reallocate_int2d*
- *Function memorymanagermodule::reallocate_str1d*
- *Function memorymanagermodule::reassignptr_dbl1d*
- *Function memorymanagermodule::reassignptr_dbl2d*
- *Function memorymanagermodule::reassignptr_int1d*
- *Function memorymanagermodule::reassignptr_int2d*
- *Function memorymanagermodule::setptr_dbl*
- *Function memorymanagermodule::setptr_dbl1d*
- *Function memorymanagermodule::setptr_dbl2d*
- *Function memorymanagermodule::setptr_int*
- *Function memorymanagermodule::setptr_int1d*
- *Function memorymanagermodule::setptr_int2d*
- *Function memorymanagermodule::setptr_logical*

Variables

Namespace memorysethandlermodule

Contents

Functions

Variables

Namespace memorytypemodule

Contents

Functions

- *Function memorytypemodule::mt_associated*
- *Function memorytypemodule::table_entry*

Namespace mf6bmi

This module contains the MODFLOW 6 BMI.

Contents

Detailed Description

This BMI interface matches the CSDMS standard, with a few modifications:

- This interface will build into a shared library that can be called from other executables and scripts, not necessarily written in Fortran. Therefore we have omitted the type-boundness of the routines, since they cannot have the bind(C,"...") attribute.
- MODFLOW has internal data arrays with rank > 1 that we would like to expose. An example would be access to data in the BOUND array of GWF boundary packages (BndType). The get_value_ptr calls below support this, returning a C-style pointer to the arrays and methods have been added to query the variable's rank and shape.

Note on style: BMI apparently uses underscores, we use underscores in some places but camelcase in other. Since this is a dedicated BMI interface module, we'll use underscores here as well.

Functions

- *Function mf6bmi::bmi_finalize*
- *Function mf6bmi::bmi_initialize*
- *Function mf6bmi::bmi_update*
- *Function mf6bmi::get_current_time*
- *Function mf6bmi::get_end_time*
- *Function mf6bmi::get_input_item_count*
- *Function mf6bmi::get_input_var_names*
- *Function mf6bmi::get_output_item_count*
- *Function mf6bmi::get_output_var_names*
- *Function mf6bmi::get_start_time*
- *Function mf6bmi::get_time_step*
- *Function mf6bmi::get_value_double*

- *Function mf6bmi::get_value_int*
- *Function mf6bmi::get_value_ptr_double*
- *Function mf6bmi::get_value_ptr_int*
- *Function mf6bmi::get_var_itemsize*
- *Function mf6bmi::get_var_nbytes*
- *Function mf6bmi::get_var_rank*
- *Function mf6bmi::get_var_shape*
- *Function mf6bmi::get_var_type*
- *Function mf6bmi::set_value_double*
- *Function mf6bmi::set_value_int*

Variables

- *Variable mf6bmi::bind*
- *Variable mf6bmi::istdout_to_file*

Namespace mf6bmigrid

This module contains BMI routines to expose the MODFLOW 6 discretization.

Contents

Detailed Description

NB: this module is experimental and still under development.

Functions

- *Function mf6bmigrid::get_grid_face_count*
- *Function mf6bmigrid::get_grid_face_nodes*
- *Function mf6bmigrid::get_grid_node_count*
- *Function mf6bmigrid::get_grid_nodes_per_face*
- *Function mf6bmigrid::get_grid_rank*
- *Function mf6bmigrid::get_grid_shape*
- *Function mf6bmigrid::get_grid_size*
- *Function mf6bmigrid::get_grid_type*

- *Function mf6bmigrid::get_grid_x*
- *Function mf6bmigrid::get_grid_y*
- *Function mf6bmigrid::get_var_grid*

Namespace mf6bmiUtil

Namespace mf6bmiutil

This module contains helper routines and parameters for the MODFLOW 6 BMI.

Contents

Functions

- *Function mf6bmiutil::char_array_to_string*
- *Function mf6bmiutil::confirm_grid_type*
- *Function mf6bmiutil::extract_model_name*
- *Function mf6bmiutil::get_grid_type_model*
- *Function mf6bmiutil::get_memory_access_type*
- *Function mf6bmiutil::get_model_name*
- *Function mf6bmiutil::getsolution*
- *Function mf6bmiutil::split_address*
- *Function mf6bmiutil::string_to_char_array*
- *Function mf6bmiutil::strlen*

Variables

- *Variable mf6bmiutil::bind*
- *Variable mf6bmiutil::bmi_lengridtype*
- *Variable mf6bmiutil::bmi_lenvaraddress*
- *Variable mf6bmiutil::bmi_lenvartype*
- *Variable mf6bmiutil::lengridtype*

Namespace Mf6CoreModule

Namespace mf6xmi

This module contains the eXtended Model Interface.

Contents

Detailed Description

In this module we expose functionality in the MODFLOW 6 shared library, that is *beyond* the basic model interface: https://bmi-spec.readthedocs.io/en/latest/. The main extensions are

- Controlling the kernel at a finer granularity, isolating the call to the linear solve of the system. This way, the interface can be used to set up a non-linear coupling with, for example, an external unsaturated zone model.
- Expose the concept of subcomponents, which in case of MODFLOW 6 are 'Numerical Solution' objects, each of which represents a separate linear system to solve. An example here would be a transport model (GWT) coupled to a groundwater model (GWF).

The common BMI control flow is initialize()

whilet<t_end: update()

finalize()

With the XMI you can now also use it as: initialize()

while(t<t_end):

prepare_time_step()

#modifysomevalueshere

do_time_step()

#andmaybesomethinghereaswell

finalize_time_step()

finalize()

Or, when you want to isolate the call to the linear solve, a typical application could look like this: initialize()

whilet<t_end:

prepare_time_step()

fori_solinsolutions:

prepare_solve(i_sol)

whilek_iter<max_iter:

#exchangecoupledvariables exchange_data()

#theMODFLOWlinearsolve: solve()

#maybesolvesomeother,externalmodelhere: solveExternalModel()

#andexchangeback exchange_data()

#checkforconvergence convergence_check()

finalize_solve(i_sol)

finalize_time_step()

finalize()

Note that the last example can only work when there is a single Solution Group defined. This will typically not be a problem, though, as applications with multiple Solution Groups should be quite rare.

Functions

- *Function mf6xmi::get_var_address*
- *Function mf6xmi::xmi_do_time_step*
- *Function mf6xmi::xmi_finalize_solve*
- *Function mf6xmi::xmi_finalize_time_step*
- *Function mf6xmi::xmi_get_subcomponent_count*
- *Function mf6xmi::xmi_prepare_solve*
- *Function mf6xmi::xmi_prepare_time_step*
- *Function mf6xmi::xmi_solve*

Variables

- *Variable mf6xmi::iterationcounter*

Namespace NumericalSolutionModule

Namespace SolutionGroupModule

2.3.2 Functions

Function memoryhelpermodule::create_mem_address

- Defined in file_src_Utilities_Memory_MemoryHelper.f90

Function Documentation

character(len=lenmemaddress) function memoryhelpermodule::create_mem_address (mem_path, va
returns the address string of the memory object

Returns the memory address, i.e. the full path plus name of the stored variable

NB: no need to trim the input parameters

Return full address string to the memory object

Parameters

- `[in] mem_path`: path to the memory object
- `[in] var_name`: name of the stored variable

Function memoryhelpermodule::create_mem_path

- Defined in file_src_Utilities_Memory_MemoryHelper.f90

Function Documentation

character(len=lenmempath) function memoryhelpermodule::create_mem_path (component, subcompo
returns the path to the memory object

Returns the path to the location in the memory manager where the variables for this (sub)component are stored, the 'memoryPath'

NB: no need to trim the input parameters

Return the memory path

Parameters

- `[in] component`: name of the solution, model, or exchange
- `[in] subcomponent`: name of the package (optional)

Function memoryhelpermodule::mem_check_length

- Defined in file_src_Utilities_Memory_MemoryHelper.f90

Function Documentation

subroutine `memoryhelpermodule`**::mem_check_length**(name, max_length, description)
Generic routine to check the length of (parts of) the memory address.

The string will be trimmed before the measurement.

The description should describe the part of the address that is checked (variable, package, model, solution, exchange name) or the full memory path itself

Warning {if the length exceeds the maximum, a message is recorded and the program will be stopped}

Parameters

- `[in] name`: string to be checked
- `[in] max_length`: maximum length
- `[in] description`: a descriptive string

Function memoryhelpermodule::split_mem_address

- Defined in file_src_Utilities_Memory_MemoryHelper.f90

Function Documentation

subroutine `memoryhelpermodule::`**`split_mem_address`**(mem_address, mem_path, var_name)
Split a memory address string into memory path and variable name.

Parameters

- `[in] mem_address`: the full memory address string
- `[out] mem_path`: the memory path
- `[out] var_name`: the variable name

Function memoryhelpermodule::split_mem_path

- Defined in file_src_Utilities_Memory_MemoryHelper.f90

Function Documentation

subroutine `memoryhelpermodule::`**`split_mem_path`**(mem_path, component, subcomponent)
Split the memory path into component(s)

NB: when there is no subcomponent in the path, the value for

subcomponent is set to an empty string.

Parameters

- `[in] mem_path`: path to the memory object
- `[out] component`: name of the component (solution, model, exchange)
- `[out] subcomponent`: name of the subcomponent (package)

Function memorylistmodule::add

- Defined in file_src_Utilities_Memory_MemoryList.f90

Function Documentation

subroutine memorylistmodule::add (this, mt)

Function memorylistmodule::clear

- Defined in file_src_Utilities_Memory_MemoryList.f90

Function Documentation

subroutine memorylistmodule::clear (this)

Function memorylistmodule::count

- Defined in file_src_Utilities_Memory_MemoryList.f90

Function Documentation

integer(i4b) function memorylistmodule::count (this)

Function memorylistmodule::get

- Defined in file_src_Utilities_Memory_MemoryList.f90

Function Documentation

type(memorytype) function, pointer memorylistmodule::get (this, ipos)

Function memorymanagermodule::allocate_dbl

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule`**::allocate_dbl**(sclr, name, mem_path, memtype)

Allocate a real scalar.

Parameters

- `[inout] sclr`: variable for allocation
- `[in] name`: variable name
- `[in] mem_path`: path where variable is stored
- `[in] memtype`: optional integer value that defines memaccess for variable name. valid values are MEMHIDDEN, MEMREADONLY, and MEMREADWRITE.

Function memorymanagermodule::allocate_dbl1d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule`**`::allocate_dbl1d`**(adbl, nrow, name, mem_path, memtype)
Allocate a 1-dimensional real array.

Parameters

- `[inout] adbl`: variable for allocation
- `[in] nrow`: number of rows
- `[in] name`: variable name
- `[in] mem_path`: path where variable is stored
- `[in] memtype`: optional integer value that defines memaccess for variable name. valid values are MEMHIDDEN, MEMREADONLY, and MEMREADWRITE.

Function memorymanagermodule::allocate_dbl2d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule`**`::allocate_dbl2d`**(adbl, ncol, nrow, name, mem_path, memtype)
Allocate a 2-dimensional real array.

Parameters

- `[inout] adbl`: variable for allocation
- `[in] ncol`: number of columns
- `[in] nrow`: number of rows
- `[in] name`: variable name
- `[in] mem_path`: path where variable is stored
- `[in] memtype`: optional integer value that defines memaccess for variable name. valid values are MEMHIDDEN, MEMREADONLY, and MEMREADWRITE.

Function memorymanagermodule::allocate_dbl3d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine memorymanagermodule::**allocate_dbl3d**(adbl, ncol, nrow, nlay, name, mem_path, memtype)

Allocate a 3-dimensional real array.

Parameters

- [inout] adbl: variable for allocation
- [in] ncol: number of columns
- [in] nrow: number of rows
- [in] nlay: number of layers
- [in] name: variable name
- [in] mem_path: path where variable is stored
- [in] memtype: optional integer value that defines memaccess for variable name. valid values are MEMHIDDEN, MEMREADONLY, and MEMREADWRITE.

Function memorymanagermodule::allocate_error

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine memorymanagermodule::**allocate_error**(varname, mem_path, istat, isize)

Issue allocation error message and stop program execution.

Parameters

- [in] varname: variable name
- [in] mem_path: path where the variable is stored
- [in] istat: status code
- [in] isize: size of allocation

Function memorymanagermodule::allocate_int

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine memorymanagermodule::**allocate_int**(sclr, name, mem_path, memtype)

Allocate a integer scalar.

Parameters

- [inout] sclr: variable for allocation
- [in] name: variable name

- `[in] mem_path`: path where the variable is stored
- `[in] memtype`: optional integer value that defines memaccess for variable name. valid values are MEMHIDDEN, MEMREADONLY, and MEMREADWRITE.

Function memorymanagermodule::allocate_int1d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule`**`::allocate_int1d`**(aint, nrow, name, mem_path, memtype)
Allocate a 1-dimensional integer array.

Parameters

- `[inout] aint`: variable for allocation
- `[in] nrow`: integer array number of rows
- `[in] name`: variable name
- `[in] mem_path`: path where variable is stored
- `[in] memtype`: optional integer value that defines memaccess for variable name. valid values are MEMHIDDEN, MEMREADONLY, and MEMREADWRITE.

Function memorymanagermodule::allocate_int2d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule`**`::allocate_int2d`**(aint, ncol, nrow, name, mem_path, memtype)
Allocate a 2-dimensional integer array.

Parameters

- `[inout] aint`: variable for allocation
- `[in] ncol`: number of columns
- `[in] nrow`: number of rows
- `[in] name`: variable name
- `[in] mem_path`: path where variable is stored
- `[in] memtype`: optional integer value that defines memaccess for variable name. valid values are MEMHIDDEN, MEMREADONLY, and MEMREADWRITE.

Function memorymanagermodule::allocate_int3d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule::`**`allocate_int3d`**(aint, ncol, nrow, nlay, name, mem_path, memtype)

Allocate a 3-dimensional integer array.

Parameters

- `[inout] aint`: variable for allocation
- `[in] ncol`: number of columns
- `[in] nrow`: number of rows
- `[in] nlay`: number of layers
- `[in] name`: variable name
- `[in] mem_path`: path where variable is stored
- `[in] memtype`: optional integer value that defines memaccess for variable name. valid values are MEMHIDDEN, MEMREADONLY, and MEMREADWRITE.

Function memorymanagermodule::allocate_logical

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule::`**`allocate_logical`**(sclr, name, mem_path)

Allocate a logical scalar.

Parameters

- `[inout] sclr`: variable for allocation
- `[in] name`: variable name
- `[in] mem_path`: path where the variable is stored

Function memorymanagermodule::allocate_str

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine memorymanagermodule::**allocate_str**(sclr, ilen, name, mem_path)
Allocate a character string.

Parameters

- [in] ilen: string length
- [inout] sclr: variable for allocation
- [in] name: variable name
- [in] mem_path: path where the variable is stored

Function memorymanagermodule::allocate_str1d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine memorymanagermodule::**allocate_str1d**(astr, ilen, nrow, name, mem_path)
Allocate a 1-dimensional defined length string array.

Parameters

- [in] ilen: string length
- [inout] astr: variable for allocation
- [in] nrow: number of strings in array
- [in] name: variable name
- [in] mem_path: path where the variable is stored

Function memorymanagermodule::checkin_dbl1d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine memorymanagermodule::**checkin_dbl1d**(adbl, name, mem_path, name2, mem_path2, memtype)
Check in an existing 1d double precision array with a new address (name + path)

Parameters

- [inout] adbl: the existing array
- [in] name: new variable name
- [in] mem_path: new path where variable is stored
- [in] name2: existing variable name
- [in] mem_path2: existing path where variable is stored

- `[in] memtype`: optional integer value that defines memaccess for variable name. valid values are MEMHIDDEN, MEMREADONLY, and MEMREADWRITE.

Function memorymanagermodule::checkin_int1d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule`**`::checkin_int1d`**(aint, name, mem_path, name2, mem_path2, memtype)

Check in an existing 1d integer array with a new address (name + path)

Parameters

- `[in] aint`: the existing array
- `[in] name`: new variable name
- `[in] mem_path`: new path where variable is stored
- `[in] name2`: existing variable name
- `[in] mem_path2`: existing path where variable is stored
- `[in] memtype`: optional integer value that defines memaccess for variable name. valid values are MEMHIDDEN, MEMREADONLY, and MEMREADWRITE.

Function memorymanagermodule::copy_dbl1d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

`subroutine, public memorymanagermodule::copy_dbl1d (adbl, name, mem_path)`

Copy values from a 1-dimensional real array in the memory.

Parameters

- `[inout] adbl`: target array
- `[in] name`: variable name
- `[in] mem_path`: path where variable is stored

Function memorymanagermodule::copyptr_dbl1d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule::`**`copyptr_dbl1d`**(adbl, name, mem_path, mem_path_copy)
Make a copy of a 1-dimensional real array.

Parameters

- `[inout] adbl`: returned copy of 1d real array
- `[in] name`: variable name
- `[in] mem_path`: path where variable is stored
- `[in] mem_path_copy`: optional path where the copy wil be stored, if passed then the copy is added to the memory manager

Function memorymanagermodule::copyptr_dbl2d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule::`**`copyptr_dbl2d`**(adbl, name, mem_path, mem_path_copy)
Make a copy of a 2-dimensional real array.

Parameters

- `[inout] adbl`: returned copy of 2d real array
- `[in] name`: variable name
- `[in] mem_path`: path where variable is stored
- `[in] mem_path_copy`: optional path where the copy wil be stored, if passed then the copy is added to the memory manager

Function memorymanagermodule::copyptr_int1d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule::`**`copyptr_int1d`**(aint, name, mem_path, mem_path_copy)
Make a copy of a 1-dimensional integer array.

Parameters

- `[inout] aint`: returned copy of 1d integer array
- `[in] name`: variable name

- `[in] mem_path`: path where variable is stored
- `[in] mem_path_copy`: optional path where the copy wil be stored, if passed then the copy is added to the memory manager

Function memorymanagermodule::copyptr_int2d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule::`**`copyptr_int2d`**(aint, name, mem_path, mem_path_copy)
Make a copy of a 2-dimensional integer array.

Parameters

- `[inout] aint`: returned copy of 2d integer array
- `[in] name`: variable name
- `[in] mem_path`: path where variable is stored
- `[in] mem_path_copy`: optional path where the copy wil be stored, if passed then the copy is added to the memory manager

Function memorymanagermodule::deallocate_dbl

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule::`**`deallocate_dbl`**(sclr)
Deallocate a real scalar.

Parameters

- `[inout] sclr`: real variable to deallocate

Function memorymanagermodule::deallocate_dbl1d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule::`**`deallocate_dbl1d`**(adbl, name, mem_path)
Deallocate a 1-dimensional real array.

Parameters

- `[inout] adbl`: 1d real array to deallocate
- `name`: variable name
- `mem_path`: path where variable is stored

Function memorymanagermodule::deallocate_dbl2d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule`**`::deallocate_dbl2d`**(adbl, name, mem_path)
Deallocate a 2-dimensional real array.

Parameters

- `[inout] adbl`: 2d real array to deallocate
- `name`: variable name
- `mem_path`: path where variable is stored

Function memorymanagermodule::deallocate_dbl3d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule`**`::deallocate_dbl3d`**(adbl, name, mem_path)
Deallocate a 3-dimensional real array.

Parameters

- `[inout] adbl`: 3d real array to deallocate
- `name`: variable name
- `mem_path`: path where variable is stored

Function memorymanagermodule::deallocate_int

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule`**`::deallocate_int`**(sclr)
Deallocate a integer scalar.

Parameters

- `[inout] sclr`: integer variable to deallocate

Function memorymanagermodule::deallocate_int1d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule`**`::deallocate_int1d`**(aint, name, mem_path)
Deallocate a 1-dimensional integer array.

Parameters

- `[inout] aint`: 1d integer array to deallocate
- `name`: variable name
- `mem_path`: path where variable is stored

Function memorymanagermodule::deallocate_int2d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule`**`::deallocate_int2d`**(aint, name, mem_path)
Deallocate a 2-dimensional integer array.

Parameters

- `[inout] aint`: 2d integer array to deallocate
- `name`: variable name
- `mem_path`: path where variable is stored

Function memorymanagermodule::deallocate_int3d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule`**`::deallocate_int3d`**(aint, name, mem_path)
Deallocate a 3-dimensional integer array.

Parameters

- `[inout] aint`: 3d integer array to deallocate
- `name`: variable name
- `mem_path`: path where variable is stored

Function memorymanagermodule::deallocate_logical

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule`**`::deallocate_logical`**(sclr)
Deallocate a logical scalar.

Parameters

- `[inout] sclr`: logical scalar to deallocate

Function memorymanagermodule::deallocate_str

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule`**`::deallocate_str`**(sclr, name, mem_path)
Deallocate a variable-length character string.

Parameters

- `[inout] sclr`: pointer to string
- `[in] name`: variable name
- `[in] mem_path`: path where variable is stored

Function memorymanagermodule::deallocate_str1d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule`**`::deallocate_str1d`**(astr, name, mem_path)
Deallocate an array of variable-length character strings.

Todo:
confirm this description versus the previous doc

Parameters

- `[inout] astr`: array of strings
- `[in] name`: variable name
- `[in] mem_path`: path where variable is stored

Function memorymanagermodule::get_from_memorylist

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine, public memorymanagermodule::get_from_memorylist (name, mem_path, mt, found, che

@ brief Get a memory type entry from the memory list

Default value for

check is .true. which means that this routine will kill the program when the memory entry cannot be found.

Parameters

- `[in] name`: variable name
- `[in] mem_path`: path where the variable is stored
- `[inout] mt`: memory type entry
- `[out] found`: set to .true. when found
- `[in] check`: to suppress aborting the program when not found, set check = .false.

Function memorymanagermodule::get_isize

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine, public memorymanagermodule::get_isize (name, mem_path, isize)

@ brief Get the number of elements for this variable

Returns with isize = -1 when not found.

Parameters

- `[in] name`: variable name
- `[in] mem_path`: path where the variable is stored
- `[out] isize`: number of elements (flattened)

Function memorymanagermodule::get_mem_elem_size

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine, public memorymanagermodule::get_mem_elem_size (name, mem_path, size)
@ brief Get the memory size of a single element of the stored variable

Memory size in bytes, returns size = -1 when not found.

Parameters

- `[in] name`: variable name
- `[in] mem_path`: path where the variable is stored
- `[out] size`: size of the variable in bytes

Function memorymanagermodule::get_mem_rank

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine, public memorymanagermodule::get_mem_rank (name, mem_path, rank)
@ brief Get the variable rank

Returns rank = -1 when not found.

Parameters

- `[in] name`: variable name

Function memorymanagermodule::get_mem_shape

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine, public memorymanagermodule::get_mem_shape (name, mem_path, mem_shape)
@ brief Get the variable memory shape

Returns an integer array with the shape (Fortran ordering), and set shape(1) = -1 when not found.

Parameters

- `[in] name`: variable name
- `[in] mem_path`: path where the variable is stored
- `[out] mem_shape`: shape of the variable

Function memorymanagermodule::get_mem_type

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine, public memorymanagermodule::get_mem_type (name, mem_path, var_type)
@ brief Get the variable memory type

Returns any of 'LOGICAL', 'INTEGER', 'DOUBLE', 'STRING'. returns 'UNKNOWN' when the variable is not found.

Parameters

- `[in] name`: variable name
- `[in] mem_path`: path where the variable is stored
- `[out] var_type`: memory type

Function memorymanagermodule::mem_cleanup_table

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule::`**`mem_cleanup_table`**`()`
Generic function to clean a memory manager table.

Function memorymanagermodule::mem_da

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine, public memorymanagermodule::mem_da ()
Deallocate memory in the memory manager.

Function memorymanagermodule::mem_detailed_table

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine memorymanagermodule**::mem_detailed_table** (iout, nrows)
Create a table if memory_print_option is 'ALL'.

Parameters

- [in] iout: unit number for mfsim.lst
- [in] nrows: number of table rows

Function memorymanagermodule::mem_set_print_option

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine, public memorymanagermodule::mem_set_print_option (iout, keyword, error_msg)
Set the memory print option.

Parameters

- [in] iout: unit number for mfsim.lst
- [in] keyword: memory print option
- [inout] error_msg: returned error message if keyword is not valid option

Function memorymanagermodule::mem_summary_line

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine memorymanagermodule**::mem_summary_line** (component, rchars, rlog, rint, rreal, bytes)
Write a row for the memory_print_option 'SUMMARY' table.

Parameters

- [in] component: character defining the program component (e.g. solution)
- [in] rchars: allocated size of characters (in common units)
- [in] rlog: allocated size of logical (in common units)
- [in] rint: allocated size of integer variables (in common units)
- [in] rreal: allocated size of real variables (in common units)
- [in] bytes: total allocated memory in memory manager (in common units)

Function memorymanagermodule::mem_summary_table

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule`**`::mem_summary_table`**(iout, nrows, cunits)

Create a table if memory_print_option is 'SUMMARY'.

Parameters

- `[in] iout`: unit number for mfsim.lst
- `[in] nrows`: number of table rows
- `[in] cunits`: memory units (bytes, kilobytes, megabytes, or gigabytes)

Function memorymanagermodule::mem_summary_total

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule`**`::mem_summary_total`**(iout, bytes)

Create and fill a table with the total allocated memory.

Parameters

- `[in] iout`: unit number for mfsim.lst
- `[in] bytes`: total number of bytes allocated in the memory manager

Function memorymanagermodule::mem_unique_origins

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule`**`::mem_unique_origins`**(cunique)

Create a array with unique first components from all memory paths. Only the first component of the memory path is evaluated.

Parameters

- `[inout] cunique`: array with unique first components

Function memorymanagermodule::mem_units

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule`**`::mem_units`**(bytes, fact, cunits)

Determine appropriate memory unit and conversion factor.

Parameters

- `[in] bytes`: total nr. of bytes
- `[inout] fact`: conversion factor
- `[inout] cunits`: string with memory unit

Function memorymanagermodule::mem_write_usage

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

`subroutine, public memorymanagermodule::mem_write_usage (iout)`

Write memory manager memory usage based on the user-specified memory_print_option.

The total memory usage by data types (int, real, etc.) is written for every simulation.

Parameters

- `[in] iout`: unit number for mfsim.lst

Function memorymanagermodule::reallocate_dbl1d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule`**`::reallocate_dbl1d`**(adbl, nrow, name, mem_path)

Reallocate a 1-dimensional real array.

Parameters

- `[inout] adbl`: the reallocated 1d real array
- `[in] nrow`: number of rows
- `[in] name`: variable name
- `[in] mem_path`: path where variable is stored

Function memorymanagermodule::reallocate_dbl2d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule::`**`reallocate_dbl2d`**(adbl, ncol, nrow, name, mem_path)
Reallocate a 2-dimensional real array.

Parameters

- `[inout] adbl`: the reallocated 2d real array
- `[in] ncol`: number of columns
- `[in] nrow`: number of rows
- `[in] name`: variable name
- `[in] mem_path`: path where variable is stored

Function memorymanagermodule::reallocate_int1d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule::`**`reallocate_int1d`**(aint, nrow, name, mem_path)
Reallocate a 1-dimensional integer array.

Parameters

- `[inout] aint`: the reallocated integer array
- `[in] nrow`: number of rows
- `[in] name`: variable name
- `[in] mem_path`: path where variable is stored

Function memorymanagermodule::reallocate_int2d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule::`**`reallocate_int2d`**(aint, ncol, nrow, name, mem_path)
Reallocate a 2-dimensional integer array.

Parameters

- `[inout] aint`: the reallocated 2d integer array
- `[in] ncol`: number of columns
- `[in] nrow`: number of rows

- [in] name: variable name
- [in] mem_path: path where variable is stored

Function memorymanagermodule::reallocate_str1d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine memorymanagermodule::**reallocate_str1d**(astr, ilen, nrow, name, mem_path)
Reallocate a 1-dimensional defined length string array.

Parameters

- [in] ilen: string length
- [in] nrow: number of rows
- [inout] astr: the reallocated string array
- [in] name: variable name
- [in] mem_path: path where variable is stored

Function memorymanagermodule::reassignptr_dbl1d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine memorymanagermodule::**reassignptr_dbl1d**(adbl, name, mem_path, name_target, mem_path_target)
Set the pointer for a 1-dimensional real array to.

Parameters

- [inout] adbl: pointer to 1d real array
- [in] name: variable name
- [in] mem_path: path where variable is stored
- [in] name_target: name of target variable
- [in] mem_path_target: path where target variable is stored

Function memorymanagermodule::reassignptr_dbl2d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule::`**`reassignptr_dbl2d`**(adbl, name, mem_path, name_target, mem_path_target)

Set the pointer for a 2-dimensional real array to.

Parameters

- `[inout] adbl`: pointer to 2d real array
- `[in] name`: variable name
- `[in] mem_path`: path where variable is stored
- `[in] name_target`: name of target variable
- `[in] mem_path_target`: path where target variable is stored

Function memorymanagermodule::reassignptr_int1d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule::`**`reassignptr_int1d`**(aint, name, mem_path, name_target, mem_path_target)

Set the pointer for a 1-dimensional integer array to.

Parameters

- `[inout] aint`: pointer to 1d integer array
- `[in] name`: variable name
- `[in] mem_path`: path where variable is stored
- `[in] name_target`: name of target variable
- `[in] mem_path_target`: path where target variable is stored

Function memorymanagermodule::reassignptr_int2d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine memorymanagermodule::**reassignptr_int2d**(aint, name, mem_path, name_target, mem_path_target)

Set the pointer for a 2-dimensional integer array to.

Parameters

- [inout] aint: pointer to 2d integer array
- [in] name: variable name
- [in] mem_path: path where variable is stored
- [in] name_target: name of target variable
- [in] mem_path_target: path where target variable is stored

Function memorymanagermodule::setptr_dbl

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine memorymanagermodule::**setptr_dbl**(sclr, name, mem_path)

Set pointer to a real scalar.

Parameters

- [inout] sclr: pointer to a real scalar
- [in] name: variable name
- [in] mem_path: path where variable is stored

Function memorymanagermodule::setptr_dbl1d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine memorymanagermodule::**setptr_dbl1d**(adbl, name, mem_path)

Set pointer to a 1d real array.

Parameters

- [inout] adbl: pointer to 1d real array
- [in] name: variable name
- [in] mem_path: path where variable is stored

Function memorymanagermodule::setptr_dbl2d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule::`**`setptr_dbl2d`**(adbl, name, mem_path)
Set pointer to a 2d real array.

Parameters

- `[inout] adbl`: pointer to 2d real array
- `[in] name`: variable name
- `[in] mem_path`: path where variable is stored

Function memorymanagermodule::setptr_int

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule::`**`setptr_int`**(sclr, name, mem_path)
Set pointer to integer scalar.

Parameters

- `[inout] sclr`: pointer to integer scalar
- `[in] name`: variable name
- `[in] mem_path`: path where variable is stored

Function memorymanagermodule::setptr_int1d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule::`**`setptr_int1d`**(aint, name, mem_path)
Set pointer to 1d integer array.

Parameters

- `[inout] aint`: pointer to 1d integer array
- `[in] name`: variable name
- `[in] mem_path`: path where variable is stored

Function memorymanagermodule::setptr_int2d

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule::`**`setptr_int2d`**(aint, name, mem_path)
Set pointer to 2d integer array.

Parameters

- `[inout] aint`: pointer to 2d integer array
- `[in] name`: variable name
- `[in] mem_path`: path where variable is stored

Function memorymanagermodule::setptr_logical

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Function Documentation

subroutine `memorymanagermodule::`**`setptr_logical`**(sclr, name, mem_path)
Set pointer to a logical scalar.

Parameters

- `[inout] sclr`: pointer to logical scalar
- `[in] name`: variable name
- `[in] mem_path`: path where variable is stored

Function memorysethandlermodule::mem_register_handler

- Defined in file_src_Utilities_Memory_MemorySetHandler.f90

Function Documentation

subroutine, public memorysethandlermodule::mem_register_handler (var_name, mem_path, handle
Register the event handler and context for this variable.

The event handler and its ctx are called whenever the trigger is given by calling `on_set_memory()`. This allows to handle side effects, e.g. when a variable is from outside a class (the context) such as happens with the BMI.

Parameters

- `[in] var_name`: the variable name
- `[in] mem_path`: the memory path
- `handler`: called after memory is set

- `ctx`: the context with which the handler should be called

Function memorysethandlermodule::on_memory_set

- Defined in file_src_Utilities_Memory_MemorySetHandler.f90

Function Documentation

subroutine, public memorysethandlermodule::on_memory_set (var_name, mem_path, status)

Triggers the calling of the side effect handler for this variable.

The handler can be set by calling `mem_register_handler()`. When the status contains an error code, the program should be stopped because the data in memory is no longer consistent...

Parameters

- `[in] var_name`: the variable name
- `[in] mem_path`: the memory path
- `[out] status`: status: 0 for success, -1 when failed

Function memorytypemodule::mt_associated

- Defined in file_src_Utilities_Memory_Memory.f90

Function Documentation

logical function memorytypemodule::mt_associated (this)

Function memorytypemodule::table_entry

- Defined in file_src_Utilities_Memory_Memory.f90

Function Documentation

subroutine memorytypemodule::table_entry (this, memtab)

Function mf6bmi::bmi_finalize

- Defined in file_srcbmi_mf6bmi.f90

Function Documentation

integer(kind=c_int) function mf6bmi::bmi_finalize ()

Clean up the initialized simulation.

Performs teardown tasks for the initialized simulation, this call should match the call to *bmi_initialize()*

Return BMI status code

Function mf6bmi::bmi_initialize

- Defined in file_srcbmi_mf6bmi.f90

Function Documentation

integer(kind=c_int) function mf6bmi::bmi_initialize ()

Initialize the computational core.

It is required to have the MODFLOW 6 configuration file 'mfsim.nam' available in the current working directory when calling this routine.

NOTE: initialization should always be matched with a call to *bmi_finalize()*. However, we currently do not support the reinitialization of a model in the same memory space... You would have to create a new process for that.

Return BMI status code

Function mf6bmi::bmi_update

- Defined in file_srcbmi_mf6bmi.f90

Function Documentation

integer(kind=c_int) function mf6bmi::bmi_update ()

Perform a computational time step.

It will prepare the timestep, call the calculation routine on all the solution groups in the simulation, and finalize the timestep by printing out diagnostics and writing output. It can be called in succession to perform multiple steps up to the simulation's end time is reached.

Return BMI status code

Function mf6bmi::get_current_time

- Defined in file_srcbmi_mf6bmi.f90

Function Documentation

integer(kind=c_int) function mf6bmi::get_current_time (current_time)
Get the current time of the simulation.

As MODFLOW currently does not have internal time, this will be equal to the time passed w.r.t. the start time of the simulation.

Return BMI status code

Parameters

- `[out] current_time`: current time

Function mf6bmi::get_end_time

- Defined in file_srcbmi_mf6bmi.f90

Function Documentation

integer(kind=c_int) function mf6bmi::get_end_time (end_time)
Get the end time of the simulation.

As MODFLOW does currently does not have internal time, this will be equal to the total runtime.

Return BMI status code

Parameters

- `[out] end_time`: end time

Function mf6bmi::get_input_item_count

- Defined in file_srcbmi_mf6bmi.f90

Function Documentation

integer(kind=c_int) function mf6bmi::get_input_item_count (count)
Get the number of input variables in the simulation.

This concerns all those variables which have their access specifier set to "readwrite" in the internal memory manager

Return BMI status code

Parameters

- `[out] count`: the number of input variables

Function mf6bmi::get_input_var_names

- Defined in file_srcbmi_mf6bmi.f90

Function Documentation

integer(kind=c_int) function mf6bmi::get_input_var_names (c_names)

Returns all input variables in the simulation.

This functions returns the full address for all variables in the memory manager, that have their access specified as "readwrite" and can therefore be used as input variables. The array `c_names` should be pre-allocated of proper size:

size = BMI_LENVARADDRESS * *get_input_item_count()*

The strings will be written contiguously with stride equal to BMI_LENVARADDRESS and nul-terminated where the trailing spaces start:

c_names = 'variable_address_1 ... variable_address_2 ... ' etc.

Return BMI status code

Parameters

- `[inout] c_names`: array with memory paths for input variables

Function mf6bmi::get_output_item_count

- Defined in file_srcbmi_mf6bmi.f90

Function Documentation

integer(kind=c_int) function mf6bmi::get_output_item_count (count)

Get the number of output variables in the simulation.

This concerns all those variables which have their access specifier set to "readonly|readwrite" in the internal memory manager

Return BMI status code

Parameters

- `[out] count`: the number of output variables

Function mf6bmi::get_output_var_names

- Defined in file_srcbmi_mf6bmi.f90

Function Documentation

integer(kind=c_int) function mf6bmi::get_output_var_names (c_names)
Returns all output variables in the simulation.

This function works analogously to *get_input_var_names()*, for all variables that have their access specified as either "readonly" or "readwrite"

Return BMI status code

Parameters

- `[inout] c_names`: array with memory paths for output variables

Function mf6bmi::get_start_time

- Defined in file_srcbmi_mf6bmi.f90

Function Documentation

integer(kind=c_int) function mf6bmi::get_start_time (start_time)
Get the start time of the simulation.

As MODFLOW currently does not have internal time, this will be returning 0.0 for now. New version...

Return BMI status code

Parameters

- `[out] start_time`: start time

Function mf6bmi::get_time_step

- Defined in file_srcbmi_mf6bmi.f90

Function Documentation

integer(kind=c_int) function mf6bmi::get_time_step (time_step)
Get the time step for the simulation.

Note that the returned value may vary between and within stress periods, depending on your time discretization settings in the TDIS package.

Return BMI status code

Parameters

- `[out] time_step`: current time step

Function mf6bmi::get_value_double

- Defined in file_srcbmi_mf6bmi.f90

Function Documentation

integer function mf6bmi::get_value_double (c_var_address, c_arr_ptr)

Copy the double precision values of a variable into the array.

The copied variable us located at `c_var_address`. The caller should provide `c_arr_ptr` pointing to an array of the proper shape (the BMI function *get_var_shape()* can be used to create it). Multi-dimensional arrays are supported.

Return BMI status code

Parameters

- `[in] c_var_address`: memory address string of the variable
- `[in] c_arr_ptr`: pointer to the double precision array

Function mf6bmi::get_value_int

- Defined in file_srcbmi_mf6bmi.f90

Function Documentation

integer function mf6bmi::get_value_int (c_var_address, c_arr_ptr)

Copy the integer values of a variable into the array.

The copied variable us located at `c_var_address`. The caller should provide `c_arr_ptr` pointing to an array of the proper shape (the BMI function *get_var_shape()* can be used to create it). Multi-dimensional arrays are supported.

Return BMI status code

Parameters

- `[in] c_var_address`: memory address string of the variable
- `[in] c_arr_ptr`: pointer to the double precision array

Function mf6bmi::get_value_ptr_double

- Defined in file_srcbmi_mf6bmi.f90

Function Documentation

integer(kind=c_int) function mf6bmi::get_value_ptr_double (c_var_address, c_arr_ptr)

Get a pointer to the array of double precision numbers.

The array is located at `c_var_address`. There is no copying of data involved. Multi-dimensional arrays are supported and the *get_var_rank()* function can be used to get the variable's dimensionality, and *get_var_shape()* for its shape.

Return BMI status code

Parameters

- `[in] c_var_address`: memory address string of the variable
- `[inout] c_arr_ptr`: pointer to the array

Function mf6bmi::get_value_ptr_int

- Defined in file_srcbmi_mf6bmi.f90

Function Documentation

integer(kind=c_int) function mf6bmi::get_value_ptr_int (c_var_address, c_arr_ptr)

Get a pointer to the array of integer numbers.

The array is located at `c_var_address`. There is no copying of data involved. Multi-dimensional arrays are supported and the *get_var_rank()* function can be used to get the variable's dimensionality.

Return BMI status code

Parameters

- `[in] c_var_address`: memory address string of the variable
- `[inout] c_arr_ptr`: pointer to the array

Function mf6bmi::get_var_itemsize

- Defined in file_srcbmi_mf6bmi.f90

Function Documentation

integer(kind=c_int) function mf6bmi::get_var_itemsize (c_var_address, var_size)

Get the size (in bytes) of a single element of a variable.

Return BMI status code

Parameters

- `[in] c_var_address`: memory address string of the variable
- `[out] var_size`: size of the element in bytes

Function mf6bmi::get_var_nbytes

- Defined in file_srcbmi_mf6bmi.f90

Function Documentation

integer(kind=c_int) function mf6bmi::get_var_nbytes (c_var_address, var_nbytes)
Get size of the variable, in bytes.

Return BMI status code

Parameters

- `[in] c_var_address`: memory address string of the variable
- `[out] var_nbytes`: size in bytes

Function mf6bmi::get_var_rank

- Defined in file_srcbmi_mf6bmi.f90

Function Documentation

integer(kind=c_int) function mf6bmi::get_var_rank (c_var_address, c_var_rank)
Get the variable rank (non-BMI)

In order to support multi-dimensional arrays, this function gives access to the rank of the array.

Return BMI status code

Parameters

- `[in] c_var_address`: memory address string of the variable
- `[out] c_var_rank`: variable rank

Function mf6bmi::get_var_shape

- Defined in file_srcbmi_mf6bmi.f90

Function Documentation

integer(kind=c_int) function mf6bmi::get_var_shape (c_var_address, c_var_shape)
Get the shape of the array for the variable (non-BMI)

The shape is an integer array with size equal to the rank of the variable (see *get_var_rank()*) and values that give the length of the array in each dimension. The target shape array `c_var_shape` should be allocated before calling this routine.

Note that the returned shape representation will has been converted to C-style.

Return BMI status code

Parameters

- `[in] c_var_address`: memory address string of the variable
- `[inout] c_var_shape`: 1D array with the variable's shape

Function mf6bmi::get_var_type

- Defined in file_srcbmi_mf6bmi.f90

Function Documentation

integer(kind=c_int) function mf6bmi::get_var_type (c_var_address, c_var_type)

Get the variable type as a string.

The type returned is that of a single element. Currently we support 'INTEGER' and 'DOUBLE'. When the variable cannot be found, the string 'UNKNOWN' is assigned.

Return BMI status code

Parameters

- `[in] c_var_address`: memory address string of the variable
- `[out] c_var_type`: variable type as a string

Function mf6bmi::set_value_double

- Defined in file_srcbmi_mf6bmi.f90

Function Documentation

integer function mf6bmi::set_value_double (c_var_address, c_arr_ptr)

Set new values for a variable of type double.

The array pointed to by `c_arr_ptr` can have rank equal to 0, 1, or 2 and should have a C-style layout, which is particularly important for rank > 1. When the memory access in the manager for the variable `c_var_address` is specified as `MEMHIDDEN` or `MEMREADONLY`, the function will return with a BMI error code.

Return BMI status code

Parameters

- `[in] c_var_address`: memory address string of the variable
- `[in] c_arr_ptr`: pointer to the double precision array

Function mf6bmi::set_value_int

- Defined in file_srcbmi_mf6bmi.f90

Function Documentation

integer function mf6bmi::set_value_int (c_var_address, c_arr_ptr)

Set new values for a variable of type integer.

The array pointed to by `c_arr_ptr` can have rank equal to 0, 1, or 2. When the memory access in the manager for the variable `c_var_address` is specified as `MEMHIDDEN` or `MEMREADONLY`, the function will return with a BMI error code.

Return BMI status code

Parameters

- `[in] c_var_address`: memory address string of the variable
- `[in] c_arr_ptr`: pointer to the integer array

Function mf6bmigrid::get_grid_face_count

- Defined in file_srcbmi_mf6bmiGrid.f90

Function Documentation

integer(kind=c_int) function mf6bmigrid::get_grid_face_count (grid_id, count)

Function mf6bmigrid::get_grid_face_nodes

- Defined in file_srcbmi_mf6bmiGrid.f90

Function Documentation

integer(kind=c_int) function mf6bmigrid::get_grid_face_nodes (grid_id, face_nodes)

Function mf6bmigrid::get_grid_node_count

- Defined in file_srcbmi_mf6bmiGrid.f90

Function Documentation

```
integer(kind=c_int) function mf6bmigrid::get_grid_node_count (grid_id, count)
```

Function mf6bmigrid::get_grid_nodes_per_face

- Defined in file_srcbmi_mf6bmiGrid.f90

Function Documentation

```
integer(kind=c_int) function mf6bmigrid::get_grid_nodes_per_face (grid_id, nodes_per_face)
```

Function mf6bmigrid::get_grid_rank

- Defined in file_srcbmi_mf6bmiGrid.f90

Function Documentation

```
integer(kind=c_int) function mf6bmigrid::get_grid_rank (grid_id, grid_rank)
```

Function mf6bmigrid::get_grid_shape

- Defined in file_srcbmi_mf6bmiGrid.f90

Function Documentation

```
integer(kind=c_int) function mf6bmigrid::get_grid_shape (grid_id, grid_shape)
```

Function mf6bmigrid::get_grid_size

- Defined in file_srcbmi_mf6bmiGrid.f90

Function Documentation

```
integer(kind=c_int) function mf6bmigrid::get_grid_size (grid_id, grid_size)
```

Function mf6bmigrid::get_grid_type

- Defined in file_srcbmi_mf6bmiGrid.f90

Function Documentation

integer(kind=c_int) function mf6bmigrid::get_grid_type (grid_id, grid_type)

Function mf6bmigrid::get_grid_x

- Defined in file_srcbmi_mf6bmiGrid.f90

Function Documentation

integer(kind=c_int) function mf6bmigrid::get_grid_x (grid_id, grid_x)

Function mf6bmigrid::get_grid_y

- Defined in file_srcbmi_mf6bmiGrid.f90

Function Documentation

integer(kind=c_int) function mf6bmigrid::get_grid_y (grid_id, grid_y)

Function mf6bmigrid::get_var_grid

- Defined in file_srcbmi_mf6bmiGrid.f90

Function Documentation

integer(kind=c_int) function mf6bmigrid::get_var_grid (c_var_address, var_grid)

Function mf6bmiutil::char_array_to_string

- Defined in file_srcbmi_mf6bmiUtil.f90

Function Documentation

pure character(len=length) function mf6bmiutil::char_array_to_string (char_array, length)

Convert C-style string to Fortran character string.

Return Fortran fixed length character string

Parameters

- `[in] length`: string length without terminating null character
- `[in] char_array`: string to convert

Function mf6bmiutil::confirm_grid_type

- Defined in file_srcbmi_mf6bmiUtil.f90

Function Documentation

logical function mf6bmiutil::confirm_grid_type (grid_id, expected_type)
Confirm that grid is of an expected type.

Function mf6bmiutil::extract_model_name

- Defined in file_srcbmi_mf6bmiUtil.f90

Function Documentation

character(len=lenmodelname) function mf6bmiutil::extract_model_name (var_address)
Extract the model name from a memory address string.

Return the extracted model name

Parameters

- `[in] var_address`: the memory address for the variable

Function mf6bmiutil::get_grid_type_model

- Defined in file_srcbmi_mf6bmiUtil.f90

Function Documentation

subroutine `mf6bmiutil::`**get_grid_type_model** (model_name, grid_type_f)
Get the grid type for a named model as a fortran string.

Function mf6bmiutil::get_memory_access_type

- Defined in file_srcbmi_mf6bmiUtil.f90

Function Documentation

integer(i4b) function mf6bmiutil::get_memory_access_type (mem_path, var_name, found)
Check memory access type.

Return access type of memory in manager

Parameters

- `[in] mem_path`: memory path used by the memory manager
- `[in] var_name`: name of the variable
- `[out] found`: false, if entry does not exist

Function mf6bmiutil::get_model_name

- Defined in file_srcbmi_mf6bmiUtil.f90

Function Documentation

character(len=lenmodelname) function mf6bmiutil::get_model_name (grid_id)
Get the model name from the grid id.

Return model name

Parameters

- `[in] grid_id`: grid id

Function mf6bmiutil::getsolution

- Defined in file_srcbmi_mf6bmiUtil.f90

Function Documentation

class(numericalsolutiontype) function, pointer mf6bmiutil::getsolution (subcomponent_idx)
Get the solution object for this index.

Return Numerical Solution

Parameters

- `[in] subcomponent_idx`: index of solution

Function mf6bmiutil::split_address

- Defined in file_srcbmi_mf6bmiUtil.f90

Function Documentation

subroutine `mf6bmiutil::split_address(c_var_address, mem_path, var_name)`
Split the variable address string.

Splits the full address string into a memory path and variable name, following the rules used by the memory manager.

Parameters

- `[in] c_var_address`: full address of a variable
- `[out] mem_path`: memory path used by the memory manager
- `[out] var_name`: name of the variable

Function mf6bmiutil::string_to_char_array

- Defined in file_srcbmi_mf6bmiUtil.f90

Function Documentation

pure character(kind=c_char, len=1) function, dimension(length+1) mf6bmiutil::string_to_char
Convert Fortran string to C-style character string.

Return C-style character string

Parameters

- `[in] length`: Fortran string length
- `[in] string`: string to convert

Function mf6bmiutil::strlen

- Defined in file_srcbmi_mf6bmiUtil.f90

Function Documentation

pure integer(i4b) function mf6bmiutil::strlen (char_array)
Returns the string length without the trailing null character.

Return Fortran string length

Parameters

- `[in] char_array`: C-style character string

Function mf6xmi::get_var_address

- Defined in file_srcbmi_mf6xmi.f90

Function Documentation

integer(kind=c_int) function mf6xmi::get_var_address (c_component_name, c_subcomponent_name
Get the full address string for a variable.

This routine constructs the full address string of a variable using the exact same logic as the internal memory manager. This routine should always be used when accessing a variable through the BMI to assure compatibility with future versions of the library.

Return BMI status code

Parameters

- `[in] c_component_name`: name of the component (a Model or Solution)
- `[in] c_subcomponent_name`: name of the subcomponent (Package), or an empty string" when not applicable
- `[in] c_var_name`: name of the variable

- `[out] c_var_address`: full address of the variable

Function mf6xmi::xmi_do_time_step

- Defined in file_srcbmi_mf6xmi.f90

Function Documentation

integer(kind=c_int) function mf6xmi::xmi_do_time_step ()

Perform a single time step.

It does so by looping over all solution groups, and calling the calculate function on all solutions in there.

Return BMI status code

Function mf6xmi::xmi_finalize_solve

- Defined in file_srcbmi_mf6xmi.f90

Function Documentation

integer(kind=c_int) function mf6xmi::xmi_finalize_solve (subcomponent_idx)

Finalize the solve of the system.

This will determine convergence, reports, calculate flows and budgets, and more… It should always follow after a call to *xmi_prepare_solve()* and *xmi_solve()*.

Return bmi_status the BMI status code

Return BMI status code

Parameters

- `[in] subcomponent_idx`: the index of the subcomponent (Numerical Solution)

Parameters

- `[in] subcomponent_idx`: index of the subcomponent (i.e. Numerical Solution)

Function mf6xmi::xmi_finalize_time_step

- Defined in file_srcbmi_mf6xmi.f90

Function Documentation

integer(kind=c_int) function mf6xmi::xmi_finalize_time_step ()

Finalize the time step.

This will mostly write output and messages. It is essential to call this to finish the time step.

Return BMI status code

Function mf6xmi::xmi_get_subcomponent_count

- Defined in file_srcbmi_mf6xmi.f90

Function Documentation

integer(kind=c_int) function mf6xmi::xmi_get_subcomponent_count (count)

This will get the number of Numerical Solutions in the simulation.

For most applications, this number will be equal to 1. Note that this part of the XMI only works when the simulation is defined with a single Solution Group. (If you don't know what a Solution Group is, then you are most likely not using more than one...)

Return BMI status code

Parameters

- `[out] count`: number of solutions

Function mf6xmi::xmi_prepare_solve

- Defined in file_srcbmi_mf6xmi.f90

Function Documentation

integer(kind=c_int) function mf6xmi::xmi_prepare_solve (subcomponent_idx)

Prepare for solving the system.

This preparation mostly consists of advancing the solutions, models, and exchanges in the simulation. The index `subcomponent_idx` runs from 1 to the value returned by *xmi_get_subcomponent_count()*.

Return BMI status code

Parameters

- `subcomponent_idx`: index of the subcomponent (i.e. Numerical Solution)

Function mf6xmi::xmi_prepare_time_step

- Defined in file_srcbmi_mf6xmi.f90

Function Documentation

integer(kind=c_int) function mf6xmi::xmi_prepare_time_step (dt)

Prepare a single time step.

The routine takes the time step `dt` as an argument. However, MODFLOW (currently) does not allow to alter this value after initialization, so it is ignored here.

Return BMI status code

Parameters

- `[in] dt`: time step

Function mf6xmi::xmi_solve

- Defined in file_srcbmi_mf6xmi.f90

Function Documentation

integer(kind=c_int) function mf6xmi::xmi_solve (subcomponent_idx, has_converged)

Build and solve the linear system.

The solve is called on the Numerical Solution indicated by the value of `subcomponent_idx`, which runs from 1 to the value returned by *xmi_get_subcomponent_count()*. Before calling this, a matching call to *xmi_prepare_solve()* should be done.

Return BMI status code

Parameters

- `[in] subcomponent_idx`: index of the subcomponent (i.e. Numerical Solution)
- `[out] has_converged`: equal to 1 for convergence, 0 otherwise

2.3.3 Variables

Variable bmif::bmi_failure

- Defined in file_srcbmi_bmi.f90

Variable Documentation

integer, parameter bmif::bmi_failure = 1

Variable bmif::bmi_max_component_name

- Defined in file_srcbmi_bmi.f90

Variable Documentation

integer, parameter bmif::bmi_max_component_name = 2048

Variable bmif::bmi_max_type_name

- Defined in file_srcbmi_bmi.f90

Variable Documentation

```
integer, parameter bmif::bmi_max_type_name   = 2048
```

Variable bmif::bmi_max_units_name

- Defined in file_srcbmi_bmi.f90

Variable Documentation

```
integer, parameter bmif::bmi_max_units_name   = 2048
```

Variable bmif::bmi_max_var_name

- Defined in file_srcbmi_bmi.f90

Variable Documentation

```
integer, parameter bmif::bmi_max_var_name   = 2048
```

Variable bmif::bmi_success

- Defined in file_srcbmi_bmi.f90

Variable Documentation

```
integer, parameter bmif::bmi_success   = 0
```

Variable memoryhelpermodule::mempathseparator

- Defined in file_src_Utilities_Memory_MemoryHelper.f90

Variable Documentation

```
character(len=lenmemseparator), parameter memoryhelpermodule::mempathseparator   = '/'
```
used to build up the memory address for the stored variables

Variable memorymanagermodule::iprmem

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Variable Documentation

```
integer(i4b) memorymanagermodule::iprmem   = 0
```

Variable memorymanagermodule::memorylist

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Variable Documentation

```
type(memorylisttype), public memorymanagermodule::memorylist
```

Variable memorymanagermodule::memtab

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Variable Documentation

```
type(tabletype), pointer memorymanagermodule::memtab   => null()
```

Variable memorymanagermodule::nvalues_achr

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Variable Documentation

```
integer(i8b) memorymanagermodule::nvalues_achr   = 0
```

Variable memorymanagermodule::nvalues_adbl

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Variable Documentation

```
integer(i8b) memorymanagermodule::nvalues_adbl   = 0
```

Variable memorymanagermodule::nvalues_aint

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Variable Documentation

```
integer(i8b) memorymanagermodule::nvalues_aint   = 0
```

Variable memorymanagermodule::nvalues_alogical

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Variable Documentation

```
integer(i8b) memorymanagermodule::nvalues_alogical   = 0
```

Variable memorymanagermodule::nvalues_astr

- Defined in file_src_Utilities_Memory_MemoryManager.f90

Variable Documentation

```
integer(i8b) memorymanagermodule::nvalues_astr   = 0
```

Variable memorysethandlermodule::handler_list

- Defined in file_src_Utilities_Memory_MemorySetHandler.f90

Variable Documentation

```
type(listtype) memorysethandlermodule::handler_list
```

Variable mf6bmi::bind

- Defined in file_srcbmi_mf6bmi.f90

Variable Documentation

```
integer(c_int), dimension(c, name="istdouttofile") mf6bmi::bind
```

Variable mf6bmi::istdout_to_file

- Defined in file_srcbmi_mf6bmi.f90

Variable Documentation

integer(c_int) mf6bmi::istdout_to_file = 1
output control: =0 to screen, >0 to file

Variable mf6bmiutil::bind

- Defined in file_srcbmi_mf6bmiUtil.f90

Variable Documentation

integer(c_int), dimension(c, name="bmi_lenvaraddress") mf6bmiutil::bind

Variable mf6bmiutil::bmi_lengridtype

- Defined in file_srcbmi_mf6bmiUtil.f90

Variable Documentation

integer(c_int) mf6bmiutil::bmi_lengridtype = LENGRIDTYPE + 1
max. length for grid type C-strings

Variable mf6bmiutil::bmi_lenvaraddress

- Defined in file_srcbmi_mf6bmiUtil.f90

Variable Documentation

integer(c_int) mf6bmiutil::bmi_lenvaraddress = LENMEMADDRESS + 1
max. length for the variable's address C-string

Variable mf6bmiutil::bmi_lenvartype

- Defined in file_srcbmi_mf6bmiUtil.f90

Variable Documentation

integer(c_int) mf6bmiutil::bmi_lenvartype = LENMEMTYPE + 1
max. length for variable type C-strings

Variable mf6bmiutil::lengridtype

- Defined in file_srcbmi_mf6bmiUtil.f90

Variable Documentation

integer(i4b), parameter mf6bmiutil::lengridtype = 16
max length for Fortran grid type string

Variable mf6xmi::iterationcounter

- Defined in file_srcbmi_mf6xmi.f90

Variable Documentation

integer(i4b), pointer mf6xmi::iterationcounter => null()
the counter for the outer iteration loop, initialized in xmi_prepare_iteration()

INDEX

M