

FEFLOW File Formats

1 FEFLOW Supermesh File Format (*.smh) - Version 5.4

1.1 General remarks

The supermesh file has the following structure:

Part of the file	Contents
Header	General geometry description including graphics window extension and global coordinate offset
Geometry definition	Geometry definition of the polygonal superelements and of the line and point Add-ins
Type definition	Definition of <ul style="list-style-type: none">- the type of the superelement (polygon or line Add-in or point Add-in),- of the refinement level for the superelement and- of the proposed element number for polygonal superelements realized in FEFLOW

1.1.1 What are supermeshes, superelements, Add-ins ?

A supermesh is a collection of polygons, lines and points in the 2D plane defined by the problem orientation of a finite element problem. In this description the coordinates are called X and Y for the horizontal and vertical direction in FEFLOW graphics window. A supermesh will be used to define the modelling region and serves as input data for the finite element mesh generation. For quadrangular finite elements meshes also the supermesh polygons may be only quadrangles. For triangular finite element meshes there are no limits regarding the geometry of the supermesh polygons. (NOTE: The design of the supermesh influences numerical stability of the mesh generator.)

A supermesh consists of one or more superelements. A superelement can be a polygonal superelement or a line add in or a point add in. A supermesh must contain at least one polygonal superelement to enable the finite element mesh generation.

Supermeshes without polygonal superelements can be stored and used for other purposes than mesh generation (e.g. JOIN method in the problem editor).

For finite element simulations the supermesh can consist of more than one „island“ of polygons. Holes and fractures in the supermesh can be defined by surrounding the hole or fracture with polygonal superelements and NO polygonal superelement in the hole or fracture.

In the FEFLOW mesh generator the polygonal superelements will be filled with finite elements. On the polygon border mesh nodes and finite element edges will be generated. The mesh generator TMESH can use line and point Add-ins to specify the coordinates of finite element mesh nodes inside the polygonal superelements.

Every node in a superelement will be defined by its X- and Y coordinates and by the node index. The connection of two polygonal superelements by common edges will be defined by identical node indices and identical node coordinates for the common nodes in the two polygons. NOTE: Identical coordinates but different node indices for a „common edge“ between two polygonal superelements produce two unconnected mesh regions in the finite element mesh generator.

Rules for polygonal superelements:

One polygonal superelement defines one polygon without any holes. The polygons are closed. The endpoint of the polyline (that is identical with the startpoint) will not be stored and counted. For every polygon border side a so called „midside node“ will be stored to define curved and stretched border lines. So in the geometry definition of a polygon the 1st, 3rd, 5th , ... node are polygon edge nodes and the 2nd, 4th, .. node are midside nodes. For polygonal superelements the number of nodes will always be

even.

Rules for line Add-ins:

One line add in defines a polyline that is not closed (startpoint and endpoint are not identical). The polyline consists of one or more straight lines between the line edge nodes. There are no midside nodes defined. The polyline has no ramification, but slopes are possible.

Rules for point Add-ins:

One point add in defines one or more point coordinates without any connections between. The point coordinates must not ly exactly on the edge node of a polygonal superelement or a line Add-in.

1.2 *.smh file format

1.2.1 The header

The header data are organized in 5 lines. Bold written words are contents of the file. Not cursive written words are keyword, cursive written words are variable data in the file. :

```

Supermeshdata:
window_width, pixelsize, y_to_x_exaggeration, shift_x, shift_y
date
GK_COOR x1GK, y1GK
numb_superelem, mesh_type
  
```

Description of the logical values:

Name	Format	Meaning
window width	%.6e	Horizontal window extent of FEFLOW graphics window (Xdirection) in real world coordinates [m]
pixelsize	%.6e	Unused. In former versions the horizontal extension of one pixel in FEFLOW graphics window.
y to x exaggeration	%.6e	The Y to X exaggeration for drawing in the working window. Usually only used for vertical and axisymmetric problems. Otherwise: 1.0 as default value.
shift x, shift y	%.6e	Local coordinate shift in X and Y direction (as defined in the problem measure menu using the „shift origin“ operation).
date	like UNIX date command	Creation date of the file. Will not be analyzed on file reaping process.
x1GK, y1GK	%.6e	Gauss Krueger coordinates of the lower left corner of the working window. Used to define the global coordinates position of the supermesh.
numb superelem	%d	Number of superelements (polygons plus Add-ins)
mesh type	%d	0 = triangular mesh 1 = quadrangular mesh

1.2.2 The geometry definition

The following lines describe the geometry definition of all superelements. Note that in the loop of elements the polygonal superelements will be defined before the Add-ins. :

```

for (elem = 0; elem < numb_superelem; elem++) {
  numb_nodes[elem]
  for (nn; nn < numb_nodes[elem]; nn++) {
    node_index[elem][nn], x_coor[elem][nn], y_coor[elem][nn]
  }
  x_min[elem], x_max[elem]
  y_min[elem], y_max[elem]
  for (side = 0; side < numb_nodes[elem] / 2; side++) {
    side_type[elem][side]
  }
}
END

```

Description of the logical values:

Name	Format	Meaning
elem		Index of a superelement. In the loop of elements the polygonal superelements will be followed by the Add-ins.
nn		Index of superelement nodes (edge and midside nodes).
side		Index of the polygon or polyline line piece. (It is also defined for point Add-ins but not used there.)
numb nodes [elem]	%d	Number of nodes of the superelement <i>elem</i> . (For polygonal superelements: The sum of edge and midside nodes. Start and endpoint are only one node.)
node index [elem] [nn]	%d	Node index of point <i>nn</i> in element <i>elem</i> . NOTE: Pay attention to the definition of identical node indices for connected superelements!
x coor [elem] [nn] y coor [elem] [nn]	%.6e	Local X- and Y-coordinates of the point <i>nn</i> in element <i>elem</i> .
x min [elem] x max [elem]	%.6e	Extremal X coordinates of the superelement <i>elem</i> . (Will be stored to increase performane of FEFLOW operations).
y min [elem] y max [elem]	%.6	Extremal Y coordinates of the superelement <i>elem</i> . (Will be stored to increase performane of FEFLOW operations).
side type [elem] [side]	%d	Type of polygonal border piece: If the midside node of borderpiece <i>side</i> will be outside the centerpoint of the line (arithmetic average of the two neighbored edge nodes) , than the border will be curved using one of the following side types: 0 = parabolic curved borderpiece (default value) 1 = circular curved borderpiece In case of noncurved border pieces or line add-in's the border piece will be a straight line. NOTE: The number of entries in this loop is the result of the integer quotient $\text{numb_nodes}[elem] / 2$. This formula will be applied also for line and point Add-ins.
END		This keyword finishes the geometry definition.

1.2.3 Type definition

The order of superelements in this type definition is the same as in the geometry definition loop. For every superelement finds one data line.

```
for (elem = 0; elem < numb_superelem; elem++) {
  elem_type[elem], refinement_level[elem], proposed_elem_number[elem]
}
```

Description of the logical values:

Name	Format	Meaning
elem		Index of a superelement. In the loop of elements the polygonal superelements will be followed by the Add-ins.
elem type[elem]	%d	type of the superelement <i>elem</i> : 1 = polygonal superelement 3 = line Add-in 4 = point Add-in
refinement level[elem]	%d	Class of mesh refinement around polygon border or line and point Add-ins (corresponding to „generator options“ menu): 0 = no refinement 1 = low refinement 2 = medium refinement 3 = high refinement 4 = user defined refinement (refinement value will not be stored)
proposed elem number[elem]	%d	For polygonal superelements this entry stores the proposed element numbers defined in the mesh generator using „areal refinement“. For line and point Add-ins this value is zero.

1.2.4 Edge type definition

For each polygonal superelement there is a data block containing information about the curve type of the edges.

```
EDGE_FLAG
for (i = 0; i < number_superelem; i++) {
  elem n_edges/2
  for (j = 0; j < n_edges/2; j++) {
    c_type[j]
  }
}
END
```

Name	Format	Meaning
n edges	%d	Number of edges of superelement <i>i</i>
elem type[elem]	%d	type of the superelement <i>elem</i> : 1 = polygonal superelement 3 = line Add-in 4 = point Add-in
c type	%d	Curve type for superelement edge: 0 = linear edge 1 = parabolic edge 2 = circular edge

1.3 *.smx file format

Additionally to the *.smh file a *.smx file is stored, if background maps are loaded. The *.smx file uses the syntax of the *.fem file to save the paths of the background maps and their style/color settings. See the corresponding chapters in the FEFLOW Model format description below.

1.4 A simple example for a *.smh file

The file contents will be written in the left column, comments are written with italic font in the right column. Blank lines in the file contents are caused only by the comments and will not occur in the original file.

Supermeshdata:

1.) Header

```
1.000000e+002,1.000000e-001,1.000000e+000,1.000000e+001,1.000000e+001      width of working
                                         window in meters,
                                         pixelsize (unused),
                                         Y/X exaggeration,
                                         local coordinate shift
                                         (x,y)
Thu Sep 25 11:13:28 1997                                                         date of creation
GK_COOR 0.000000, 0.000000      Gauss Krueger coordinates of lower left corner of the working window
3,0                               number of superelements, 0 = triangulation
```

2.) Geometry definition

2.1. A polygonal superelement

```
8                                     number of superelement nodes (inclusive midside nodes)
1,0.000000e+000,0.000000e+000      list of node coordinates of first polygon (ID,X,Y)
2,5.000000e+000,0.000000e+000
3,1.000000e+001,0.000000e+000
4,1.000000e+001,5.000000e+000
5,1.000000e+001,1.000000e+001
6,5.000000e+000,1.000000e+001
7,0.000000e+000,1.000000e+001
8,0.000000e+000,5.000000e+000
0.000000e+000,1.000000e+001      Xmin, Xmax
0.000000e+000,1.000000e+001      Ymin, Ymax
0                                     Border side type
0
0
0
0
```

2.2. A line Add-in

```
2                                     number of superelementnodes (NO midside nodes!)
1,0.000000e+000,0.000000e+000      list of node coordinates of first line (ID,X,Y)
5,1.000000e+001,1.000000e+001
0.000000e+000,1.000000e+001      Xmin, Xmax
0.000000e+000,1.000000e+001      Ymin, Ymax
0                                     Border side type
```

2.3. A point Add-in

```
( No Border side type because of 1/2 == 0)
1                                     number of superelement nodes
9,5.000000e+000,5.000000e+000      list of node coordinates of first point (ID,X,Y)
5.000000e+000,5.000000e+000      Xmin, Xmax
5.000000e+000,5.000000e+000      Ymin, Ymax
End
```

3. Type definition

```
1,0,2000      -of the polygon: No refinement, 2000 elements proposed
3,2,0         - of the line add in: medium refinement
4,3,0         - of the point add in: high refinement
```

2 FEFLOW Model File Format (*.fem) - Version 5.4

2.1 General structure of the file

2.1.1 Statement types

The FEFLOW problem ASCII files with the filename extension *.fem (FEFLOW model) has a general structure as described in section 2.1.2. The Problem file consists in a variable number of statements. The number of statements necessary to describe a FEFLOW model depends on the problem class and optional model properties (e.g. groups of observation points). The following table lists all statement types.

Chapter	Statement keyword	Contents of the statement
2.2.1	PROBLEM	Title of the problem
2.2.2	CLASS	Problem class identifier
2.2.3	DIMENS	Dimension descriptors
2.2.4	SCALE	Scaling parameters of graphic output
2.2.5	NODE	Node incidence matrix for the finite elements
2.2.6	COOR	xy coordinates of the finite element mesh
2.2.7	FLOW_I_BC	boundary conditions of the flow problem (flow boundaries)
2.2.8	INIT_I_FLOW	initial conditions of the flow problem (head initials)
2.2.9	MAT_I_FLOW	material conditions of the flow problem (flow materials)
2.2.10	TRAN_I_BC	boundary conditions of the mass transport problem (mass boundaries)
2.2.11	INIT_I_TRANS	initial conditions of the mass transport problem (mass initials)
2.2.12	MAT_I_TRAN	material conditions of the mass transport problem (mass materials)
2.2.13	HTRAN_I_BC	boundary conditions of the heat transport problem (heat boundaries)
2.2.14	INIT_I_HTRANS	initial conditions of the heat transport problem (heat initials)
2.2.15	MAT_I_HTRAN	material conditions of the heat transport problem (heat materials)
2.2.16	GK_COOR	Gauss-Krueger reference coordinates
2.2.17	EXTENTS	View extents at saving
2.2.18	TINI	Initial time for simulation
2.2.19	STEPS2	Time steps characteristics
2.2.20	PCS_OPTS	Predictor-corrector schemes options
2.2.21	GOBS	Observation point data
2.2.22	GROUPOBS	Definition of groups of observation points
2.2.23	REF_DIS_I	Definition of nodal reference distributions
2.2.24	REF_DISE_I	Definition of elemental reference distributions
2.2.25	SPECIES	Chemical species and reaction data
2.2.26	POWER	Time varying function data
2.2.27	MMLIST	Usage of time dependent material data
2.2.28	FENCE_SEG	Fences / segments data
2.2.29	ELEV_I	z coordinates of the mesh nodes
2.2.30	STATELEV	Slice types in 3D unconfined problems
2.2.31	PHREA_FLAG	Behavior of a phreatic layer at top
2.2.32	GRAVITY	Gravity vector orientation
2.2.33	PROJGRAVITY	Projected gravity for 2D horizontal density-dependent models
2.2.34	DIVERGENCE	Marker for divergence formulation of transport equations
2.2.35	ERROR_NORM	Type of used error norm for simulation
2.2.36	ANISOTROPIC	Use anisotropic computation of conductivity in 3D problems
2.2.37	ADAPTIVE	Levels and flags of nodes and elements for an adaptive mesh
2.2.38	AMR_METHOD	Method identifier for mesh adaption
2.2.39	REFERENCE_TRAN	Description of reference values for mass and heat transport
2.2.40	PHYSICS	Description of variable physical formulations
2.2.41	VISCOSITY	Properties of a user-defined viscosity relationship
2.2.42	UNSAT_OPTS	Options for unsaturated conditions
2.2.43	TVFDE	Parameters for thermal fluid density expansion

Chapter	Statement keyword	Contents of the statement
2.2.44	TSTAGES	Time stages for the simulation run
2.2.45	MAP_PALETTE	Custom color settings for background maps in 256 color mode
2.2.46	MAP_PATH	Path and color/style settings for background maps in true color mode
2.2.47	LEGEND	Legend settings incl. user-defined legends
2.2.48	FRACTURES	Data of discrete feature elements
2.2.49	OPTIONS	Option settings
2.2.50	SYMSOLV	Solver settings
2.2.51	MODULE	IFM module data
2.2.52	MOD_INFO	Information about loaded modules
2.2.53	END	Marker of the end of problem data

2.1.2 Order of statements

The first six statements and the last statement have fixed positions in the file. The last statement has to be the END statement. The other statements can be ordered arbitrary. Most of the statement types have to be unique in the fem FEFLOW model file.

Statement No.	Statement description	Statement keyword
1	Problem title	PROBLEM
2	Problem class identifiers	CLASS
3	Dimension	DIMENS
4	Scaling parameters	SCALE
5	Node incidence matrix of the elements	NODE
6	XY coordinates of the mesh nodes	COOR
7 - (n-1)	arbitrary ordered mandatory and optional model data	
n	End of fem description	END

2.1.3 Rules for statements

Statements consists of a keyword to identify the statement and the statement data. There are also statements consisting of the keyword only. The following format rules for statements are to be considered:

- All keywords have to begin at the column 1 of the line. All other data must not begin with a capital letter at position 1 in a line.
- Keyword has to be written in capital letters.
- The statement data can begin in the line of the keyword or in the next line.
- Statement data can be organized in one or more lines (see chapter 2).
- The physical units of all saved data are the native FEFLOW units. They do not necessarily coincide with the input units in the problem editor. For material data, units can also differ between time-constant and time-dependent data. Please check the corresponding sections of this file format description for the unit of a specific parameter.

2.2 Description of the statement formats

2.2.1 PROBLEM

This problem title description consists of only one line. Directly after the keyword a „:“ appears. All characters up to the end of this line are the text of the problem description.

```
PROBLEM: Description text of some words
```

2.2.2 CLASS

The statement data are organized in two lines:

```
CLASS (v.version)
ic1 ic2 proj ndm n_layers
```

Description of the logical values:

Name	Format	Description
version	%.4g	FEFLOW version used to create this file
ic1	%4d	Problem class: 0 = Flow and mass transport problem 2 = Separate flow problem 5 = Flow and heat transport problem 8 = Flow and thermohaline transport problem
ic2	%4d	Time mode of problem class: 0 = Steady state flow / transient transport 1 = Transient flow / transient transport 2 = Steady state flow / steady state transport
proj	%4d	Problem orientation: 0 = Horizontal problem 1 = Vertical problem 2 = Axisymmetric problem
ndm	%4d	Problem dimension: 2 = Two dimensional problem (2D) 3 = Three dimensional problem (3D)
n_layers	%4d	Number of element layers of 3D problems
ic0	%4d	Switch for saturated / unsaturated conditions: 0 = Saturated media (groundwater) 1 = Unsaturated media 2 = Unsaturated steady-state linearized Richards equation: Fast solution of Unsaturated flow SYstems (FUSY)
save_fsize_rreal	%4d	Precision of the result savings in bytes: 4 = 32 bit precision (format: 13.6e) 8 = 64 bit precision (format: 21.14le)
save_fsize_creal	%4d	Precision of the coordinate savings in bytes: 4 = 32 bit precision (format: 13.6e) 8 = 64 bit precision (format: 21.14le)

Note: All format specifications below refer to 64 bit precision! For 32 bit precision replace '21.14le' by '13.6e'!

2.2.3 DIMENS

The following two lines describe the problem dimensions:

```
DIMENS
np ne nbn numb_dt icrank upwind obs optim phreatic nwca np_cor
adaptive_mesh special_fem_process_id sorption_type reaction_type
```

dispersion_type

Description of the logical values:

Name	Format	Description
np	%6d	Number of mesh nodes (in 3D: sum of all slices)
ne	%6d	Number of finite elements (in 3D: sum of all layers)
nbn	%6d	Number of nodes of every finite element
numb_dt	%6d	Number of time steps for constant time step mode Default value: 20
icrank	%6d	Crank-Nicolson control variable to switch to CN scheme Default value: 200
upwind	%6d	Flag to define upwinding: 0 = No upwinding (default) 1 = Streamline upwinding 2 = Full upwinding 3 = Flux corrected transport (FCT)
obs	%6d	Number of observation points
optim	%6d	Mesh optimization identifier: 0 = No mesh optimization 1 = Mesh optimization using the RCM technique
phreatic	%6d	Kind of aquifer class: 0 = confined aquifer (no free surface) 1 = unconfined aquifer (possible free surface)
nwca	%6d	Kind of automatic time stepping predictor-corrector schemes: 1 = Forward Euler / backward Euler 2 = Forward Adams-Bashforth / backward trapezoid rule
np_cor	%6d	Number of edge nodes per slice (only for triparabolic 20-noded elements in 3D, else np_cor = 0)
adaptive_mesh	%6d	Usage of adaptive mesh refinement: 0 = No adaptive finite element mesh 1 = Performing adaptive finite element mesh
special_fem_process_id	%6d	internal flag, set it to 0 if creating a *.fem file
sorption_type	%6d	Type of sorption: 0 = Henry sorption isotherm 1 = Freundlich sorption isotherm 2 = Langmuir sorption isotherm
reaction_type	%6d	Type of reaction: 0 = First-order decay 1 = Michaelis-Menten mechanism 2 = Decay chains (consecutive reaction)
dispersion_type	%6d	Type of dispersion: 0 = Standard linear dispersion tensor (Bear & Scheidegger) 1 = Nonlinear (second-order) dispersion tensor (Hassanizadeh)

2.2.4 SCALE

The scaling data are important to define the coordinate system of the graphics view and the internal coordinate representation. They are described by:

```
SCALE
scale_fac, scale_l, scale_ratio, eps, x_o, y_o
```

Attention: comma separated values.

The logical values have the following meaning:

Name	Format	Description
scale_fac	%13.6e	Scaling factor (meters per pixel of the graphics view). Unused.
scale_l	%13.6e	Extension of the graphics view [m].
scale_ratio	%13.6e	Y to X scaling factor.
eps	%13.6e	Error tolerance of simulation convergence (RMS norm)
x_o	%13.6e	x coordinate of the Gauss Krueger offset in the graphics view [m]
y_o	%13.6e	y coordinate of the Gauss Krueger offset in the graphics view [m]

2.2.5 NODE

The nodes incidence matrix defines for each finite element the mesh nodes defining the element edges. The orientation of the nodes is always counter clockwise. In case of 3D problems first the upper areas edges and then the bottom area edges are defined.

```

NODE
for (elem = 1; elem <= ne; elem++) {
  for (node = 1; node <= nbn; node++)
    nop[elem][node]
  }
<newline>
}

```

The logical values are:

Name	Format	Description
elem		Element index, maximal ne elements
ne		Number of elements (see DIMENS)
node		Index for nodes of the element
nbn		Number of nodes per element (see DIMENS)
nop	%5d (if np < 10000) %7d (otherwise)	nop[elem][node] is the index of the mesh node (in the COOR data) of the node-th node of element elem

NOTE: All nodes defining one element take place in one line. The order of the element line defines the internal element index of FEFLOW. There are no commata between the values.

2.2.6 COOR

The xy coordinates of the finite element mesh are stored in the following format:

```

COOR
for (line =1, node=1; line <= (np_2d+np_cor-1)/12 + 1; line++) {
  for (col=1; col <=12; col++,node++) {
    if (node <= np)
      x [DIM_X][node]
  }
}
for (line =1, node=1; line <= (np_2d+np_cor-1)/12 + 1; line++) {
  for (col=1; col <=12; col++,node++) {
    if (node <= np)
      x [DIM_Y][node]
  }
}
}

```

The logical variables are:

Name	Format	Description
line		Index of lines in the fem file
col		Index of columns in the fem file
np_2d		Number of mesh nodes in one slice np_2d = np / (n_layers + 1) for 3D problems np_2d = np for 2D problems
np		Number of mesh nodes at all slices (see DIMENS)
DIM_X		Array index for the X coordinates
DIM_Y		Array index for the Y coordinates node Index of the mesh node
x[DIM_Y][node]	%21.14le	Y coordinate of mesh node <i>node</i> [m]
x[DIM_X][node]	%21.14le	X coordinate of mesh node <i>node</i> [m]

NOTE: The coordinates are written in 12 columns per line as comma separated values. First all X coordinates are saved. Then - beginning at a new line - the Y coordinates follow. The order of node coordinates follows the internal node order.

2.2.7 FLOW_I_BC

The flow boundary conditions are stored in the following format:

```

FLOW_I_BC
for (bc_type = MIN_flow_bc_type; bc_type <= MAX_flow_bc_type;
bc_type++) {
    if (boundary_conditions_with_this_bc_type_exist) {
        bc_type bc_value_list
        bc_node_list
    }
}

```

The logical variables are:

Name	Format	Description
bc_type	%8x	Hexadecimal code of the boundary condition type including constraint information
bc_value list	n(%21.14le)	n Values describing the boundary condition and constraints: The first value is the original boundary condition value or the time function index used for the BC. The following values are the constraint values in the same order they appear in the FEFLOW menus from top to bottom. These constraints can also be values or time function Ids. <i>Units boundary conditions:</i> 1 st kind BC: [m] 2 nd kind BC: [m/d] (2d unconfined/3D), [m ² /d] (2D confined) 3 rd kind BC: [m] 4 th kind BC: [m ³ /d] <i>Units constraints:</i> Constrained by head: [m] Constrained by flux: [m ³ /d]
bc_node_list	see next table	The node indices for this type and value of boundary condition. May be one or more node indices.

Format of node lists (e.g.: bc_node_lists):

A node list consists of blank separated entries of two types:

- single node entry: a single *node_index*
- node range entry: A range of nodes (from start node to end node) will be stored as: *node_index-node_index*.
The first node value is the startnode index.
The second node value is the endnode index
Directly behind the first node value has to follow a „minus“ sign.
- *node_index*: A right bound integer value.

The number of digits is always the same for all node indices to get a good readable design. It is not mandatory.

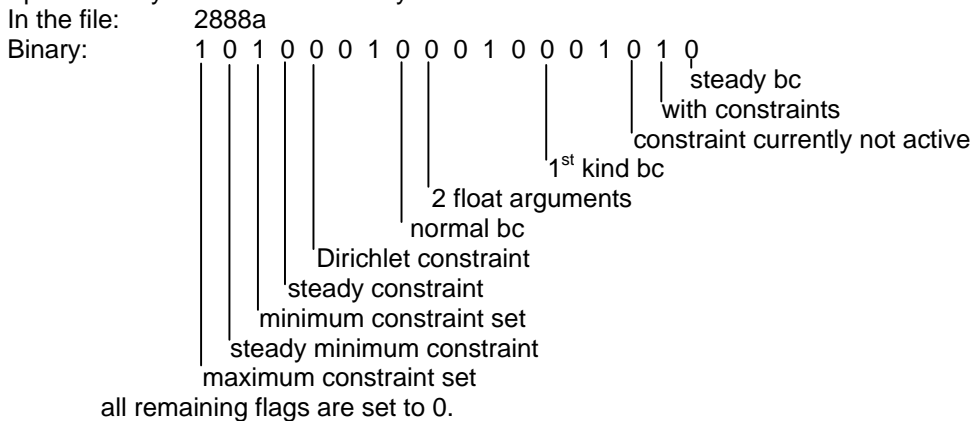
The number of columns of *node_values* in one line depends on the number of digits of the node values. The rule is: columns = 80 / (digits + 1).

The node list begins after a value (e.g. at MAT_I_TRAN) or after two tabs at the beginning of the line (e.g. at FLOW_I_BC).

Decoding / encoding the bc_type:

The bc type including the bcc information is encoded transferring a binary code into a hexadecimal code, which is stored. In the corresponding binary code each bit represents a switch. The order is always from the right side to the left side.

For example a steady-state head boundary condition with min/max flux constraint:



The following source code contains:

- The structure type definition of the boundary condition code
- The encoding routine
- The decoding routine

```
typedef struct fem_bccode { /* Bit structure for coding boundary conditions and constraints
*/
/* BC bits: */
unsigned bc_unst : 1; /* Status bit for b.c.:
0 - steady,
1 - unsteady (transient) power function */
unsigned bc_cset : 1; /* Constraint set bit:
0 - no constraints are introduced (or operative),
1 - constraints are set and correlated to b.c.'s */
unsigned bc_cuse : 1; /* Constraint usage bit:
0 - not in use,
1 - constraints are in use */
unsigned bc_type : 3; /* Identify type of b.c.:
0 - no b.c.
1 - Dirichlet b.c.
2 - Neumann b.c.
3 - Cauchy b.c.
4 - Single well b.c.
5 - Gradient-type 2nd kind (Neuman) b.c. */
unsigned bc_count : 4; /* Number of used float arguments */
unsigned bc_spec : 1; /* Special bits: Identify normal or integrated flux b.c.'s:
0 - normal b.c.
1 - integrated flux b.c. */
/* BCC type bits: */
```

```

unsigned bcc_type : 3; /* Identify type of constraint to be used:
0 - no constraint
1 - Dirichlet constraint
2 - Neumann constraint
3 - Cauchy constraint
4 - single well constraint
5 - special 3rd kind head constraint */
/* BCC temporal bits: */
unsigned bcc_unst : 1; /* Status bit for constraint b.c.:
0 - steady,
1 - unsteady (transient) power function */
/* BCCmin bits: */
unsigned bcc_min_set: 1; /* BCC minimum status in the 'bc' array:
0 - no minimum constraint is set,
1 - minimum constraint is set (2nd place in 'bc') */
unsigned bcc_min_unst : 1; /* BCC minimum temporal status bit:
0 - steady minimum constraint,
1 - unsteady (transient) power function */
/* BCCmax bits: */
unsigned bcc_max_set: 1; /* BCC maximum status in the 'bc' array:
0 - no maximum constraint is set,
1 - maximum constraint is set (3rd place in 'bc') */
unsigned bcc_max_unst : 1; /* BCC maximum temporal status bit:
0 - steady maximum constraint,
1 - unsteady (transient) power function */
/* BCChmin bits: */
unsigned bcc_hmi_set: 1; /* BCC h-minimum status in the 'bc' array:
0 - no h-minimum constraint is set,
1 - h-minimum constraint is set (4th place in 'bc') */
unsigned bcc_hmi_unst : 1; /* BCC h-minimum temporal status bit:
0 - steady h-minimum constraint,
1 - unsteady (transient) power function */
/* BCChmax bits: */
unsigned bcc_hma_set: 1; /* BCC h-maximum status in the 'bc' array:
0 - no h-maximum constraint is set,
1 - h-maximum constraint is set (5th place in 'bc') */
unsigned bcc_hma_unst : 1; /* BCC h-maximum temporal status bit:
0 - steady h-maximum constraint,
1 - unsteady (transient) power function */
/* BC holder bits: (intermediate saving) */
unsigned h_bc_type : 3; /* Save original boundary condition type */
unsigned h_bc_unst : 1; /* Save original temporal bit */
unsigned h_bc_loc : 3; /* Save entry location for the changed boundary value:
0 - B.C. value is replaced by bc.v[n][1], minimum,
1 - B.C. value is replaced by bc.v[n][2], maximum,
2 - B.C. value is replaced by bc.v[n][3], head minimum,
3 - B.C. value is replaced by bc.v[n][4], head maximum,
4 - B.C. value is replaced by bc.v[n][5], zero flux */
unsigned h_bc_neg : 1; /* Marks the negative sign for the changed boundary value:
0 - No change of sign,
1 - Negative sign of values has been used */
unsigned bcc_fs : 1; /* Marks an intermediate free-surface constraint:
hydraulic head touches bottom elevation (dry nodes) */
} BCCODE;
/*-----

** encode_bccode - Encoding the b.c. coding flag
*/
unsigned int encode_bccode (flag)
BCCODE flag;
{
#define ENCODE_FLAG(name, size) mask |= (unsigned int)(name & ((1<<size)-1)) << bp; bp += size
unsigned int mask;
int bp; /* bit position */
mask = 0L; bp = 0;
ENCODE_FLAG(flag.bc_unst, 1);
ENCODE_FLAG(flag.bc_cset, 1);
ENCODE_FLAG(flag.bc_cuse, 1);
ENCODE_FLAG(flag.bc_type, 3);
ENCODE_FLAG(flag.bc_count, 4);

```

```

ENCODE_FLAG(flag.bc_spec, 1);
ENCODE_FLAG(flag.bcc_type, 3);
ENCODE_FLAG(flag.bcc_unst, 1);
ENCODE_FLAG(flag.bcc_min_set, 1);
ENCODE_FLAG(flag.bcc_min_unst, 1);
ENCODE_FLAG(flag.bcc_max_set, 1);
ENCODE_FLAG(flag.bcc_max_unst, 1);
ENCODE_FLAG(flag.bcc_hmi_set, 1);
ENCODE_FLAG(flag.bcc_hmi_unst, 1);
ENCODE_FLAG(flag.bcc_hma_set, 1);
ENCODE_FLAG(flag.bcc_hma_unst, 1);
ENCODE_FLAG(flag.h_bc_type, 3);
ENCODE_FLAG(flag.h_bc_unst, 1);
ENCODE_FLAG(flag.h_bc_loc, 3);
ENCODE_FLAG(flag.h_bc_neg, 1);
ENCODE_FLAG(flag.bcc_fs, 1);
return mask;
#undef ENCODE_FLAG
}
/*-----

```

```

** decode_bccode - Decoding the b.c. coding flag
*/
BCCODE decode_bccode (mask)
unsigned int mask;
{
#define DECODE_FLAG(size) (mask >> bp) & ((1<<size)-1); bp += size
BCCODE flag;
int bp;
ZERO(flag); bp = 0;
flag.bc_unst = DECODE_FLAG(1);
flag.bc_cset = DECODE_FLAG(1);
flag.bc_cuse = DECODE_FLAG(1);
flag.bc_type = DECODE_FLAG(3);
flag.bc_count = DECODE_FLAG(4);
flag.bc_spec = DECODE_FLAG(1);
flag.bcc_type = DECODE_FLAG(3);
flag.bcc_unst = DECODE_FLAG(1);
flag.bcc_min_set = DECODE_FLAG(1);
flag.bcc_min_unst = DECODE_FLAG(1);
flag.bcc_max_set = DECODE_FLAG(1);
flag.bcc_max_unst = DECODE_FLAG(1);
flag.bcc_hmi_set = DECODE_FLAG(1);
flag.bcc_hmi_unst = DECODE_FLAG(1);
flag.bcc_hma_set = DECODE_FLAG(1);
flag.bcc_hma_unst = DECODE_FLAG(1);
flag.h_bc_type = DECODE_FLAG(3);
flag.h_bc_unst = DECODE_FLAG(1);
flag.h_bc_loc = DECODE_FLAG(3);
flag.h_bc_neg = DECODE_FLAG(1);
flag.bcc_fs = DECODE_FLAG(1);
return flag;
#undef DECODE_FLAG
}

```

2.2.8 INIT_I_FLOW

The initial head distribution at mesh nodes will be stored as follows:

```
INIT_I_FLOW
LOOP_OF_head_ini_levels_FROM_MINIMUM_TO_MAXIMUM {
    head_ini_level node_list
}
```

The logical values are:

Name	Format	Description
head_ini_level	%21.14le	Head initial value [m]
node_list	see at FLOW_I_BC	Nodes carrying the head_ini_level

2.2.9 MAT_I_FLOW

The flow material data are stored in the following format:

```
MAT_I_FLOW
LOOP_OF_material_type_index_FROM_MINIMUM_TO_MAXIMUM {
    material_type_index default_value comment
    material_value node_list
    mathed_param_data

                                SAVE, &f[i].mat);
}
```

The logical values are:

Name	Format	Description
material_type_index	%d	Code number of the material type (see list below this table)
default_value	%13e	The default value for all elements.
comment	„comment it“	Short description of the material property. Enclosed in double quotes.
material_value	%13e\t	A material property value
node_list	see at FLOW_I_BC	The elements where the material_value is to overwrite the default value.
mathed_param_data	see Online Help	equations specified in FEMATHED in XML format

The following types of flow material are supported:

material_type_index	Comment
100	Transmissivity [m ² /d] or Conductivity [m/d]
101	Conductivity in x-direction for 3D [m/d]
102	Anisotropy factor [1]
103	Conductivity in y-direction for 3D [m/d]
104	Anisotropy angle [degree]
105	Conductivity in z-direction for 3D [m/d]
106	Aquifer bottom elevation [m]
107	In/outflow on top or bottom for 3D [m/d]
108	Aquifer top elevation [m]
109	Density ratio for 3D and/or thermohaline transport [1]
110	Storativity (drain- or fillable) or density ratio [1]
111	Volumetric expansion coefficient for heat transport [1/K]
112	Storage compressibility [1/m] (2D unconfined / 3D) [1] (2D confined)
113	Source/sink for flow [m/d] (2D) [1/d] (3D)
114	Transfer rate for inflow [m/d] (2D unconfined / 3D) [1/d] (2D confined)
115	Transfer rate for outflow [m/d] (2D unconfined / 3D) [1/d] (2D confined)
116	Parametric unsaturated model type: 1... [1]
117	Porosity for unsaturated storativity [1]
118	Maximum saturation (S _s) [1]
119	Residual saturation (S _r) [1]
120	Fitting coefficient in capillary head (A) [1/m] or (Alpha) [m]
121	Fitting exponent in capillary head curve (n) [1], (B) [1] or (Beta) [1]
122	Fitting exponent or coefficient in relative conductivity curve (delta) [1] or (A) [1/m]
123	Fitting exponent in relative conductivity curve (B) [1]
124	Air-entry pressure head [m]
125	not used
126	Transmissivity [m ² /d]
127	Maximum saturation (S _s) hysteretic drying curve [1]
128	Residual saturation (S _r) hysteretic drying curve [1]
129	Fitting coefficient in capillary head (A) [1/m] or (Alpha) [m], hysteretic drying curve
130	Fitting exponent in capillary head curve (n) [1], (B) [1] or (Beta) [1], hysteretic drying curve
131	Fitting exponent or coefficient in relative conductivity curve (delta) [1] or (A) [1/m], hysteretic drying curve
132	Fitting exponent in relative conductivity curve (B) [1] hysteretic drying curve
133	Air-entry pressure head hysteretic drying curve [m]
134	Euler phi-angle for general 3D anisotropy of conductivity [deg]
135	Euler psi-angle for general 3D anisotropy of conductivity [deg]
136	Euler theta-angle for general 3D anisotropy of conductivity [deg]

2.2.10 TRAN_I_BC

The mass transport boundary conditions are stored in the following format:

```
TRAN_I_BC
for (bc_type = MIN_mass_bc_type; bc_type <= MAX_mass_bc_type;
    bc_type++) {
    if (boundary_conditions_with_this_bc_type_exist) {
        bc_type bc_value_list
        bc_node_list
    }
}
```

TRAN_I_BC has multiple implementations in the file if multi-species mass transport is used. The order of the implementations refers to the order of the species in FEFLOW.

The logical variables are:

Name	Format	Description
bc_type	%8x	Hexadecimal code of the boundary condition type including the constraints informations
bc_value list	n(%21.14e)	n Values describing the boundary condition and constraints: The first value is the original boundary condition value or the time function index used for the BC. The following values are the constraint values in the same order they appear in the FEFLOW menus from top to bottom. These constraints can also be values or time function IDs. There are always 5 values for the constraints, the last one is a dummy. <i>Units boundary conditions:</i> 1 st kind BC: [mg/l] 2 nd kind BC: [g/(m ² d)] (2d unconfined/3D), [g/(md)] (2D confined) 3 rd kind BC: [mg/l] 4 th kind BC: [g/d] <i>Units constraints:</i> Constrained by head: [m] Constrained by mass flux: [g/d] Constrained by mass: [mg/l]
Bc_node_list	see at FLOW_I_BC	The nodes indices for this type and value of boundary condition. May be one or more node indizes.

2.2.11 INIT_I_TRANS

The initial mass distribution at mesh nodes will be stored as follows:

```
INIT_I_TRANS
LOOP_OF_mass_ini_values_FROM_MINIMUM_TO_MAXIMUM {
    mass_ini_value node_list
}
```

INIT_I_TRANS has multiple implementations in the file if multi-species mass transport is used. The order of the implementations refers to the order of the species in FEFLOW.

The logical values are:

Name	Format	Description
mass_ini_value	%21.14le	Mass initial value (concentration) [mg/l]
node_list	see at FLOW_I_BC	Nodes carrying the mass_ini_value

2.2.12 MAT_I_TRAN

FEFLOW system information Description of the *fem* File Format (Version 4.50) page 10
The mass transport material data are stored in the following format:

```

MAT_I_TRAN
LOOP OF chemical species index {
  LOOP_OF_material_type_index_FROM_MINIMUM_TO_MAXIMUM {
    material_type_index default_value comment
    material_value node_list
    mathed_param_data
  }
}

```

MAT_I_TRAN has multiple implementations in the file in case of multi-species mass transport. The order of the implementations refers to the order of the species in FEFLOW. The logical values are:

Name	Format	Description
material_type_index	%d	Code number of the material type (see list below this table)
default_value	%13e	The default value for all elements.
Comment	„comment it“	Short description of the material property. Enclosed in double quotes.
Material_value	%13e	A material property value
node_list	see at FLOW_I_BC	The nodes where the material_value is to overwrite the default value.
mathed_param_data	XML	equations from FEMATHED in XML format – for format see Online Help

The following material types for mass transport are supported:

material_type_index	Comment
200	Aquifer thickness [m]
201	Porosity [1]
202	Sorption coefficient [1]
203	Molecular diffusion [m ² /d]
204	Longitudinal dispersivity [m]
205	Transverse dispersivity [m]
206	Decay rate [1/s]
207	Source/sink for transport [mg/(ld)]
208	Transfer rate for influxing transport [m/d] (2D unconfined / 3D) [m ² /d] (2D confined)
209	Transfer rate for outfluxing transport [m/d] (2D unconfined / 3D) [m ² /d] (2D confined)
210	Additional sorption coefficient for the Freundlich [1] or Langmuir [l/mg] isotherm
211	Additional reaction coefficient for Michaelis-Menten mechanism [mg/l]
212	Additional dispersion coefficient for nonlinear dispersion [1]

2.2.13 HTRAN_I_BC

The heat transport boundary conditions are stored in the following format:

```
HTRAN_I_BC
for (bc_type = MIN_heat_bc_type; bc_type <= MAX_heat_bc_type;
    bc_type++) {
    if (boundary_conditions_with_this_bc_type_exist) {
        bc_type bc_value_list
        bc_node_list
    }
}
```

The logical variables are:

Name	Format	Description
bc_type	%8x	Hexadecimal code of the boundary condition type including the constraints informations
bc_value list	n(%21.14le)	n Values describing the boundary condition and constraints: The first value is the original boundary condition value or the time function index used for the BC. The following values are the constraint values in the same order they appear in the FEFLOW menus from top to bottom. These constraints can also be values or time function IDs. There are always 5 values for the constraints, the last one is a dummy. <i>Units boundary conditions:</i> 1 st kind BC: [deg C] 2 nd kind BC: [J/(m ² d)] (2d unconfined/3D), [g/(md)] (2D confined) 3 rd kind BC: [deg C] 4 th kind BC: [g/d] <i>Units constraints:</i> Constrained by head: [m] Constrained by mass flux: [g/d] Constrained by mass: [mg/l]
Bc_node_list	see at FLOW_I_BC	The nodes indices for this type and value of boundary condition. May be one or more node indices.

2.2.14 INIT_I_HTRANS

The initial temperature distribution at mesh nodes will be stored as follows:

```
INIT_I_HTRANS
LOOP_OF_heat_ini_values_FROM_MINIMUM_TO_MAXIMUM {
    heat_ini_value node_list
}
```

The logical values are:

Name	Format	Description
heat_ini_value	%21.14le	Heat initial value (temperature) [deg C]
node_list	see at FLOW_I_BC	Nodes carrying the heat_ini_value

2.2.15 MAT_I_HTRAN

The heat transport material data are stored in the following format:

```
MAT_I_HTRAN
LOOP_OF_material_type_index_FROM_MINIMUM_TO_MAXIMUM {
    material_type_index default_value comment
    material_value node_list
    mathed_param_data
}
```

The logical values are:

Name	Format	Description
material_type_index	%d	Code number of the material type (see list below this table)
default_value	%13e	The default value for all elements.
comment	„comment it“	Short description of the material property. Enclosed in double quotes.
material_value	%13e\t	A material property value
node_list	see at FLOW_I_BC	The nodes where the material_value is to overwrite the default value.
mathed_param_data	XML	equations from FEMATHED in XML format – for format see Online Help

The following material types for heat transport are supported:

material_type_index	Comment
300	Aquifer thickness (heat) [m]
301	Porosity (heat) [1]
302	Volumetric heat capacity of solid [J/(m ³ K)]
303	Heat conductivity of solid [J/m/d/K]
304	Longitudinal thermo-dispersivity [m]
305	Transverse thermo-dispersivity [m]
306	Heat source/sink of fluid [J/(m ³ K)]
307	Heat source/sink of solid [J/(m ³ d)]
308	Thermo-transfer rate for inflowing transport [J/(m ² dK)]
309	Thermo-transfer rate for outflowing transport [J/(m ² dK)]
310	Volumetric heat capacity of fluid [J/(m ³ K)]

311	Heat conductivity of fluid [J/(mK)]
-----	-------------------------------------

2.2.16 GK_COOR

The Gauss-Krueger offset are the coordinates of the lower left corner of the graphics view in FEFLOW. If x_o and y_o are not zero, this point will be shifted into the graphics window (see the „shift origin“ function in the „problem measure“ menu. The Gauss-Krueger offset is defined as:

```
GK_COOR
xlGK, ylGK
```

with two comma separated values:

Name	Format	Description
xlGK	%16.6f	X coordinate of the Gauss-Krueger offset
ylGK	%16.6f	X coordinate of the Gauss-Krueger offset

2.2.17 EXTENTS

Under EXTENTS, the current view extent of the working window is saved.

```
EXTENTS
defext.xmin, defext.ymin, defext.xmax, defext.ymax,
curext.xmin, curext.ymin, curext.xmax, curext.ymax,
```

Name	Format	Description
defext	-	default extent
curext	-	current extent
xmin	%16.6f	x coordinate of the lower left corner
ymin	%16.6f	y coordinate of the lower left corner
xmax	%16.6f	x coordinate of the upper right corner
ymax	%16.6f	y coordinate of the upper right corner

2.2.18 TINl

The starting time ini_time of the simulation will be defined as :

```
TINl
ini_time
```

with

Name	Format	Description
ini_time	%16.6f	Start time of simulation [d]

2.2.19 STEPS2

The **data for the predefined time levels** are defined in the following format (12 values per line):

```
STEPS2
for (line=1, level=1; line <= (numb_dt+1)/12 + 1; line++) {
  for (col=1; col <= 12; col++, level++) {
    if (level <= numb_dt)
      tsteps [level]
  }
}
```

The logical variables are:

Name	Format	Description
line		Index of lines in the fem file
col		Index of columns in the fem file
numb_dt		Number of time levels
level		Index of the time level
tsteps[level]	%13.6f	Time niveau value of the level-th timestep

Alternatively the **data to control the automatic timestepping schemes** are defined in the following format (3 values) - if the automatic timestepping schemes are used:

```
STEPS2
init_time, init_stepsize, final_time
```

The logical variables are:

Name	Format	Description
init_time	%13.6f	Initial time value (at simulation start)
init_stepsize	%13.6f	Time step size of the first simulation step
final_time	%13.6f	Final time of simulation

2.2.20 PCS_OPTS

The options for the predictor-corrector schemes used for automatic time stepsize control are defined by:

```
PCS_OPTS
lim_incrate, incrate, lim_mtsize, mtsize, cntr_mode, targets.targ_h,
targets.targ_s, targets.targ_c, targets.targ_t
```

The logical values are declared as:

Name	Format	Description
lim_incrate	%2d	Boolean value to indicated if the increasing time step size rate is limited to the value of incrate or not: 0 = not limited 1 = limited
incrate	%13.6le	Limiting value of maximal increasing time step size rate
lim_mtsize	%2d	Boolean value to indicated if the maximal time step size is limited to the value of mtsize or not: 0 = not limited 1 = limited
mtsize	%13.6le	Limiting value of maximal time step size
cntr_mode	%2d	Time step control: 0 = Predictor-corrector error-based time step control 1 = Aggressive target-based time step control
targets.targ_h	%13.6le	Target change parameter: Head
targets.targ_s	%13.6le	Target change parameter: Saturation
targets.targ_c	%13.6le	Target change parameter: Concentration
targets.targ_t	%13.6le	Target change parameter: Temperature

2.2.21 GOBS

Observation point data are stored as:

```
GOBS
for (id=1; id <= obs; id++) {
  type[id], node[id], ele[id], slc[id], x[id], y[id]
  "curve_title", #colors[0].red#colors[0].green#colors[0].blue,
  line_style, line_width
  colors[1].red#colors[1].green#colors[1].blue, marker_type,
  marker_size, visible
```

}

with the following definitions:

Name	Format	Description
id		Index of current observation point
type[id]	%5d	Type of observation point: 0 = Nodal 1 = Arbitrary
node[id]	%5d	Mesh node index (if type == nodal), default value: 0
ele[id]	%5d	Finite element index (if type == arbitrary), default value: 0
slc[id]	%5d	slice index containing the observation point (beginning with 0 for top slice)
x[id]	%21.14le	X coordinate of the observation point
y[id]	%21.14le	Y coordinate of the observation point
curve_title	%s	Curve title
color[0]...	%02X	RGB color of the line
line_style	%d	line style
line_width	%d	line width
color[1]...	%02X	RGB color of the markers
marker_type	%d	marker type
marker_size	%d	marker size
visible	%d	visibility of the markers: 0 = invisible, 1 = visible

2.2.22 GROUPOBS

Data for groups of observation points are stored in the following format:
FEFLOW system information Description of the *fem* File Format (Version 4.50) page 14

```

GROUPOBS
group_number version_flag maxgroups
for (group_id=0; group_id < group_number; group_id++) {
  npoints[group_id] group_id
  for (id=1; id <= npoints[group_id]; id++) {
    type[group_id][id], node[group_id][id], ele[group_id][id],
    slc[group_id][id], x[group_id][id], y[group_id][id]
  }
}

```

with the following definitions:

Name	Format	Description
group_number	%5d	Number of used groups of observation points
version_flag		version flag: 0 = version < 4.5.22, maxgroups is not read 1 = version >= 4.5.22, maxgroups is read
maxgroups	%8d	max. number of observation point groups taking place in the current block
npoints[group_id]	%5d	Number of observation points in the group
group_id	%4d	Group identifier (starting with 0)
id		Index of current observation point
type[id]	%5d	Type of observation point: 0 = Nodal 1 = Arbitrary
node[id]	%5d	Mesh node index (if type == nodal), default value: 0
ele[id]	%5d	Finite element index (if type == arbitrary), default value: 0
slc[id]	%5d	slice index containing the observation point (beginning with 0 for top slice)
x[id]	%21.14le	X coordinate of the observation point
y[id]	%21.14le	Y coordinate of the observation point

2.2.23 REF_DIS_I

The nodal reference distributions will be stored as follows:

```
REF_DIS_I
  ref_distr.numb, np
  for (i=0; i < ref_distr.numb;i++) {
    name
    LOOP_OF_ref_value_FROM_MINIMUM_TO_MAXIMUM {
      ref_value node_list
    }
  }
```

The logical values are:

Name	Format	Description
ref_distr.numb	%5d	Number of reference distributions
np	%7d	Number of mesh nodes
name	%s	Name of the nodal reference distribution
ref_value	21.14le	Value at a node
node_list	see at FLOW_I_BC	Nodes carrying the ref_value

2.2.24 REF_DISE_I

The elemental reference distributions will be stored as follows:

```
REF_DISE_I
  ref_distr.e.numb, ne
  for (i=0; i < ref_distr.e.numb;i++) {
    name
    LOOP_OF_ref_value_FROM_MINIMUM_TO_MAXIMUM {
      ref_value element_list
    }
  }
```

The logical values are:

Name	Format	Description
ref_distr.e.numb	%5d	Number of reference distributions
ne	%7d	Number of mesh elements
name	%s	Name of the elemental reference distribution
ref_value	21.14le	Value at an element
element_list	see at FLOW_I_BC	Elements carrying the ref_value

2.2.25 SPECIES

For multi-species mass transport, the species information and reaction kinetics are saved in the following format:

```
SPECIES
for (i=0; i <= n_species; i++) {
    phase,name
    i
    [reaction]
}
```

[reaction] depends on the kind of reaction kinetics:

- **Degradation**
index_1 ... index_m
k_1 ... k_n
n_1 ... n_n
- **Arrhenius**
index_1 ... index_m
k_1 ... k_n
n_1 ... n_n
- **Monod**
index_1 ... index_m
k_1 ... k_n
n_1 ... n_o
b_1 ... b_p
d_1 ... b_q
e_1 ... e_r
- **User-defined**
In user-defined mode, the reaction equations are stored in XML format. Find a list of the possible tags below. They should be self-explanatory. In case of doubts, please do not hesitate to contact support@wasy.de!

XML-style elements used in the .fme format

mathed	wraps everything, usually contains a 'prog', an optional 'blue' and an optional 'green'.
prog	wraps the formula definition
blue	wraps list of blue vars. informative only. will be ignored when read.
green	list of green vars. informative only. will be ignored when read.
row	container to make a single element from sequences of char, op, num etc.
id	a name, usually a variable. only A-Z, a-z and 0-9 allowed
num	a real number

integer	an integer number
char	internally used by id
op	<p>supertype for unary and binary operations. In the .fme format the operators are stored together with operands in a 'flat' 'row', not in a hierarchy suitable for evaluation. This is because the .fme format reflects the representation in the dialog which is tuned to make editing as simple as possible</p> <ul style="list-style-type: none"> + - addition - - subtraction * - multiplication eq - test for equality. The result is numeric '1' for equality and '0' for inequality ne - test for inequality lt - test for 'lower than' le - test for 'lower or equal' gt - test for 'greater than' ge - test for 'greater or equal' and - logical 'and' or - logical 'or' not - logical 'not'
newline	line break in long formulas, no influence on computations
sup	exponentiation template, has two subterms for base and xponent
if	selection template with three subcells (condition, 'then' branch, 'else' branch)
frac	fraction template, two subcells for numerator and denominator
sqrt	square root, one subcell
fence	parantheses to wrap subexpressions
unit	physical unit, for decoration purposes only
ln	natural logarithm template, one subcell for the argument
sin	sine template, one subcell for the argument
cos	cosine template, one subcell for the argument
abs	absolute value template, one subcell for the argument
piecewise	template for the 'piecewise ' operation: if a1 then a2 elsif a3 then a4 elseif a5 then ... else an endif
assign	assignment for temporary ('black' variables) within a 'prog' element.

2.2.26 POWER

The time function data are defined in the following format:

```
POWER
numb_power
for (pow_id=1; pow_id <= numb_power; pow_id++) {
    pow_id npoints[pow_id] type_old[pow_id] comment[pow_id] flag[pow_id]
        type_new[pow_id] cycle[pow_id] slope[pow_id]
    for (t=1; t <= npoints[pow_id]; t++) {
        time[pow_id][t], value[pow_id][t]
    }
}
```

with the following definitions:

Name	Format	Description
numb_power	%5d	Number of used power functions
pow_id	%4d	Power function identifier
npoints[pow_id]	%4d	Number of observation points in the power function pow_id
type_old[pow_id]	%4d	Type of interpolation between data points: 0 = polylined/constant 1 = Akima linear 2 = Akima cubic
comment[pow_id]	%s	Description string (included in double quotas)
flag[pow_id]	%c	Internal/interpolated ('i') or user defined ('u') power function
type_new[pow_id]	%d	Type of interpolation between data points: 0 = polylined 1 = Akima linear 2 = Akima cubic 7 = constant
cycle[pow_id]	%1d	Time mode: 0 = linear 1 = cyclic
slope[pow_id]	%6e	Slope duration while step-wise constant curves, default value: 0
t		Index of the data points in a power function
time[pow_id][t]	%14.6le	timelevel of point t in power function pow_id
value[pow_id][t]	%14.6le	function value of point t in power function pow_id

2.2.27 MMLIST

Time dependent material data will be stored in separate files (*.mli) or in the *.fem file (default). The syntax used in the ASCII *.mli file is mostly identical to the following description of the storage of time-dependent material data directly in the *.fem file.

Please note that the units differ from the storage of time-constant data!

```
MMLIST
n_mlist path
for (n_mlist = 0; id < n_mlist; n_mlist++) {
    kind material n_entries mod cycle active
    if (path == inline){
        MM_ENTRY_DATA
        for (i = 0; i < n_entries){
            kind material time_level ne
            Data for parameter at time time_level
            LOOP_OF_mat_value_FROM_MINIMUM_TO_MAXIMUM {
                mat_value element_list
            }
        }
    }
}
```

with the logical variables as:

Name	Format	Description
n_mlist	%5d	Number of time varying material properties
path	%s	Path of external mlist file (*.mli) or 'internal'
kind	%5d	Kind of MLIST: 2 = flow 0 = mass transport 5 = heat transport
material	%5d	Material property, see table below
n_entries	%5d	Number of time levels
mod	%5d	Mode of interpolation: 0 = polylined 1 = Akima linear
cycle	%5d	Time mode: 0 = linear 1 = cyclic
active	%5d	Usage of time dependant data, currently always 1
time_level	%13.6e	Time level for material data
ne	%6d	Number of elements
mat_value	%13.6e	Value of material property in an element at time_level
element_list	see at FLOW_I_BC	Elements carrying the mat_value

The following integers are used for coding the material parameter:

material
Flow materials (kind = 2)
Transmissivity or conductivity, conductivity in x-direction for 3D [10-4 m ² /s] (2D confined) [10-4 m/s] (2D unconfined, 3D)
Anisotropy factor [1], conductivity in y-direction for 3D [10-4 m/s]
Anisotropy angle [1], conductivity in z-direction for 3D [10-4 m/s]
Aquifer bottom elevation [m], In/outflow on top or bottom for 3D [m/d]
Aquifer top elevation [m], density ratio for 3D and/or thermohaline transport [1]
Storativity (drain- or fillable) [1] or density ratio [1], volumetric expansion coefficient for heat transport [1]
Storage compressibility [1]
Source/sink for flow [1/d] (3D) [m/d] (2D)
Transfer rate for inflow [m/d] (2D unconfined /3D) [1/d] (2D confined)
Transfer rate for outflow [m/d] (2D unconfined /3D) [1/d] (2D confined)
Coded parametric unsaturated model type, the values are as follows: 0 = Van Genuchten 1 = Brooks-Corey 2 = Haverkamp et al. 3 = Exponential 4 = Linear 5 = Van Genuchten modified
Porosity (used for unsaturated storativity) [1]
wetting curve parameters for hysteretic properties:
Maximum saturation (S _s) [1]
Residual saturation (S _r) [1]
Fitting coefficient in capillary head (A) [1/m] or (Alpha) [m]
Fitting exponent in capillary head curve (n) [1], (B) [1] or (Beta) [1]
Fitting exponent or coefficient in relative conductivity curve (delta) [1] or (A) [1/m]
Fitting exponent in relative conductivity curve (B) [1] or fitting exponent in capillary head curve (m) [1]

Air-entry pressure head [m]
drying curve parameters for hysteretic properties:
Maximum saturation (Ss) [1]
Residual saturation (Sr) [1]
Fitting coefficient in capillary head (A) [1/m] or (Alpha) [m]
Fitting exponent in capillary head curve (n) [1], (B) [1] or (Beta) [1]
Fitting exponent or coefficient in relative conductivity curve (delta) [1] or (A) [1/m]
Fitting exponent in relative conductivity curve (B) [1] or fitting exponent in capillary head curve (m) [1]
Air-entry pressure head [m]
Euler phi-angle for general 3D anisotropy of conductivity [deg]
Euler psi-angle for general 3D anisotropy of conductivity [deg]
Euler theta-angle for general 3D anisotropy of conductivity [deg]
Mass transport materials (kind = 0)
Aquifer thickness [m]
Porosity [1]
Sorption coefficient [1]
Molecular diffusion [m ² /d]
Longitudinal dispersivity [m]
Transverse dispersivity [m]
Decay rate [1/s]
Source/sink for transport [mg/d]
Transfer rate for influxing transport [m/d] (2D unconfined / 3D) [m ² /d] (2D confined)
Transfer rate for outfluxing transport [m/d] (2D unconfined / 3D) [m ² /d] (2D confined)
Additional sorption coefficient for the Freundlich [1] or Langmuir [l/mg] isotherm
Additional reaction coefficient for Michaelis-Menten mechanism [mg/l]
Additional dispersion coefficient for second-order dispersion law [1]
Heat transport materials (kind = 5)
Volumetric heat capacity of solid
Heat conductivity of solid
Heat source/sink of fluid
Heat source/sink of solid
Volumetric heat capacity of fluid
Heat conductivity of fluid

2.2.28 FENCE_SEG

The data for fences and line sections are defined in the following format:

```

FENCE_SEG
numb_fences
for (fen_id=1; fen_id <= numb_fences; fen_id++) {
    fen_id npoints[fen_id]
    for (p=1; p <= npoints[fen_id]; p++) {
        x[fen_id][p], y[fen_id][p]
    }
}

```

with the following definitions:

Name	Format	Description
numb_fences	%5d	Number of fences / sections
fen_id	%5d	Fence identifier
npoints[fen_id]	%4d	Number of points of fence fen_id
p		Index of the points in a fence
x[fen_id][p]	%21.14le	x coordinate of point p in fence fen_id
y[fen_id][p]	%21.14le	y coordinate of point p in nce fen_id

2.2.29 ELEV_I

The elevation distribution at mesh nodes will be stored as follows:

```

ELEV_I
for (slice=1; slice <= n_layers+1; slice++) {
  if (slice > 1) {
    slice
  }
  LOOP_OF_elevation_values[slice]_FROM_MINIMUM_TO_MAXIMUM {
    elevation_values[slice] node_list
  }
}

```

The logical values are:

Name	Format	Description
slice	%d	Index of slices
elevation_values [slice]	%21.14le	Z coordinates (elevation) of mesh nodes of slice <i>slice</i>
node_list	see at FLOW_I_BC	Nodes indices carrying the elevation_value[<i>slice</i>]. The node indices will be counted slice related for every slice in the range [1, np_2d]. The total node indices [1,np] will not be used here but for the FLOW_I_BC data.

2.2.30 STATELEV

The type of the slices used in 3D unconfined problems is stored in the following 20 column format:

```

STATELEV
for (line =1, slice=1; line <= (n_slices-1)/20 + 1; line++) {
  for (col=1; col <=20; col++,slice++) {
    if (slice <= n_slices)
      slice_status[slice]
  }
  <newline>
}

```

The logical variables are:

Name	Format	Description
line		Index of lines in the fem file
col		Index of columns in the fem file
n_slices		Number of slices in a 3D problem: n_slices = n_layers + 1 (see DIMENS)
slice		Index of the slices
slice_status[slice]	%5d	Status of the slice No. <i>slice</i> : 1 = Fixed 2 = Unspecified 3 = Movable 4 = Phreatic

2.2.31 PHREA_FLAG

This statement is set, if the specific option for phreatic conditions on top slice is set to 'confined'. It is used without any additional parameters:

```
PHREA_FLAG
```

2.2.32 GRAVITY

The gravity flag is only used in 3D. The gravity orientation vector is determined by the vector components in x, y and z direction in the following format:

```
GRAVITY
grav_x grav_y grav_z
```

The vector components are Boolean values defining the direction, not a gravity value. The formats are:

Name	Format	Description
grav_x	%3d	X component of gravity vector (default: 0)
grav_y	%2d	Y component of gravity vector (default: 0)
grav_z	%2d	Z component of gravity vector (default: -1)

2.2.33 PROJGRAVITY

PROJGRAVITY specifies the parameters for projecting the gravity vector in case of density-dependent simulations in a 2D horizontal model.

```
PROJGRAVITY
refid used
```

Name	Format	Description
refid	%2d	ID of the reference distribution used as aquifer bottom -1 = no distribution specified
used	%2d	0 = unused 1 = used

2.2.34 DIVERGENCE

The indicate, that the divergence form of transport equations will be used instead of the default convective form the keyword without any parameters will be inserted in the fem file:

```
DIVERGENCE
```

2.2.35 ERROR_NORM

The error norm entry defines the type of error norm computation used in the simulation.

```
ERROR_NORM
error_norm_type
```

Name	Format	Description
error_norm_type	%1d	Type of error norm computation method: 0 = Maximum error norm 1 = Absolute L1 error norm 2 = Euclidian RMS L2 error norm (default)

2.2.36 ANISOTROPIC / ANISOTROPIC2

For 3D problems the anisotropy can either be axis-parallel (no keyword), derived from the inclination of the layers (keyword ANISOTROPIC) or derived from explicitly input Euler angles (keyword ANISOTROPIC2).

2.2.37 ADAPTIVE

If adaptive mesh refinement will be used, the following data are defined:

```

ADAPTIVE
for (line =1, node=1; line <= (np_2d-1)/25 + 1; line++) {
  for (col=1; col <=25; col++,node++) {
    if (node <= np_2d)
      adapm_levels[node]
  }
  <newline>
}
for (line =1, elem=1; line <= (ne_2d-1)/25 + 1; line++) {
  for (col=1; col <=25; col++,elem++) {
    if (elem <= ne_2d)
      adapm_flags [elem]
  }
  <newline>
}

```

The logical variables are:

Name	Format	Description
line		Index of lines in the fem file
col		Index of columns in the fem file
np_2d		Number of mesh nodes at one slice
node		Index of the mesh node
adapm_levels[<i>node</i>]	%3d	Nodal level used for adaptive meshing
ne_2d		Number of finite elements per slice
elem		Index of finite element
adapm_flags[<i>elem</i>]	%3d	Flag for element <i>elem</i> for adaptive meshing

NOTE: The levels and flags are written in 25 columns per line as blank separated values. First all nodal levels are saved. Then - beginning at a new line - the elemental flags follows. The order of nodes and elements follows the internal order.

2.2.38 AMR_METHOD

The options for defining the type of mesh adaption algorithm are defined by:

```

AMR_METHOD
adap_type
amr_error

```

The logical values are declared as:

Name	Format	Description
adap_type	%1d	Flag to indicate the algorithm used for adaptive meshing: 0 = Onate and Bugeda (1994) 1 = Zienkiewicz and Zhu (1987 - 1990)
amr_error	%13.6le	Error used in the adaptive mesh refinement

2.2.39 REFERENCE_TRAN

The options for the reference mass and reference temperature are defined by:

```
REFERENCE_TRAN
mass_ref_set, mass_ref_value, heat_ref_set, heat_ref_value,
mass_max_set, mass_max_value
LOOP over all chemical species
mass_ref_set mass_ref_value
```

The logical values are declared as:

Name	Format	Description
mass_ref_set	%1d	Boolean value to indicate if the default reference concentration has been overwritten: 0 = not overwritten 1 = overwritten
mass_ref_value	%13.6le	Reference concentration value (default: 0.0 - fresh water)
heat_ref_set	%1d	Boolean value to indicate if the default reference temperature has been overwritten: 0 = not overwritten 1 = overwritten
heat_ref_value	%13.6le	Reference temperature value (default: 150 °C)
mass_max_set	%1d	Boolean value to indicate if the default maximum concentration has been overwritten: 0 = not overwritten 1 = overwritten
mass_max_value	%13.6le	Maximum concentration value (default: 100 mg/l)

2.2.40 PHYSICS

The options influencing physical rules are stored in the following format:

```
PHYSICS
cond_well_bore, mass_matrices, process_boussinesq, process_viscosity,
fs_constraint.bot, flags.quadr, fs_constraint.top, max_itera,
dry_delta_head, flags.revff, delta_hysteresis, fs_constraint.bot_spec,
fs_constraint.top_spec, process_densratio
```

The logical values are declared as:

Name	Format	Description
cond_well_bore	%1d	Boolean value to indicated if multilayer wells should be used in 3D problems 0 = With multilayer wells (default) 1 = No multilayer wells
mass_matrices	%1d	Boolean flag to set kind of mass matrix computation: 0 = Consistent mass matrix computation (default) 1 = Lumped mass matrix computation
process_boussinesq	%1d	Boolean value to indicated the type of used Boussinesq approximation: 0 = default Boussinesq approximation

Name	Format	Description
		1 = extended Boussinesq approximation
process_viscosity	%1d	Boolean flag to indicate type of viscosity handling: 0 = Use viscosity dependencies 1 = Neglect viscosity dependencies
fs_constraint.bot	%1d	Flag to indicate bottom constraint: 0 = allow constraints at bottom 1 = avoid constraints at bottom
flags.quadr	%1d	Flag to identify quadrature rule: 0 = standard quadrature rule 1 = single-point (reduced) quadrature rule
fs_constraint.top	%1d	Flag to indicate top constraint: 0 = allow constraints at top 1 = avoid constraints at top
max_itera	%1d	Maximum number of iterations
dry_delta_head	%1d	Minimum hydraulic head above an element bottom location to regularize a dry element
flags.revff	%1d	Flag to identify reverse flow field for transport problems 0 = normal flow field 1 = reverse flow field
delta_hysteresis	%1d	Pressure head difference [m] to switch the reversal point for hysteresis conditions
fs_constraint.bot_spec	%1d	Flag to indicate special bottom constraint ('seepage face') 0 = no seepage face condition 1 = use seepage face condition
fs_constraint.top_spec	%1d	Flag to indicate special bottom constraint ('seepage face') 0 = no seepage face condition 1 = use seepage face condition
process_densratio	%1d	Flag for density-dependency for multi-species solute transport models 0 = Constant solutal density ratio (default) 1 = Multiple species-dependent density ratio

2.2.41 VISCOSITY

The viscosity section is saved in case of user-defined viscosity relationship.

VISCOSITY

equations specified in FEMATHED in XML format (format description see Online Help)

2.2.42 UNSAT OPS

Options for unsaturated/variably saturated flow conditions are stored in the following format:

UNSAT OPS

capa upw cps unwt form hyst eps_res

The logical values are declared as:

Name	Format	Description
capa	%1d	Evaluation option for unsaturated capacity term: 0 = chord slope scheme 1 = time-centred analytic derivative 2 = analytic derivative (fully implicit)
upw	%1d	Unsaturated parameter weighting: 0 = central weighting 1 = upstream weighting
cps	%1d	Unsaturated flow additional error bound checking:

		0 = don't check capillary head and saturation bounds 1 = check capillary head and saturation bounds 2 = omit residual checks for Newton or Picard method
unwt	%1d	Unsaturated flow iteration schemes: 0 = Picard scheme 1 = Newton scheme
form	%1d	Forms of the Richards equations used: 0 = standard h-based form 1 = mixed (s-h) form 2 = variable substitution form
hyst	%1d	Hysteresis flag: 0 = unset hysteresis 1 = set hysteresis
eps_res	%13.6le	Residual error tolerance for unsaturated problems using the mixed head-saturation form of the Richards equation

2.2.43 TVFDE

The TVFDE entry determines the parameters for thermal fluid density expansion in the following form:

```
TVFDE
tvfde.a[0] tvfde.a[1] tvfde.a[2] tvfde.a[3] tvfde.a[4] tvfde.a[5]
tvfde.a[6]
```

The parameters are as follows:

Name	Format	Description
tvfde.a[0]	%13.6le	1 st parameter for polynomial thermal fluid density expansion
tvfde.a[1]	%13.6le	2 nd parameter for polynomial thermal fluid density expansion
tvfde.a[2]	%13.6le	3 rd parameter for polynomial thermal fluid density expansion
tvfde.a[3]	%13.6le	4 th parameter for polynomial thermal fluid density expansion
tvfde.a[4]	%13.6le	5 th parameter for polynomial thermal fluid density expansion
tvfde.a[5]	%13.6le	6 th parameter for polynomial thermal fluid density expansion
tvfde.a[6]	%13.6le	7 th parameter for polynomial thermal fluid density expansion

2.2.44 TSTAGES

The times stages for the simulation output are saved beneath this flag. The syntax is:

```
TSTAGES
numb flag
for (i = 0; i < numb; i++) {
    level[i]
}
```

with:

Name	Format	Description
numb	%d	Number of time levels
flag	%d	True (1) indicates a time saving at these levels
level[i]	%13.6e	Time levels

2.2.45 MAP_PALETTE

Map palette saves custom settings for the color palette using the 256 color mode, e.g.:

```
name = Custom3
color1=(red=255, green=106, blue=106)
color2=(red=0, green=191, blue=255)
color3=(red=37, green=95, blue=255)
```

2.2.46 MAP_PATH

In each MAP_PATH entry the path and settings for a background map are saved, e.g:

```
path = ../import+export/geology.shp
type = shape
shape = polygons
xmin = 3404019.75
xmax = 3415117
ymin = 5811568.5
ymax = 5819865.5
time = Tue Mar 26 21:24:32 1996
id(2) = "ID #2"
[use=4,text_color=#ffad00,fill_color=#ffad00,fill_style=dither]
id(4) = "ID #4"
[use=4,text_color=#eb33a3,line_color=#0d0082,fill_color=#640000,
line_style=dot,line_width=4,fill_style=diag-forw]
id(5) = "ID #5"
[use=4,text_color=#bba600,fill_color=#bba600,fill_style=dither]
id(6) = "ID #6"
[use=4,text_color=#b7df86,fill_color=#b7df86,fill_style=dither]
```

2.2.47 LEGEND

In the LEGEND section the settings of the different legends are saved, e.g.:

```
Code = 400
Scaling = 3
Resolution = 2
RainbowUp = yes
CustomLegend "Custom2" 2
  1 3.178076744079590e+001 3.257216930389404e+001
  2 3.257216930389404e+001 3.336357116699219e+001
```

2.2.48 FRACTURES

The saving of the discrete feature elements is done by:

```
FRACTURES
fep.e, fmpf.e, fmpm.e, fmph.e
for (i=0; i < fep.e; i++){
  fep.frace[i].n, fep.frace[i].nbn, fep.frace[i].type,
  fep.frace[i].mode, fep.frace[i].law, fep.frace[i].area,
  fep.frace[i].slc[0], fep.frace[i].slc[1], fep.frace[i].slc[2],
  fep.frace[i].slc[3], fep.frace[i].nop[0], fep.frace[i].nop[1],
  fep.frace[i].nop[2], fep.frace[i].nop[3]
}
for (i=0; i < fmpf.e; i++){
  fmpf.fracm[i].v[FRAC_COND], fmpf.fracm[i].v[FRAC_STOR],
  fmpf.fracm[i].v[FRAC_COMP], fmpf.fracm[i].v[FRAC_SOUF],
  fmpf.fracm[i].v[FRAC_DENS], fmpf.fracm[i].v[FRAC_EXPA],
  fmpf.fracm[i].v[FRAC_TRAF_IN], fmpf.fracm[i].v[FRAC_TRAF_OUT]
}
for (i=0; i < fmpm.e; i++){
  fmpm.fracm[i].v[FRAC_POROM], fmpm.fracm[i].v[FRAC_SORP],
  fmpm.fracm[i].v[FRAC_DIFF], fmpm.fracm[i].v[FRAC_LDISM],
  fmpm.fracm[i].v[FRAC_TDISM], fmpm.fracm[i].v[FRAC_DECA],
  fmpm.fracm[i].v[FRAC_SOUT], fmpm.fracm[i].v[FRAC_TRAT_IN],
  fmpm.fracm[i].v[FRAC_TRAT_OUT]
}
for (i=0; i < fmph.e; i++){
```

```

fmph.fracm[i].v[FRAC_POROH], fmph.fracm[i].v[FRAC_CAPACF],
fmph.fracm[i].v[FRAC_CAPACS], fmph.fracm[i].v[FRAC_CONDUCTF],
fmph.fracm[i].v[FRAC_CONDUCS], fmph.fracm[i].v[FRAC_LDISH],
fmph.fracm[i].v[FRAC_TDISH], fmph.fracm[i].v[FRAC_SOURCF],
fmph.fracm[i].v[FRAC_SOURCS], fmph.fracm[i].v[FRAC_TRAH_IN],
fmph.fracm[i].v[FRAC_TRAH_OUT],
}

```

The logical values are declared as:

Name	Format	Description
fep.e	%d	Number of fracture elements
fmpf.e	%d	Number of fracture elements for flow
fmpm.e	%d	Number of fracture elements for mass transport
fmph.e	%d	Number of fracture elements for heat transport
n	%d	Related number (to identify fracture)
nbn	%d	Nodes per element (2, 3, 4)
type	%d	Type of fracture: 0 = One-dimensional (bar) fracture element type 1 = Two-dimensional triangular fracture element type 2 = Two-dimensional quadrilateral fracture
mode	%d	Mode of fracture element: 0 = Horizontal fracture selection mode 1 = Vertical fracture selection mode 2 = Arbitrary fracture selection mode
law	%d	Kind of flux law: 0 = Darcy flow equation 1 = Hagen-Poiseuille flow equation 2 = Manning-Strickler flow equation
area	%d	Geometric aperture or area of fracture element
slc[0]	%d	Slice number related to the node
slc[1]	%d	Slice number related to the node
slc[2]	%d	Slice number related to the node
slc[3]	%d	Slice number related to the node
nop[0]	%d	Nodal incidence
nop[1]	%d	Nodal incidence
nop[2]	%d	Nodal incidence
nop[3]	%d	Nodal incidence
v[FRAC_COND]	%13.6e	Conductivity material
v[FRAC_STOR]	%13.6e	Storativity material
v[FRAC_COMP]	%13.6e	Compressibility material
v[FRAC_SOUF]	%13.6e	Source/sink
v[FRAC_DENS]	%13.6e	Density ratio
v[FRAC_EXPA]	%13.6e	Thermal expansion coefficient
v[FRAC_TRAF_IN]	%13.6e	Flow transfer rate for inflow
v[FRAC_TRAF_OUT]	%13.6e	Flow transfer rate for outflow
v[FRAC_POROM]	%13.6e	Porosity for mass transport
v[FRAC_SORP]	%13.6e	Sorption coefficient
v[FRAC_DIFF]	%13.6e	Molecular diffusion
v[FRAC_LDISM]	%13.6e	Longitudinal dispersivity
v[FRAC_TDISM]	%13.6e	Transverse dispersivity
v[FRAC_DECA]	%13.6e	Decay rate
v[FRAC_SOUT]	%13.6e	Source/sink
v[FRAC_TRAT_IN]	%13.6e	Mass transfer rate for influx
v[FRAC_TRAT_OUT]	%13.6e	Mass transfer rate for outflux
v[FRAC_POROH]	%13.6e	Porosity for heat transport

Name	Format	Description
v[FRAC_CAPACF]	%13.6e	Volumetric heat capacity of fluid
v[FRAC_CAPACS]	%13.6e	Volumetric heat capacity of solid
v[FRAC_CONDUCTF]	%13.6e	Heat conductivity of fluid
v[FRAC_CONDUCS]	%13.6e	Heat conductivity of solid
v[FRAC_LDISH]	%13.6e	Longitudinal thermo-dispersivity
v[FRAC_TDISH]	%13.6e	Transverse thermo-dispersivity
v[FRAC_SOURCF]	%13.6e	Heat source/sink of fluid
v[FRAC_SOURCS]	%13.6e	Heat source/sink of solid
v[FRAC_TRAH_IN]	%13.6e	Heat transfer rate for influx
v[FRAC_TRAH_OUT]	%13.6e	Heat transfer rate for outflux

2.2.49 OPTIONS

Under the OPTIONS entry some basic settings of FEFLOW are stored. The listing could look like the following example:

```

OPTIONS
  SkipMeshFill=false
  SkipMeshDraw=true
  BackingStore=false
  OpaqueFringeMode=false
  LegendViewMode=false
  VelocityApproximationType=0
  EquationSolverType=12
  PreconditionSymmType=0
  IterNonsymmType=4
  MaxNumbOrthogonal=5
  GridDeltaX=100
  GridDeltaY=100
  RainbowType=0
  HideVelocity=false
  VelocityType=0

```

2.2.50 SYMSOLV

The SYMSOLV section contains information about the solver settings. The record is only necessary if settings differing from the default options are used.

```

case PCG:
default:
  if (isprop.pcg_prop.set) {
    fprintf (fp, "SYMSOLV\n");
    fprintf (fp, "%d\n", iterative_sym_method);
    fprintf (fp, "%d,%13.6le\n", isprop.pcg_prop.precond_type, isprop.pcg_prop.eps);
    fprintf (fp, "%d,%d,%d,%d,%d\n", isprop.pcg_prop.mit[0], isprop.pcg_prop.mit[1],
            isprop.pcg_prop.mit[2], isprop.pcg_prop.mit[3], isprop.pcg_prop.mit[4]);
  }
  break;
case AMG:
  if (isprop.amg_prop.set) {
    fprintf (fp, "SYMSOLV\n");
    fprintf (fp, "%d\n", iterative_sym_method);
    fprintf (fp, "%13.6le,%d,%d,%d,%d,%d\n", isprop.amg_prop.eps,
            isprop.amg_prop.ncyc, isprop.amg_prop.n_default,
            isprop.amg_prop.quiet, isprop.amg_prop.coarsening_type,
            isprop.amg_prop.matrix_posedness, isprop.amg_prop.n_default_set);
    fprintf (fp, " %s\n", SSTR(isprop.amg_prop.dump_path));
  }

```

Name	Format	Description
pcg_prop.set	%d	Flag: PCG properties differ from default
iterative_sym_method	%d	Solver setting: 0 = PCG 1 = SAMG
pcg_prop.precond_type	%d	Preconditioning type: 0 = Incomplete factorization IF (default) 1 = GUSTAFSSONS modified incomplete factorization MIF
pcg_prop.mit[i]	%d	Maximum iteration numbers (for 5 intervals i)
pcg_prop.eps	%13.6 le	Termination criterion to stop PCG iteration
amg_prop.set	%d	Flag: SAMG properties differ from default
amg_prop.eps	%13.6 le	Termination criterion to stop SAMG iteration
amg_prop.ncyc	%d	Cycling and acceleration strategies
amg_prop.n_default	%d	Secondary parameters for general control switch
amg_prop.quiet	%d	Suppress all print output: 0 = print output 1 = quiet (suppress output)
amg_prop.coarsening_type	%d	Coarsening strategy: 0 = standard 1 = aggressive
amg_prop.matrix_posedness	%d	Matrix properties: 0 = ill posed 1 = well posed
amg_prop.n_default_set	%d	General control switch: 0 = not activated 1 = activated
amg_prop.dump_path	%s	Path for dumping matrix data (if specified)

2.2.51 MODULE

The MODULE section carries data of an IFM module in a syntax defined by developer of the module.

2.2.52 MOD_INFO

MOD_INFO lists information about the activated module(s) and determines whether a module is active or not, e.g:

```
MOD_INFO PEST 1.03 SIMULATION
  Enabled=true
```

2.2.53 END

To mark the end of the finite element problem data include:

```
END
```

3 FEFLOW Results File Format (*.dac) - Version 5.4

3.1 General structure of the file

The FEFLOW complete results ASCII files with the filename extension *.dac (data complete) has a general structure as explained in the table below. The results file consists of two introducing lines and following data blocks separated by marker lines. The first data block contains the data of the FEFLOW model in the same format as the „fem“ file of the FEFLOW problem.

For every saved time step follows one data block containing the complete FEFLOW results. Every results data block begins with a characteristic marker line containing the time step index and the time level of the results.

The number of data in a results data block depends on the problem class, the slice modes and the problem dimension (see chapter 2).

Block No.	Description	Markerline
1	Problem title(at line 1)	
2	Creation date and time of the file (at line 2)	
3	FEFLOW model data (see chapter 2)	\$ -1
4 – n-1	Result data for time step i at time TIME	\$ i, TIME
n	Diagram data	DIAGRAM

The marker lines define begin and end of the data blocks 3, 4, A marker line is defined by the dollar character „\$“ at position 1 of the line.

The physical units of all saved data are the native FEFLOW units. Please refer to the Reference Manual for the used physical units.

3.2 Description of the formats

3.2.1 Problem title (line 1)

This problem title description consists of only one line. All characters of this line are the text of the problem description.

Example:

```
+++++++ FEFLOW computational results ++++++
```

3.2.2 Creation date (line 2)

Line 2 contains the date and time of creation of the results file in the UNIX typical syntax.

Example:

```
Mon Sep 30 17:34:26 1996
```

3.2.3 FEFLOW model data block format

This data block consists of one marker line and the complete FEM model data in the same format as in the FEFLOW model file (*.fem).

```
$ -1 : Basic data
model_data_as_in_the_fem_file
```

3.2.4 Results data block format

3.2.4.1 General Structure of the data block

The following lines describe the results data block structure. The format description syntax corresponds to the programming language C:

```
$ i, time
head_data
```

```

vx_data
vy_data
if (is_3D_problem) {
    vz_data
}
if (is_mass_transport or is_thermohaline_transport) {
    concentration_data
}
else if (is_heat_transport) {
    temperature_data
}
if (is_thermohaline_transport) {
    temperature_data
}
if (is_3D_problem and is_unconfined and is_slicel_free&movable) {
    for (slc=1; slc <= numb_slices; slc++) {
        if (slicetype[slc] == UNSPECIFIED or slicetype[slc] == MOVABLE) {
            slice_elevation_data
        }
    }
}
if (ic2 != 2) {
    acceleration_data_mass
    if (ic1 != 2) {
        if (ic2 == 1) {
            acceleration_data_flow
        }
        if (ic1 == 8) {
            acceleration_data_heat
        }
    }
}
if (exist_cbc_of_flow) {
    flow_cbc_data
}
if (exist_cbc_of_mass) {
    mass_cbc_data
}
if (exist_cbc_of_heat) {
    heat_cbc_data
}

```

Description of the logical values:

Name	Format	Description
i	%5d	Index of current timestep, beginning with 0
time	%13.6le	time level value of the current timestep
np		Number of mesh nodes of the model. Sum of all slices.
head_data	table12	Computed hydraulic head at each mesh node. Dimension: np
vx_data	table12	Computed x component of the velocity vector of all mesh nodes Dimension: np
vy_data	table12	Computed y component of the velocity vector of all mesh nodes Dimension: np
is 3D problem		Boolean value indicating the problem dimension: is_3D_problem = False for 2D models is_3D_problem = True for 3D models
vz_data	table12	Computed z component of the velocity vector of all mesh

		nodes, Dimension: np
is mass transport		Boolean value depending on the problem class: True: for mass transport False: otherwise
is thermohaline transport		Boolean value depending on the problem class: True: for thermohaline transport False: otherwise
is heat transport		Boolean value depending on the problem class: True: for heat transport False: otherwise
concentration_data	table12	Computed concentration at the mesh nodes. Dimension: np
temperature_data	table12	Computed temperature at the mesh nodes. Dimension: np
is slice1 free&movable		Boolean value depending on the problem class: True: for unconfined problem with slice 1 of type „free&movable“ False: for type „phreatic“ or "fixed“ of slice1
is unconfined		Boolean value depending on the problem class: True: for unconfined problem class False: for confined problem class
slc		Index of slices in the loop
numb slices		Number of slices of the 3D problem
slicetype[slc]		Value to define the type of the slice <i>slc</i> . Possible values are: FIXED, UNSPECIFIED, MOVABLE, PHREATIC
slice_elevation_data	table12	Z coordinates (elevations) for all points of a slice. Dimension: number of points per slice
ic1		Problem class identifier, see description of <i>fem</i> file, statement DIMENS
ic2		Problem class identifier, see description of <i>fem</i> file, statement DIMENS
acceleration_data_flow	table12	Acceleration vector for correct budget results of flow (time derivative). Dimension: np
acceleration_data_mass	table12	Acceleration vector for correct budget results of mass transport (time derivative). Dimension: np
acceleration_data_heat	table12	Acceleration vector for correct budget results of heat transport (time derivative). Dimension: np
exist cbc of flow		Boolean value indicating existence of constraint boundary conditions for flow boundaries. True: constraint BCs exist False: constraint BCs don't exist
exist cbc of mass		Boolean value indicating existence of constraint boundary conditions for mass boundaries. True: constraint BCs exist False: constraint BCs don't exist
exist cbc of heat		Boolean value indicating existence of constraint boundary conditions for heat boundaries. True: constraint BCs exist False: constraint BCs don't exist
flow_cbc_data	bc_list	Data of flow boundary conditions with underlying constraints active at the current time
mass_cbc_data	bc_list	Data of flow boundary conditions with underlying constraints active at the current time
heat_cbc_data	bc_list	Data of flow boundary conditions with underlying constraints active at the current time

3.2.4.2 Description of the formats used for the results data

3.2.4.2.1 Format table12

The format „table12“ uses values line by line with 12 columns per line. The number of values to store depends on the data type and is marked as the dimension of these data array in table 2. In the following format description it is written as variable `dim`. The data to store are contained in the array `value`.

```

for (line =1, node=1; line <= (dim-1)/12 + 1; line++) {
  for (col=1; col <=12; col++,node++) {
    if (node <= dim)
      value[node]
  }
  <newline>
}

```

The logical variables are:

Name	Format	Description
<code>line</code>		Index of lines
<code>col</code>		Index of columns
<code>dim</code>		Number of values to store
<code>node</code>		Index of the node in array <code>value</code>
<code>value[node]</code>	<code>%21.14le</code>	Value of node with index <code>node</code>
<code><newline></code>		The newline character

3.2.4.2.2 Format bc_list

The constraint boundary conditions at the current time state will be stored sorted by the boundary condition type.

For every type mesh node indices carrying this boundary type will be listed. This is a format similar to the format used in the fem file for boundary conditions - but without keyword.

The boundary conditions are stored in the following format:

```

for (bc_type = MIN_bc_type; bc_type <= MAX_bc_type; bc_type++) {
  if (boundary_conditions_with_this_bc_type_exist) {
    bc_type bc_value_list
    bc_node_list
  }
}

```

The logical variables are:

Name	Format	Description
<code>bc_type</code>	<code>%8x</code>	Hexadecimal code of the boundary condition type including the constraints informations
<code>bc_value_list</code>	<code>n(%21.14le)</code>	n Values describing the boundary condition and constraints: The first value is the original boundary condition value or the time function index used for the BC. The following values are the constraint values in the same order they appear in the FEFLOW menus from top to bottom. These constraints can also be values or time function lds.
<code>bc_node_list</code>	<code>node_list</code>	All nodes indices carrying the current <code>bc_type</code> and <code>bc_value_list</code> . May be one or more node indices.

3.2.4.2.3 Format node list

A node list consists of blank separated entries of two types:

- single node entry: a single *node_index*
- node range entry: A range of nodes (from startnode to endnode) will be stored as: *node_index-node_index*.

The first number is the startnode index.

The second node value is the endnode index.

Directly after the first node value has to follow a „minus“ sign.

node_index: A right bound integer value.

FEFLOW writes the number of digits is always the same for all node indices to get a good readable design. Its not mandatory.

The number of columns of *node_values* in one line depends on the number of digits of the node values. The rule is: $\text{columns} = 80 / (\text{digits} + 1)$.

The node list begins after two tabs at the beginning of the line.

Example:

```
8 3.210000e+01
128- 140 340- 345 413- 416 425- 427 592 596 602 603 608 609 1031 1034
1037 1040 1043 1047 1615 1684 1702 1777 1781 1791 1794 1797 1800 1803
1806 1810
1812 1813 1817 1821 1823 1829 1832 1840
10 -7.500000e-02
12- 14 64- 68 243- 246 267 268
```

3.2.4.3 Decoding / encoding the bc_type

See the corresponding section of the *.fem file description for detailed information about the decoding and encoding of the bc and bcc type.

3.2.5 Diagram data

The last block of data in the file saves all diagram data. The block is overwritten in each time step by the latest version. It always covers all time steps of the simulation, not only the ones saved in the dac file.

```
DIAGRAMS
N_WIND, obs_pnts, numb_wells, cdiagnum, ntime
cwin_mask
cwin_permut
cwin_entr
tini0,tini1,tini2,tini3,tini4,tini5,
tini6, ..., tinin
if (logres) {
    name
    res0,res1,res2,res3,res4,res5,
    res6, ..., resn
}
for (i = 1; i < N_WIND; i++) {
    if ((k = cwin_permut[i]) > -1) {
        for (j = 0; j < cwin_entr[i]; j++) {
            name
            value0,value1,value2,value3,value4,value5,
            value6, ..., valuen
        }
    }
}
```

```

}
te_skip1 te_skip2 ...
for (i = 0; i < numb_wells; i++){
    av_well_h,av_well_m,av_well_t,
}

```

Name	Format	Description
N_WIND	%d	Number of possible diagram windows
obs_pnts	%d	Number of observation points
numb_wells	%d	Number of wells and indicated boundary conditions
cdiagnum	%d	Number of current diagram windows 1, ..., N_WIND-1
ntime	%d	Current number of time steps used in the diagram
cwin_mask	%d,%d,%d,%d, %d,%d,%d,%d, %d	Mask for current windows to be displayed, e. g.: 1,1,0,0,0,0,1,0,0 for flow, 1,1,1,1,0,0,1,1,0 for mass transport, 1,1,0,0,1,1,1,0,1 for heat transport, 1,1,1,1,1,1,1,1,1 for thermohaline transport
cwin_permut	%d,%d,%d,%d, %d,%d,%d,%d, %d	Diagram window permutation vector
cwin_entr	%d,%d,%d,%d, %d,%d,%d,%d, %d	Diagram window entry numbers for each window ID, e.g.: 0 = obs+numb_wells, 1 = numb_wells, 2 = obs, ...
tini0, ..., tinin	%21.14e	Time steps for diagrams
name	%s	Observation point, well, ... name
logres	%d	Residuum diagram flag for unsaturated flow 0 = no residuum diagram 1 = residuum diagram
res0, ..., resn	%13.6le	Residuum values for unsaturated flow
value0, ..., valuen	%13.6le	Diagram values
te_skip1, ..., te_skipn	%d	Skipped data points in diagrams, 20 values per line: 0 = not skipped 1 = skipped
av_well_h	%21.14e	Head at a well i at the last time step
av_well_m	%21.14e	Concentration at a well i at the last time step
av_well_t	%21.14e	Temperature at a well i at the last time step