EPANET

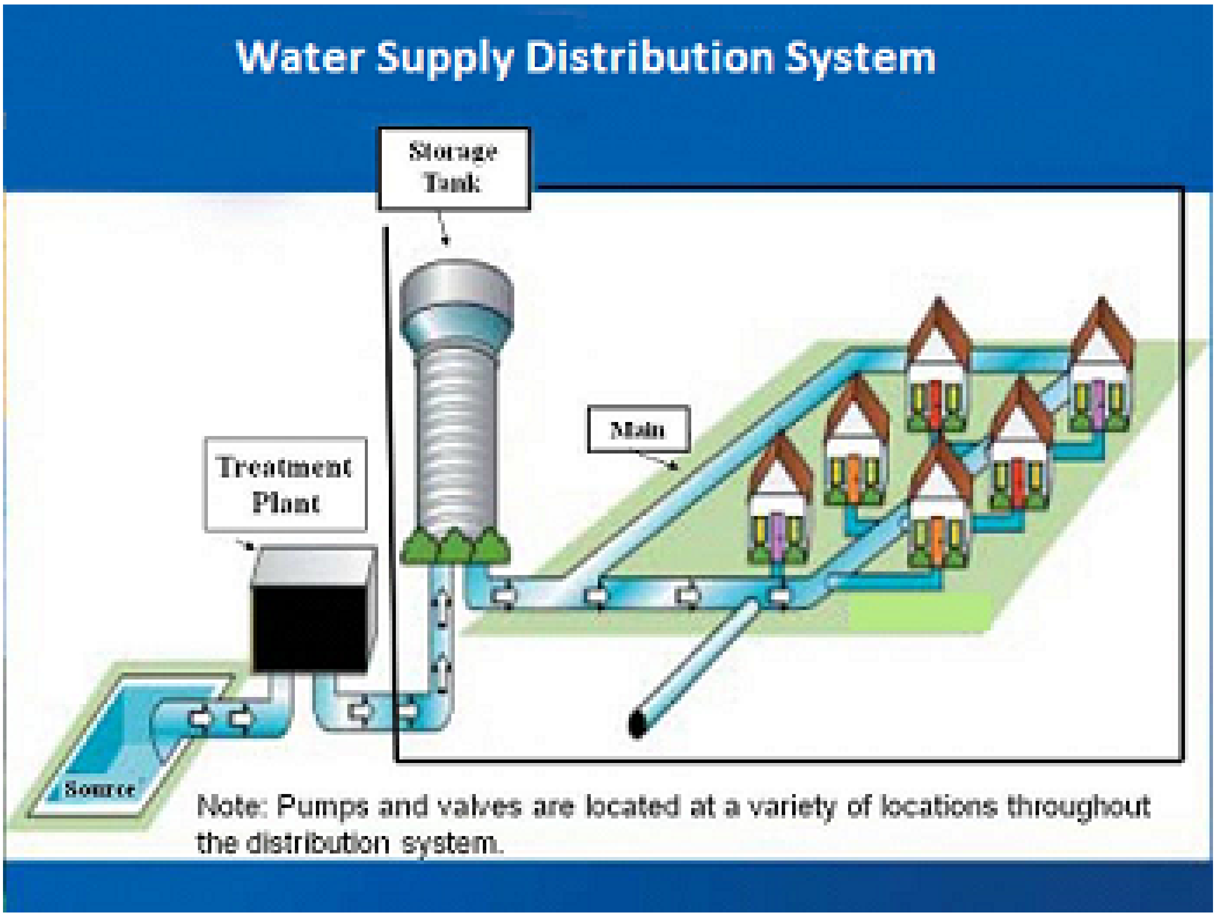
2.2.0

Generated by Doxygen 1.8.16

Chapter 1

Overview

EPANET is a program that performs extended period simulation of hydraulic and water quality behavior within water distribution system pipe networks. A network can consist of pipes, nodes (pipe junctions), pumps, valves and storage tanks or reservoirs. EPANET tracks the flow of water in each pipe, the pressure at each node, the height of water in each tank, and the concentration of a chemical species throughout the network during a multi-time period simulation. In addition to chemical species, water age and source tracing can also be simulated.



The EPANET Programmer's Toolkit is a library of functions (or API) written in C that allow programmers to customize the use of EPANET's hydraulic and water quality solution engine to their own applications. Both EPANET and its toolkit were originally developed by the U.S. Environmental Protection Agency (USEPA).

The OWA-EPANET Toolkit is an open-source version of the original EPANET Toolkit that extends its capabilities by:

- providing a full set of functions to set and retrieve values for all parameters contained in a network model
- allowing networks to be built completely from function calls instead of from an input file
- allowing multiple projects to be analyzed in parallel in a thread-safe manner
- adding the ability to use pressure dependent demands in hydraulic analyses
- producing more robust results with regard to hydraulic convergence, low/zero flow conditions, and water quality mass balance
- achieving faster run times for single period hydraulic analyses.

Before using the OWA-EPANET Toolkit one should be familiar with the way that EPANET represents a pipe network, the design and operating information it requires, and the steps it uses to simulate a network's behavior. The following topics provide some introductory material on these subjects:

- Network Data Model
- Data Flow Diagram
- Toolkit Versions

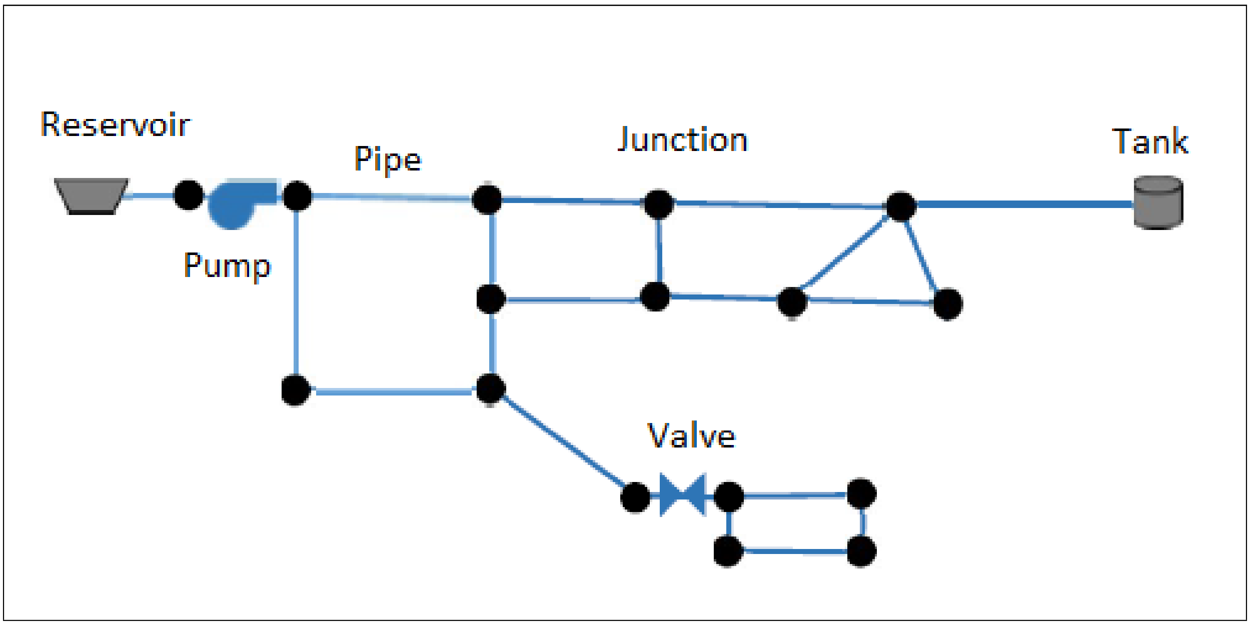
More detailed information can be obtained from reading the `EPANET 2 Users Manual`.

Note: `OWA (Open Water Analytics)` exists on GitHub as an open community for the exchange of information and ideas related to computing in the water & wastewater industries. It's activities and code projects are neither affiliated with nor endorsed by the USEPA.

Chapter 2

Network Data Model

EPANET models a pipe network as a collection of links connected to nodes. The links represent pipes, pumps, and control valves. The nodes represent junctions, tanks, and reservoirs. The figure below illustrates how these objects can be connected to one another to form a network.



Junctions have a user-supplied water withdrawal rate (i.e., consumer demand) associated with them. Tanks are storage units whose water level changes over time. Reservoirs are boundary points where a fixed hydraulic head applies.

Pipes have a length, diameter and roughness coefficient that determines their head loss as a function of flow rate. Pumps have either a constant power rating or a head curve that determines the head they add as a function of flow rate. Valves are used to regulate either flow or pressure. Controls can be applied to completely open or close a link or to adjust its setting (pump speed or valve setting).

In addition to these physical objects an EPANET model can also contain the following data objects:

- time patterns that allow demands, quality source strength and pump speed settings to vary at fixed intervals of time
- data curves that describe relationships between two quantities, such as head versus flow for pumps and volume versus water level for tanks

- simple controls that adjust a link's setting (such as a pump's status) based on node pressure, tank level, elapsed time, ot time of day
- rule-based controls that consist of one or more premises that if true result in one set of actions being taken and if false result in a different set of actions being taken
- water quality sources that introduce a chemical constituent into the network at specified nodes.

An EPANET model also contains a number of analysis options that specify:

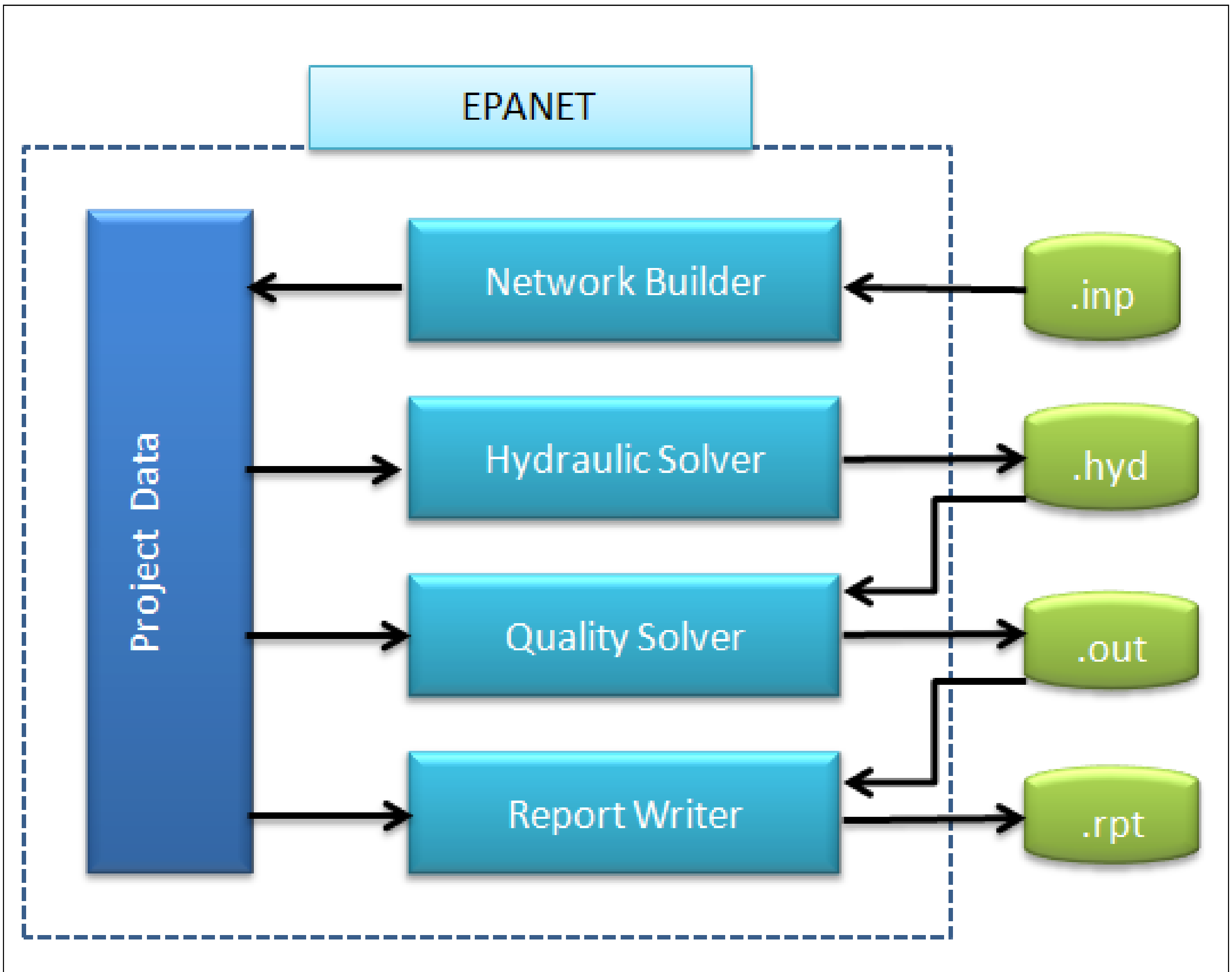
- the project's flow units which in turn determines its unit system (US or SI)
- the formula used to compute head loss
- whether to use a demand driven or a pressure driven analysis
- hydraulic convergence criteria
- time steps used for hydraulic, water quality and reporting
- the type of water quality analysis to perform (chemical reaction, source tracing or water age)
- global values for chemical reaction coefficients that can be overridden for individual pipes
- global values for energy usage parameters that can be overridden for individual pumps.

Please refer to the `EPANET 2 Users Manual` for more information on EPANET's data model.

Chapter 3

Data Flow Diagram

The EPANET Toolkit contains separate code modules for network building, hydraulic analysis, water quality analysis, and report generation. The data flow diagram for analyzing a pipe network is shown below. The processing steps depicted in this diagram can be summarized as follows:



- The network builder receives a description of the network being simulated either from an external input file (.inp) or from a series of function calls that create network objects and assign their properties via code. These data are stored in a Project data structure.

- The hydraulics solver carries out an extended period hydraulic simulation. The results obtained at every time step can be written to an external, unformatted (binary) hydraulics file (.hyd). Some of these time steps might represent intermediate points in time where system conditions change because of tanks becoming full or empty or pumps turning on or off due to level controls or timed operation.

- If a water quality simulation is requested, the water quality solver accesses the flow data from the hydraulics file as it computes substance transport and reaction throughout the network over each hydraulic time step. During this process it can write both the formerly computed hydraulic results as well as its water quality results for each preset reporting interval to an unformatted (binary) output file (.out). If no water quality analysis was called for, then the hydraulic results stored in the .hyd file can simply be written out to the binary output file at uniform reporting intervals.

- If requested, a report writer reads back the computed simulation results from the binary output file (.out) for each reporting period and writes out selected values to a formatted report file (.rpt). Any error or warning messages generated during the run are also written to this file.

Toolkit functions exist to carry out all of these steps under the programmer's control, including the ability to read and modify the contents of the Project data structure.

Chapter 4

Toolkit Versions

The Toolkit comes with two sets of identical functions that programmers can utilize:

- the single-threaded version of the Toolkit is compatible with previous releases and only works with single threaded applications.
- the multi-threaded version allows users to create multiple EPANET data sets (called projects) that can be analyzed concurrently.

Both Toolkit versions utilize identical function names and argument lists with the following exceptions:

- The `#include "epanet2.h"` directive must appear in all C/C++ code modules that use the single-threaded library while `#include "epanet2_2.h"` must be used for the multi-threaded library.
- Function names in the single-threaded library begin with **EN** while those in the multi-threaded library begin with **EN_** .
- The multi-threaded functions contain an additional argument that references a particular network project that the function is applied to.
- The multi-threaded library contains two additional functions that allow users to create and delete EPANET projects.
- The single-threaded library uses single precision for its floating point arguments while the multi-threaded library uses double precision.

To avoid unnecessary duplication this document only discusses the multi-threaded version of the Toolkit.

Chapter 5

Generating Documentation for OWA-EPANET 2.2

You must have `Doxygen` installed on your machine to generate documentation for the OWA-EPANET Toolkit. Assuming this is the case, open a terminal window, navigate to the project's `doc` directory and issue the command `doxygen`. This will generate HTML documentation placed in a sub-directory named `html`. From that directory you can launch the `index.html` file to view the full documentation in a web browser.

To generate a Windows compiled HTML Help file you must have `Microsoft's HTML Help Workshop` installed. You then need to edit the Doxygen configuration file `doxyfile` as follows:

1. Change the `GENERATE_HTMLHELP` setting to `YES`.
2. Enter the location where the Help Workshop system was installed next to the `HHC_LOCATION` setting.

After running Doxygen again the resulting Help file named `owa-epanet.chm` will appear in the `html` sub-directory.

Doxygen uses the special comments placed in the project's `epanet2_2.h` and `epanet2_enums.h` header files to document EPANET's API. It also uses supplementary material contained in the following files of the project's `doc` directory to generate additional pages of documentation:

- `main.dox`: generates the *Overview* section.
- `usage.dox`: generates the *Usage* section.
- `toolkit-examples.dox` : generates the *Examples* section.
- `toolkit-files.dox`: generates the *Toolkit Files* section.
- `input-file.dox`: generates the *Input File* sub-section.
- `toolkit-units.dox`: generates the *Measurement Units* section.
- `modules.dox`: defines the contents of the *API Reference* section.

Finally, a group of special Doxygen files are used to customize the format of the generated documentation. These include the following:

- `doxyfile`: the main Doxygen configuration file
- `DoxygenLayout.xml`: sets the title of the automatically generated *Modules* section to *API Reference* and hides the *Files* section in the tree view pane of the document.
- `extrastylesheet.css`: reduces the size of the the h1 heading style.
- `newfooter.html`: replaces the default Doxygen footer in HTML output with a custom one.

Chapter 6

Examples

Here are several examples of how the Toolkit can be used for different types of network analyses.

- Embedded Engine Example
- Network Building Example
- Hydrant Rating Curve Example
- Chlorine Dosage Example

6.1 Embedded Engine Example

This example shows how simple it is for the Toolkit to provide a network analysis engine for other applications. There are three steps that the application would need to take:

1. Have the application write network data to an EPANET-formatted input file.
2. Create a project and call EN_runproject, supplying the name of the EPANET input file, the name of a Report file where status and error messages are written, and the name of a binary Output file which will contain analysis results.
3. Have the application access the output file to display desired analysis results (see Output File).

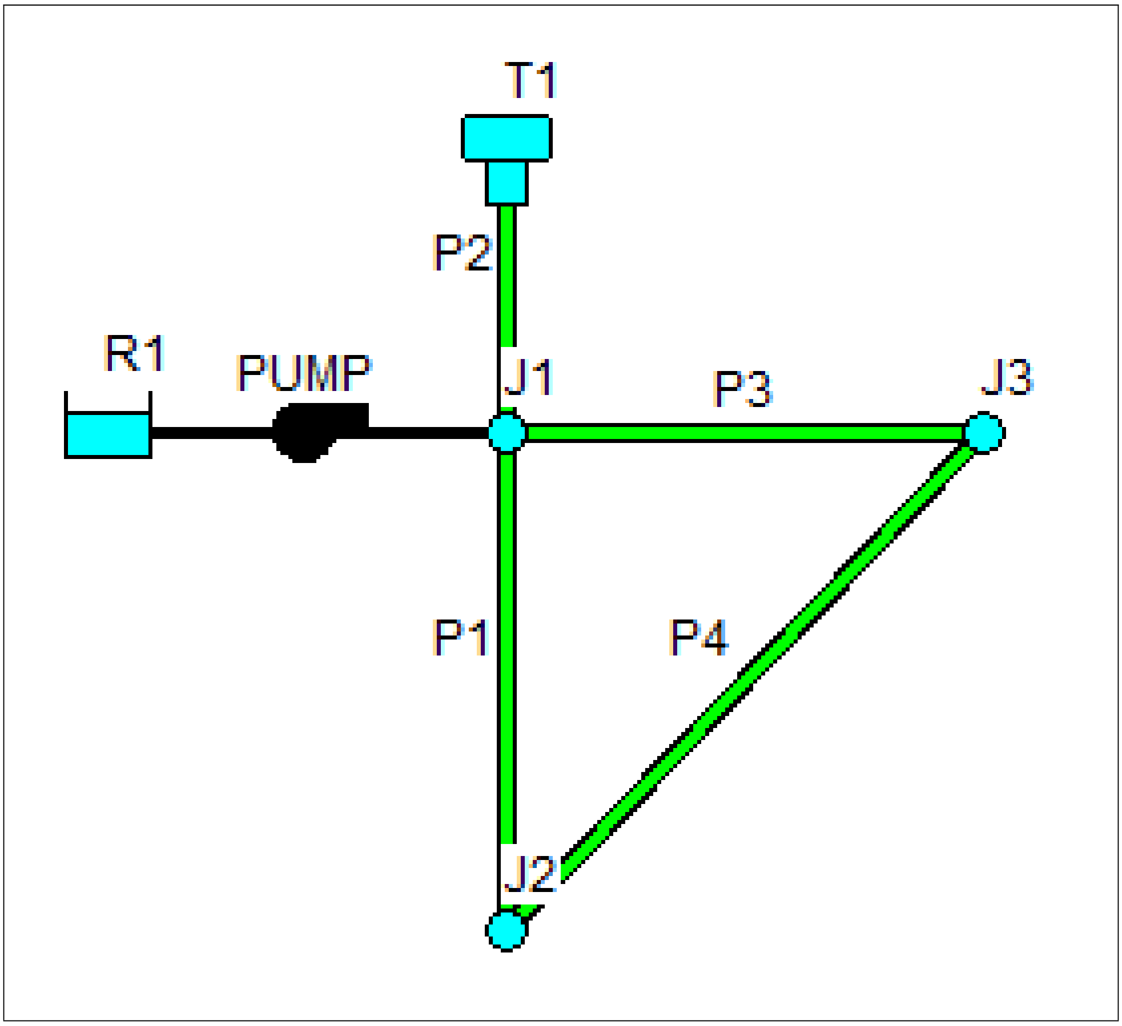
Here is an example where a callback function `writeConsole` is provided to write EPANET's progress messages to the console:

```
#include "epanet2_2.h"
void  writeConsole(char *s)
{
    fprintf(stdout, "\n%s", s);
}
int runEpanet(char* inpFile, char* rptFile, char* outFile)
{
    int errcode;
    EN_project ph;
    EN_createproject(&pH);
    errcode = EN_runproject(ph, inpFile, rptFile, outFile, &writeConsole);
    EN_deleteproject(ph);
    return errcode;
}
```

6.2 Network Building Example

This example shows how a network can be built just through toolkit function calls, eliminating the need to always use an EPANET formatted input file. This creates opportunities to use other sources of network data in one's code, such as relational database files or GIS/CAD files.

Below is a schematic of the network to be built.




```
#include "epanet2_2.h"
void netbuilder()
{
    // Create a project that uses gpm for flow units and
    // the Hazen-Williams formula for head loss
    int index;
    EN_Project ph;
    EN_createproject(&ph);
    EN_init(ph, "", "", EN_GPM, EN_HW);
    // Add the first junction node to the project with
    // an elevation of 700 ft and a demand of 0
    EN_addnode(ph, "J1", EN_JUNCTION, &index);
    EN_setjuncdata(ph, index, 700, 0, "");
    // Add the remaining two junctions with elevations of
    //  710 ft and demands of 250 and 500 gpm, respectively
    EN_addnode(ph, "J2", EN_JUNCTION, &index);
    EN_setjuncdata(ph, index, 710, 250, "");
    EN_addnode(ph, "J3", EN_JUNCTION, &index);
    EN_setjuncdata(ph, index, 710, 500, "");
    // Add the reservoir at an elevation of 650 ft
    EN_addnode(ph, "R1", EN_RESERVOIR, &index);
    EN_setnodevalue(ph, index, EN_ELEVATION, 650);
    // Add the tank node at elevation of 850 ft, initial water level
    // at 120 ft, minimum level at 100 ft, maximum level at 150 ft
    // and a diameter of 50.5 ft
    EN_addnode(ph, "T1", EN_TANK, &index);
    EN_settankdata(ph, index, 850, 120, 100, 150, 50.5, 0, "");
    // Add the pipes to the project, setting their length,
    // diameter, and roughness values
    EN_addlink(ph, "P1", EN_PIPE, "J1", "J2", &index);
    EN_setpipedata(ph, index, 10560, 12, 100, 0);
    EN_addlink(ph, "P2", EN_PIPE, "J1", "T1", &index);
    EN_setpipedata(ph, index, 5280, 14, 100, 0);
    EN_addlink(ph, "P3", EN_PIPE, "J1", "J3", &index);
    EN_setpipedata(ph, index, 5280, 14, 100, 0);
    EN_addlink(ph, "P4", EN_PIPE, "J2", "J3", &index);
    EN_setpipedata(ph, index, 5280, 14, 100, 0);
```

```
    // Add a pump to the project
    EN_addlink(ph, "PUMP", EN_PUMP, "R1", "J1", &index);
    // Create a single point head curve (index = 1) and
    // assign it to the pump
    EN_addcurve(ph, "C1");
    EN_setcurvevalue(ph, 1, 1, 1500, 250);
    EN_setlinkvalue(ph, index, EN_PUMP_HCURVE, 1);
    // Save the project for future use
    EN_saveinpfile(ph, "example2.inp");
    // Delete the project
    EN_deleteproject(ph);
}
```

6.3 Hydrant Rating Curve Example

This example illustrates how the Toolkit could be used to develop a hydrant rating curve used in fire flow studies. This curve shows the amount of flow available at a node in the system as a function of pressure. The curve is generated by running a number of steady state hydraulic analyses with the node of interest subjected to a different demand in each analysis. For this example we assume that the ID label of the node of interest is `MyNode` and that `N` different demand levels stored in the array `D` need to be examined. The corresponding pressures will be stored in `P`. To keep the code more readable, no error checking is made on the results returned from the Toolkit function calls.

```
#include "epanet2_2.h"
void HydrantRating(char *MyNode, int N, double D[], double P[])
{
    EN_Project ph;
    int   i, nodeindex;
    long  t;
    double pressure;
    // Create a project
    EN_createproject(&ph);
    // Retrieve network data from an input file
    EN_open(ph, "example2.inp", "example2.rpt", "");
    // Open the hydraulic solver
    EN_openH(ph);
    // Get the index of the node of interest
    EN_getnodeindex(ph, MyNode, &nodeindex);
    // Iterate over all demands
    for (i=1; i<N; i++)
    {
        // Set nodal demand, initialize hydraulics, make a
        // single period run, and retrieve pressure
        EN_setnodevalue(ph, nodeindex, EN_BASEDEMAND, D[i]);
        EN_initH(ph, 0);
        EN_runH(ph, &t);
        EN_getnodevalue(ph, nodeindex, EN_PRESSURE, &pressure);
        P[i] = pressure;
    }
    // Close hydraulics solver & delete the project
    EN_closeH(ph);
    EN_deleteproject(ph);
}
```

6.4 Chlorine Dosage Example

This example illustrates how the Toolkit could be used to determine the lowest dose of chlorine applied at the entrance to a distribution system needed to ensure that a minimum residual is met throughout the system. We assume that the EPANET input file contains the proper set of kinetic coefficients that describe the rate at which chlorine will decay in the system being studied. In the example code, the ID label of the source node is contained in `SourceID`, the minimum residual target is given by `Ctarget`, and the target is only checked after a start-up duration of 5 days (432,000 seconds). To keep the code more readable, no error checking is made on the results returned from the Toolkit function calls.

```
#include "epanet2_2.h"
double cl2dose(char *SourceID, double Ctarget)
{
    int i, nnodes, sourceindex, violation;
    double c, csource;
    long t, tstep;
    EN_Project ph;
```

```
    // Open the toolkit & obtain a hydraulic solution
    EN_createproject(&ph);
    EN_open(ph, "example3.inp", "example3.rpt", "");
    EN_solveH(ph);
    // Get the number of nodes and the source node's index
    EN_getcount(ph, EN_NODECOUNT, &nnodes);
    EN_getnodeindex(ph, SourceID, &sourceindex);
    // Setup the system to analyze for chlorine
    // (in case it was not done in the input file)
    EN_setqualtype(ph, EN_CHEM, "Chlorine", "mg/L", "");
    // Open the water quality solver
    EN_openQ(ph);
    // Begin the search for the source concentration
    csource = 0.0;
    do {
        // Update source concentration to next level
        csource = csource + 0.1;
        EN_setnodevalue(ph, sourceindex, EN_SOURCEQUAL, csource);
        // Run WQ simulation checking for target violations
        violation = 0;
        EN_initQ(ph, 0);
        do {
            EN_runQ(ph, &t);
            if (t > 432000) {
                for (i=1; i<=nnodes; i++) {
                    EN_getnodevalue(ph, i, EN_QUALITY, &c);
                    if (c < Ctarget) {
                        violation = 1;
                        break;
                    }
                }
            }
            EN_nextQ(ph, &tstep);
        // End WQ run if violation found
        } while (!violation && tstep > 0);
    // Continue search if violation found
    } while (violation && csource <= 4.0);
    // Close up the WQ solver and delete the project
    EN_closeQ(ph);
    EN_deleteproject(ph);
    return csource;
}
```

Chapter 7

Toolkit Files

The Toolkit can make use of several different types of files when analyzing a pipe network. These include:

- Input File
- Report File
- Output File
- Hydraulics File
- Header Files

7.1 Input File

The Input file is a standard EPANET input data file that describes the system being analyzed. It can either be created external to the application being developed with the Toolkit or by the application itself. It is the first file name supplied to the EN_open function. A project's data associated with its Input file remains accessible until the project is closed down with the EN_close or deleted with EN_deleteproject.

The file is organized by sections where each section begins with a keyword enclosed in brackets. The various keywords are listed below. Click on a section to see the format of the data it contains.

Network Components	System Operation	Water Quality	Options & Reporting	GUI Support
[Title]	[Curves]	[Quality]	[Options]	[Backdrop]
[Junctions]	[Patterns]	[Reactions]	[Times]	[Coordinates]
[Reservoirs]	[Energy]	[Sources]	[Report]	[Vertices]
[Tanks]	[Status]	[Mixing]		[Labels]
[Pipes]	[Controls]			
[Pumps]	[Rules]			
[Valves]	[Demands]			
[Emitters]				

The order of sections is not important. However, whenever a node or link is referred to in a section it must have already been defined in the [JUNCTIONS], [RESERVOIRS], [TANKS], [PIPES], [PUMPS], or [VALVES] sections. Thus it is recommended that these sections be placed first.

Each section can contain one or more lines of data. Blank lines can appear anywhere in the file and the semicolon (;) can be used to indicate that what follows on the line is a comment, not data. A maximum of 1024 characters can appear on a line.

The ID labels used to identify nodes, links, curves and patterns can be any combination of up to 31 characters and numbers.

The GUI Support sections are provided to assist an external program that wishes to draw a visual representation of a project's network.

7.1.1 [TITLE]

Purpose:

Attaches a descriptive title to the network being analyzed.

Format:

Any number of lines of text.

Remarks:

The **[TITLE]** section is optional.

7.1.2 [CURVES]

Purpose:

Defines data curves and their X,Y points.

Format:

One line for each X,Y point on each curve containing:

- Curve ID label
- an X value
- a Y value

Remarks:

1. Curves can be used to represent the following relations:
   - Head v. Flow for pumps
   - Efficiency v. Flow for pumps
   - Volume v. Depth for tanks
   - Head Loss v. Flow for General Purpose Valves
2. The points of a curve must be entered in order of increasing X-values (lower to higher).
3. If the input file will be used with the Windows version of EPANET, then adding a comment which contains the curve type and description, separated by a colon, directly above the first entry for a curve will ensure that these items appear correctly in EPANET's Curve Editor. Curve types include **PUMP, EFFICIENCY, VOLUME**, and **HEADLOSS**. See the examples below.

Example:

```
[CURVES]
;ID   Flow    Head
;PUMP: Curve for Pump 1
C1    0       200
C1    1000    100
C1    3000    0
;ID   Flow    Effic.
;EFFICIENCY:
E1    200     50
E1    1000    85
E1    2000    75
E1    3000    65
```

7.1.3 [QUALITY]

Purpose:

Defines initial water quality at nodes.

Format:

One line per node containing:

- Node ID label
- Initial quality

Remarks:

1. Quality is assumed to be zero for nodes not listed.
2. Quality represents concentration for chemicals, hours for water age, or percent for source tracing.
3. The **[QUALITY]** section is optional.

7.1.4 [OPTIONS]

Purpose:

Defines various simulation options.

Formats:

UNITS	**CFS** / **GPM** / **MGD** / **IMGD** / **AFD** /
	LPS / **LPM** / **MLD** / **CMH** / **CMD**
HEADLOSS	**H-W** / **D-W** / **C-M**
HYDRAULICS	**USE** / **SAVE** *filename*
VISCOSITY	*value*
SPECIFIC GRAVITY	*value*
TRIALS	*value*
ACCURACY	*value*
FLOWCHANGE	*value*
HEADERROR	*value*
CHECKFREQ	*value*
MAXCHECK	*value*
DAMPLIMIT	*value*
UNBALANCED	**STOP** / **CONTINUE** / **CONTINUE** *n*
DEMAND MODEL	**DDA** / **PDA**
MINIMUM PRESSURE	*value*
REQUIRED PRESSURE	*value*
PRESSURE EXPONENT	*value*
PATTERN	*id*
DEMAND MULTIPLIER	*value*
EMITTER EXPONENT	*value*
QUALITY	**NONE** / **CHEMICAL** / **AGE** / **TRACE** *nodeID*
DIFFUSIVITY	*value*
TOLERANCE	*value*
MAP	*filename*

Definitions:

UNITS sets the units in which flow rates are expressed where:

- **CFS** = cubic feet per second
- **GPM** = gallons per minute
- **MGD** = million gallons per day
- **IMGD** = Imperial MGD
- **AFD** = acre-feet per day
- **LPS** = liters per second
- **LPM** = liters per minute
- **MLD** = million liters per day
- **CMH** = cubic meters per hour
- **CMD** = cubic meters per day

For **CFS, GPM, MGD, IMGD**, and **AFD** other input quantities are expressed in US Customary Units. If flow units are in liters or cubic meters then Metric Units must be used for all other input quantities as well. (See the Measurement Units topic). The default flow units are **GPM**.

HEADLOSS selects a formula to use for computing head loss for flow through a pipe. The choices are the Hazen-Williams (**H-W**), Darcy-Weisbach (**D-W**), or Chezy-Manning (**C-M**) formulas. The default is **H-W**.

The **HYDRAULICS** option allows you to either **SAVE** the current hydraulics solution to a file or **USE** a previously saved hydraulics solution. This is useful when studying factors that only affect water quality behavior.

VISCOSITY is the kinematic viscosity of the fluid being modeled relative to that of water at 20 deg. C (1.0 centistoke). The default value is 1.0.

SPECIFIC GRAVITY is the ratio of the density of the fluid being modeled to that of water at 4 deg. C (unitless). The default value is 1.0.

TRIALS are the maximum number of trials used to solve network hydraulics at each hydraulic time step of a simulation. The default is 40.

ACCURACY prescribes the convergence criterion that determines when a hydraulic solution has been reached. The trials end when the sum of all flow changes from the previous solution divided by the total flow in all links is less than this number. The default is 0.001.

FLOWCHANGE is a similar convergence criterion requiring that the largest absolute flow change between the current and previous solutions be less than the specified value (in flow units). The default is 0 which means that this criterion is not used.

HEADERROR is yet another convergence criterion requiring that the head loss computed by the head loss formula compared to the difference in nodal heads across each link be less than the specified value (in ft or m). The default is 0 which means that this criterion is not used.

CHECKFREQ sets the number of solution trials that pass during hydraulic balancing before the status of pumps, check valves, flow control valves and pipes connected to tanks are once again updated. The default value is 2, meaning that status checks are made every other trial.

MAXCHECK is the number of solution trials after which periodic status checks are discontinued. Instead, a status check is made only after convergence is achieved. The default value is 10, meaning that after 10 trials, instead of checking status every **CHECKFREQ** trials, status is checked only at convergence.

DAMPLIMIT is the accuracy value at which solution damping and status checks on PRVs and PSVs should begin. Damping limits all flow changes to 60% of what they would otherwise be as future trials unfold. The default is 0 which indicates that no damping should be used and that status checks on control valves are made at every iteration.

UNBALANCED determines what happens if a hydraulic solution cannot be reached within the prescribed number of **TRIALS** at some hydraulic time step into the simulation. **STOP** will halt the entire analysis at that point. **CONTINUE** will continue the analysis with a warning message issued. **CONTINUE n** will continue the search for a solution for another **n** trials with the status of all links held fixed at their current settings. The simulation will be continued at this point with a message issued about whether convergence was achieved or not. The default choice is **STOP**.

DEMAND MODEL specifies whether a demand driven analysis (**DDA**) or a pressure driven analysis (**PDA**) should be made. Under **DDA** full nodal demands are always met even if negative pressures result. **PDA** assumes that demand varies between 0 and its full value as a power function of nodal pressure. The default demand model is **DDA**.

MINIMUM PRESSURE is the pressure below which no demand can be delivered under a pressure driven analysis. It has no effect on a demand driven analysis. Its default value is 0.

REQUIRED PRESSURE is the pressure required to supply a node's full demand under a pressure driven analysis. It has no effect on a demand driven analysis. It must be at least 0.1 psi or m higher than the MINIMUM PRESSURE, which is also its default value.

PRESSURE EXPONENT is the power to which pressure is raised when computing the demand delivered to a node under a pressure driven analysis. It has no effect on a demand driven analysis. Its default value is 0.5.

PATTERN provides the ID label of a default demand pattern to be applied to all junctions where no demand pattern was specified. If no such pattern exists in the [PATTERNS] section then by default the pattern consists of a single multiplier equal to 1.0. If this option is not used, then the global default demand pattern has a label of "1".

The **DEMAND MULTIPLIER** is used to adjust the values of baseline demands for all junctions and all demand categories. For example, a value of 2 doubles all baseline demands, while a value of 0.5 would halve them. The default value is 1.0.

EMITTER EXPONENT specifies the power to which the pressure at a junction is raised when computing the flow issuing from an emitter. The default is 0.5.

QUALITY selects the type of water quality analysis to perform. The choices are **NONE, CHEMICAL, AGE**, and **TRACE**. In place of **CHEMICAL** the actual name of the chemical can be used followed by its concentration units (e.g., **CHLORINE mg/L**). If **TRACE** is selected it must be followed by the ID label of the node being traced. The default selection is **NONE** (no water quality analysis).

DIFFUSIVITY is the molecular diffusivity of the chemical being analyzed relative to that of chlorine in water. The default value is 1.0. Diffusivity is only used when mass transfer limitations are considered in pipe wall reactions. A value of 0 will cause EPANET to ignore mass transfer limitations.

TOLERANCE is the difference in water quality level below which one can say that one parcel of water is essentially the same as another. The default is 0.01 for all types of quality analyses (chemical, age (measured in hours), or source tracing (measured in percent)).

MAP is used to supply the name of a file containing coordinates of the network's nodes so that a map of the network can be drawn. It is not used for any hydraulic or water quality computations.

Remarks:

1. All options assume their default values if not explicitly specified in this section.
2. Items offset by slashes (/) indicate allowable choices.

Example:

```
[OPTIONS]
UNITS  CFS
HEADLOSS  D-W
DEMAND MODEL  PDA
REQUIRED PRESSURE  40
QUALITY  TRACE  Tank23
UNBALANCED  CONTINUE  10
```

7.1.5 [BACKDROP]

Purpose:

Identifies a backdrop image and dimensions for visualizing the network's layout.

Formats:

DIMENSIONS	*LLx LLy URx URy*
UNITS	**FEET/METERS/DEGREES/NONE**
FILE	*filename*
OFFSET	*X Y*

Definitions:

DIMENSIONS provides the X and Y coordinates of the lower-left and upper-right corners of the network's bounding rectangle. Defaults are the extents of the nodal coordinates supplied in the [COORDINATES] section.

UNITS specifies the units that the network's dimensions are given in. Default is **NONE**.

FILE supplies the name of the file that contains a backdrop image for the network.

OFFSET lists the X and Y distance that the upper-left corner of the backdrop image is offset from the upper-left corner of the network's bounding rectangle. Default is zero offset.

Remarks:

1. The [BACKDROP] section is optional and only provides support for an external GUI program that uses the EPANET engine.
2. Only Windows Enhanced Metafiles and bitmap files can be used as backdrops.

7.1.6 [JUNCTIONS]

Purpose:

Defines junction nodes contained in the network.

Format:

One line for each junction containing:

- ID label
- Elevation, ft (m)
- Base demand flow (flow units) (optional)
- Demand pattern ID (optional)

Remarks:

1. A **[JUNCTIONS]** section with at least one junction is required.
2. If no demand pattern is supplied then the junction demand follows the Default Demand Pattern provided in the [OPTIONS] section, or Pattern 1 if no Default Pattern is specified. If the Default Pattern (or Pattern 1) does not exist, then the demand remains constant.
3. Demands can also be entered in the [DEMANDS] section and include multiple demand categories per junction.

Example:

```
[JUNCTIONS]
;ID    Elev.   Demand   Pattern
;------------------------------
J1     100     50       Pat1
J2     120     10              ;Uses default demand pattern
J3     115                     ;No demand at this junction
```

7.1.7 [PATTERNS]

Purpose:

Defines time patterns.

Format:

One or more lines for each pattern containing:

- Pattern ID label
- One or more multipliers

Remarks:

1. Multipliers define how some base quantity (e.g., demand) is adjusted for each time period.
2. All patterns share the same time period interval as defined in the [TIMES] section.
3. Each pattern can have a different number of time periods.
4. When the simulation time exceeds the pattern length the pattern wraps around to its first period.
5. Use as many lines as it takes to include all multipliers for each pattern.

Example:

```
[PATTERNS]
;Pattern P1
P1    1.1   1.4   0.9   0.7
P1    0.6   0.5   0.8   1.0
;Pattern P2
P2    1     1     1     1
P2    0     0     1
```

7.1.8 [REACTIONS]

Purpose:

Defines parameters related to chemical reactions occurring in the network.

Formats:

ORDER BULK / WALL / TANK *value*

GLOBAL BULK / WALL *value*

BULK / WALL *pipeID value*

TANK *tankID value*

LIMITING POTENTIAL *value*

ROUGHNESS CORRELATION *value*

Definitions:

ORDER is used to set the order of reactions occurring in the bulk fluid, at the pipe wall, or in tanks, respectively. Values for wall reactions must be either 0 or 1. If not supplied the default reaction order is 1.0.

GLOBAL is used to set a global value for all bulk reaction coefficients (pipes and tanks) or for all pipe wall coefficients. The default value is zero.

BULK, WALL, and **TANK** are used to override the global reaction coefficients for specific pipes and tanks.

LIMITING POTENTIAL specifies that reaction rates are proportional to the difference between the current concentration and some limiting potential value.

ROUGHNESS CORRELATION will make all default pipe wall reaction coefficients be related to pipe roughness in the following manner:

Head Loss Equation	Roughness Correlation
Hazen-Williams	**F / C**
Darcy-Weisbach	**F / log(e/D)**
Chezy-Manning	**F** $*$ **n**

where **F** = roughness correlation, **C** = Hazen-Williams C-factor, **e** = Darcy-Weisbach roughness, **D** = pipe diameter, and **n** = Chezy-Manning roughness coefficient. The default value computed this way can be overridden for any pipe by using the **WALL** format to supply a specific value for the pipe.

Remarks:

1. Remember to use positive numbers for growth reaction coefficients and negative numbers for decay coefficients.
2. The time units for all reaction coefficients are 1/days.
3. All entries in this section are optional. Items offset by slashes (/) indicate allowable choices.

Example:

```
[REACTIONS]
ORDER WALL    0    ;Wall reactions are zero-order
GLOBAL BULK  -0.5  ;Global bulk decay coeff.
GLOBAL WALL  -1.0  ;Global wall decay coeff.
WALL   P220  -0.5  ;Pipe-specific wall coeffs.
WALL   P244  -0.7
```

7.1.9 [TIMES]

Purpose: Defines various time step parameters used in the simulation.

Formats:

DURATION	*value* (units)
HYDRAULIC TIMESTEP	*value* (units)
QUALITY TIMESTEP	*value* (units)
RULE TIMESTEP	*value* (units)
PATTERN TIMESTEP	*value* (units)
PATTERN START	*value* (units)
REPORT TIMESTEP	*value* (units)
REPORT START	*value* (units)
START CLOCKTIME	*value* (**AM** / **PM**)
STATISTIC	**NONE** / **AVERAGED** / **MINIMUM** / **MAXIMUM** / **RANGE**

Definitions:

DURATION is the duration of the simulation. Use 0 to run a single period snapshot analysis. The default is 0.

HYDRAULIC TIMESTEP determines how often a new hydraulic state of the network is computed. If greater than either the **PATTERN** or **REPORT** time step it will be automatically reduced. The default is 1 hour.

QUALITY TIMESTEP is the time step used to track changes in water quality throughout the network. The default is 1/10 of the hydraulic time step.

RULE TIMESTEP is the time step used to check for changes in system status due to activation of rule-based controls between hydraulic time steps. The default is 1/10 of the hydraulic time step.

PATTERN TIMESTEP is the interval between time periods in all time patterns. The default is 1 hour.

PATTERN START is the time offset at which all patterns will start. For example, a value of 6 hours would start the simulation with each pattern in the time period that corresponds to hour 6. The default is 0.

REPORT TIMESTEP sets the time interval between which output results are reported. The default is 1 hour.

REPORT START is the length of time into the simulation at which output results begin to be reported. The default is 0.

START CLOCKTIME is the time of day (e.g., 3:00 PM) at which the simulation begins. The default is 12:00 AM midnight.

STATISTIC determines what kind of statistical post-processing should be done on the time series of simulation results generated. **AVERAGED** reports a set of time-averaged results, **MINIMUM** reports only the minimum values, **MAXIMUM** the maximum values, and **RANGE** reports the difference between the minimum and maximum values. **NONE** reports the full time series for all quantities for all nodes and links and is the default.

Remarks:

1. Units can be **SECONDS (SEC), MINUTES (MIN), HOURS**, or **DAYS**. The default is **HOURS**.
2. If no units are supplied, then time values can be expressed in either decimal hours or in hours:minutes notation.
3. All entries in the [TIMES] section are optional. Items offset by slashes (/) indicate allowable choices.

Example:

```
[TIMES]
DURATION   240 HOURS
QUALITY TIMESTEP   3 MIN
QUALITY TIMESTEP   0:03
REPORT START   120
START CLOCKTIME   6:00 AM
```

7.1.10 [COORDINATES]

Purpose:

Assigns map coordinates to network's nodes.

Format:

One line for each node containing:

- Node ID label
- X-coordinate
- Y-coordinate

Remarks:

1. Include one line for each node that has coordinates.
2. The coordinates represent the distance from the node to an arbitrary origin at the lower left of the network. Any convenient units of measure for this distance can be used.
3. The locations of the nodes need not be to actual scale.
4. A [COORDINATES] section is optional and only provides support for an external GUI program that uses the EPANET engine.

7.1.11 [RESERVOIRS]

Purpose:

Defines all reservoir nodes contained in the network.

Format:

One line for each reservoir containing:

- ID label
- Head, ft (m)
- Head pattern ID (optional)

Remarks:

1. Head is the hydraulic head (elevation + pressure head) of water in the reservoir.
2. A head pattern can be used to make the reservoir head vary with time.
3. At least one reservoir or tank must be contained in the network.

Example:

```
[RESERVOIRS]
;ID    Head      Pattern
;-----------------------
R1     512              ;Head stays constant
R2     120       Pat1    ;Head varies with time
```

7.1.12 [ENERGY]

Purpose:

Defines parameters used to compute pumping energy and cost.

Formats:

GLOBAL PRICE / **PATTERN** / **EFFIC** *value*

PUMP *pumpID* **PRICE** / **PATTERN** / **EFFIC** *value*

DEMAND CHARGE *value*

Remarks:

1. First format is used to set global default values of energy price, price pattern, and pumping efficiency for all pumps.
2. Second format is used to override global defaults for specific pumps.
3. Parameters are defined as follows:
   - **PRICE** = average cost per kW-hour,
   - **PATTERN** = ID label of time pattern describing how energy price varies with time,
   - **EFFIC** = either a single percent efficiency for global setting or the ID label of an efficiency curve for a specific pump,
   - **DEMAND CHARGE** = added cost per maximum kW usage during the simulation period.
4. The default global pump efficiency is 75% and the default global energy price is 0.
5. All entries in this section are optional. Items offset by slashes (/) indicate allowable choices.

Example:

```
[ENERGY]
GLOBAL PRICE      0.05   ;Sets global energy price
GLOBAL PATTERN    PAT1   ;and time-of-day pattern
PUMP   23  PRICE  0.10   ;Overrides price for Pump 23
PUMP   23  EFFIC  E23    ;Assigns effic. curve to Pump 23
```

7.1.13 [SOURCES]

Purpose:

Defines locations of water quality sources.

Format:

One line for each water quality source containing:

- Node ID label
- Source type (**CONCEN, MASS, FLOWPACED, or SETPOINT**)
- Baseline source strength
- Time pattern ID (optional)

Remarks:

1. For **MASS** type sources, strength is measured in mass flow per minute. All other types measure source strength in concentration units.
2. Source strength can be made to vary over time by specifying a time pattern.
3. A **CONCEN** source:
   - represents the concentration of any external source inflow to the node
   - applies only when the node has a net negative demand (water enters the network at the node)
   - if the node is a junction, reported concentration is the result of mixing the source flow and inflow from the rest of the network
   - if the node is a reservoir, the reported concentration is the source concentration
   - if the node is a tank, the reported concentration is the internal concentration of the tank
   - is best used for nodes that represent source water supplies or treatment works (e.g., reservoirs or nodes assigned a negative demand)
   - do not use at storage tanks with simultaneous inflow/outflow.
4. A **MASS, FLOWPACED, or SETPOINT** source:
   - represents a booster source, where the substance is injected directly into the network regardless of what the demand at the node is
   - affects water leaving the node to the rest of the network in the following way:
     - a **MASS** booster adds a fixed mass flow to that resulting from inflow to the node
     - a **FLOWPACED** booster adds a fixed concentration to the resultant inflow concentration at the node
     - a **SETPOINT** booster fixes the concentration of any flow leaving the node (as long as the concentration resulting from the inflows is below the setpoint)
   - the reported concentration at a junction or reservoir booster source is the concentration that results after the boosting is applied; the reported concentration for a tank with a booster source is the internal concentration of the tank
   - is best used to model direct injection of a tracer or disinfectant into the network or to model a contaminant intrusion.
5. A [SOURCES] section is not needed for simulating water age or source tracing.

Example:

```
[SOURCES]
;Node  Type    Strength  Pattern
;--------------------------------
N1     CONCEN  1.2       Pat1    ;Concentration varies with time
N44    MASS    12                ;Constant mass injection
```

7.1.14 [REPORT]

Purpose:

Describes the contents of the output report produced from a simulation.

Formats:

PAGESIZE	*value*
FILE	*filename*
STATUS	**YES** / **NO** / **FULL**
SUMMARY	**YES** / **NO**
MESSAGES	**YES** / **NO**
ENERGY	**YES** / **NO**
NODES	**NONE** / **ALL**/ node1 node2 ...
LINKS	**NONE** / **ALL**/ node1 node2 ...
variable	**YES** / **NO**
variable	**BELOW** / **ABOVE** / **PRECISION** *value*

Definitions:

PAGESIZE sets the number of lines written per page of the output report. The default is 0, meaning that no line limit per page is in effect.

FILE supplies the name of a file to which the output report will be written (ignored by the Windows version of EPANET). The default is to write the report to the project's Report File file.

STATUS determines whether a hydraulic status report should be generated. If **YES** is selected the report will identify all network components that change status during each time step of the simulation. If **FULL** is selected, then the status report will also include information from each trial of each hydraulic analysis. This level of detail is only useful for de-bugging networks that become hydraulically unbalanced. The default is **NO**.

SUMMARY determines whether a summary table of number of network components and key analysis options is generated. The default is **YES**.

ENERGY determines if a table reporting average energy usage and cost for each pump is provided. The default is **NO**.

NODES identifies which nodes will be reported on. You can either list individual node ID labels or use the keywords **NONE** or **ALL**. Additional **NODES** lines can be used to continue the list. The default is **NONE**.

LINKS identifies which links will be reported on. You can either list individual link ID labels or use the keywords **NONE** or **ALL** Additional **LINKS** lines can be used to continue the list. The default is **NONE**.

The *variable* reporting option is used to identify which quantities are reported on, how many decimal places are displayed, and what kind of filtering should be used to limit output reporting. Node variables that can be reported on include:

- **Elevation**
- **Demand**
- **Head**
- **Pressure**
- **Quality**

Link variables include:

- **Length**
- **Diameter**
- **Flow**
- **Velocity**
- **Headloss**
- **State** (same as status: open, active, closed)
- **Setting** (roughness for pipes, speed for pumps, pressure/flow setting for valves)
- **Reaction** (reaction rate)
- **F-Factor** (friction factor).

The default quantities reported are **Demand, Head, Pressure,** and **Quality** for nodes and **Flow, Velocity,** and **Headloss** for links. The default precision is two decimal places.

Remarks:

1. All options assume their default values if not explicitly specified in this section.
2. Items offset by slashes (/) indicate allowable choices.
3. The default is to not report on any nodes or links, so a **NODES** or **LINKS** option must be supplied if you wish to report results for these items.

Example:

The following example reports on nodes N1, N2, N3, and N17 and all links with velocity above 3.0. The standard node variables (Demand, Head, Pressure, and Quality) are reported on while only Flow, Velocity, and F-Factor (friction factor) are displayed for links.

```
[REPORT]
NODES N1 N2 N3 N17
LINKS ALL
FLOW YES
VELOCITY PRECISION 4
F-FACTOR PRECISION 4
VELOCITY ABOVE 3.0
```

7.1.15 [VERTICES]

Purpose:

Assigns interior vertex points that describe the shape of network links.

Format:

One line for each vertex point in each link containing such points that includes:

- Link ID label
- X-coordinate
- Y-coordinate

Remarks:

1. Vertex points allow links to be drawn as polylines instead of simple straight-lines between their end nodes.
2. The coordinates refer to the same coordinate system used for node and label coordinates.
3. A [VERTICES] section is optional and only provides support for an external GUI program that uses the EP↔ ANET engine.

7.1.16 [TANKS]

Purpose:

Defines all tank nodes contained in the network.

Format:

One line for each junction containing:

- ID label
- Bottom elevation, ft (m)
- Initial water level, ft (m)
- Minimum water level, ft (m)
- Maximum water level, ft (m)
- Nominal diameter, ft (m)
- Minimum volume, cubic ft (cubic meters)
- Volume curve ID (optional)
- Overflow indicator (**YES** / **NO**) (optional)

Remarks:

1. Water surface elevation equals bottom elevation plus water level.
2. Non-cylindrical tanks can be modeled by specifying a curve of volume versus water depth in the [CURVES] section.
3. If a volume curve is supplied the diameter value can be any non-zero number
4. Minimum volume (tank volume at minimum water level) can be zero for a cylindrical tank or if a volume curve is supplied.
5. If the overflow indicator is **YES** then the tank is allowed to overflow once it reaches it maximum water level. The default is no overflow.
6. If the tank does not use a volume curve then an asterisk (∗) can be used as a placeholder for it if an overflow indicator is specified.
7. A network must contain at least one tank or reservoir.

Example:

```
[TANKS]
;ID   Elev.  InitLvl  MinLvl  MaxLvl  Diam  MinVol  VolCurve  Overflow
;-------------------------------------------------------------------
;Cylindrical tank that can overflow
T1    100     15       5       25     120   0       *          YES
;Non-cylindrical tank with arbitrary diameter
T2   100     15       5       25     1     0        VC1
```

7.1.17 [STATUS]

Purpose:

Defines initial status of selected links at the start of a simulation.

Format:

One line per link being controlled containing:

- Link ID label
- Status or setting

Remarks:

1. Links not listed in this section have a default status of **OPEN** (for pipes and pumps) or **ACTIVE** (for valves).
2. The Status value assigned in this section can be **OPEN** or **CLOSED**. For control valves (e.g., PRVs, FCVs, etc.) this means that the valve is either fully opened or closed, not active at its control setting.
3. The Setting value can be a speed setting for pumps or valve setting for valves.
4. The initial status of pipes can also be set in the [PIPES] section.
5. Check valves cannot have their status be preset.
6. Use [CONTROLS] or [RULES] to change status or setting at some future point in the simulation.
7. If a **CLOSED** or **OPEN** control valve is to become **ACTIVE** again, then its pressure or flow setting must be specified in the control or rule that reactivates it.

Example:

```
[STATUS]
; Link   Status/Setting
;----------------------
  L22    CLOSED        ;Link L22 is closed
  P14    1.5           ;Speed for pump P14
  PRV1   OPEN          ;PRV1 forced open
                       ;(overrides normal operation)
```

7.1.18 [MIXING]

Purpose:

Identifies the model that governs mixing within storage tanks.

Format:

One line per tank containing:

- Tank ID label
- Mixing model (**MIXED, 2COMP, FIFO, or LIFO**)
- Compartment volume (fraction)

Remarks:

1. Mixing models include:
   - Single compartment, complete mix model (**MIXED**)
   - Two-compartment complete mix model (**2COMP**)
   - Plug flow, first in, first out model (**FIFO**)
   - Stacked plug flow, last in, first out model (**LIFO**)
2. The compartment volume parameter only applies to the two-compartment model and represents the fraction of the total tank volume devoted to the inlet/outlet compartment.
3. The **[MIXING]** section is optional. Tanks not described in this section are assumed to be completely mixed.

Example:

```
[MIXING]
;Tank      Model
;-----------------------
T12        LIFO
T23        2COMP     0.2
```

7.1.19 [LABELS]

Purpose:

Assigns coordinates to labels added to a network's visualization.

Format:

One line for each label containing:

- X-coordinate
- Y-coordinate
- Text of label in double quotes
- ID label of an anchor node (optional)

Remarks:

1. Include one line for each label.
2. The coordinates refer to the upper left corner of the label and are with respect to an arbitrary origin at the lower left of the network.
3. The optional anchor node anchors the label to the node when the network layout is re-scaled during zoom-in operations.
4. The [LABELS] section is optional and only provides support for an external GUI program that uses the EP↩ANET engine.

7.1.20 [PIPES]

Purpose:

Defines all pipe links contained in the network.

Format:

One line for each pipe containing:

- ID label
- ID of start node
- ID of end node
- Length, ft (m)
- Diameter, inches (mm)
- Roughness coefficient
- Minor loss coefficient
- Status (**OPEN, CLOSED, or CV**)

Remarks:

1. Roughness coefficient is unitless for Hazen-Williams and Chezy-Manning head loss formulas and has units of millifeet (mm) for the Darcy-Weisbach formula. Choice of head loss formula is supplied in the [OPTIONS] section.
2. Setting status to **CV** means that the pipe contains a check valve restricting flow to one direction.
3. If minor loss coefficient is 0 and pipe is **OPEN** then these two items can be dropped from the input line.

Example:

```
[PIPES]
;ID   Node1  Node2   Length   Diam.  Roughness  Mloss   Status
;-------------------------------------------------------------
P1    J1     J2      1200     12      120       0.2     OPEN
P2    J3     J2       600      6      110       0       CV
P3    J1     J10     1000     12      120
```

7.1.21 [CONTROLS]

Purpose:

Defines simple controls that modify links based on a single condition.

Format:

One line for each control which can be of the form:

LINK *linkID* *status* **IF NODE** *nodeID* **ABOVE / BELOW** *value*

LINK *linkID* *status* **AT TIME** *time*

LINK *linkID* *status* **AT CLOCKTIME** *clocktime* **AM / PM**

where:

linkID	=	a link ID label
status	=	**OPEN** / **CLOSED**, a pump speed setting, or a control valve setting
nodeID	=	a node ID label
value	=	a pressure for a junction or a water level for a tank
time	=	a time since the start of the simulation in hours
clocktime	=	a 24-hour clock time (hrs:min)

Remarks:

1. Simple controls are used to change link status or settings based on tank water level, junction pressure, time into the simulation or time of day.

2. See the notes for the [STATUS] section for conventions used in specifying link status and setting, particularly for control valves.

Examples:

```
[CONTROLS]
;Close Link 12 if the level in Tank 23 exceeds 20 ft.
LINK 12 CLOSED IF NODE 23 ABOVE 20

;Open Link 12 if the pressure at Node 130 is under 30 psi
LINK 12 OPEN IF NODE 130 BELOW 30

;Pump PUMP02's speed is set to 1.5 at 16 hours into the simulation
LINK PUMP02 1.5 AT TIME 16

;Link 12 is closed at 10 am and opened at 8 pm throughout the simulation
LINK 12 CLOSED AT CLOCKTIME 10 AM
LINK 12 OPEN AT CLOCKTIME 8 PM
```

7.1.22 [PUMPS]

Purpose:

Defines all pump links contained in the network.

Format:

One line for each pump containing:

- ID label
- ID of start node
- ID of end node
- Keyword and Value (can be repeated)

Remarks:

1. Keywords consists of:
    - **POWER** - power for constant energy pump, hp (kw)
    - **HEAD** - ID of curve that describes head versus flow for the pump
    - **SPEED** - relative speed setting (normal speed is 1.0, 0 means pump is off)
    - **PATTERN** - ID of time pattern that describes how speed setting varies with time
2. Either **POWER** or **HEAD** must be supplied for each pump. The other keywords are optional.

Example:

```
[PUMPS]
;ID    Node1    Node2    Properties
;---------------------------------------------
Pump1   N12      N32      HEAD Curve1
Pump2   N121     N55      HEAD Curve1  SPEED 1.2
Pump3   N22      N23      POWER 100
```

7.1.23 [RULES]

Purpose:

Defines rule-based controls which modify links based on a combination of conditions.

Format:

Each rule is a series of statements of the form:

RULE *ruleID*
IF *condition_1*
AND *condition_2*
OR *condition_3*
AND *condition_4*
etc.
THEN *action_1*
AND *action_2*
etc.
ELSE *action_3*
AND *action_4*
etc.
PRIORITY *value*

where:

ruleID	=	an ID label assigned to the rule
conditon_n	=	a condition clause
action_n	=	an action clause
PRIORITY	=	a priority value (e.g., a number from 1 to 5)

Remarks:

1. Only the **RULE, IF** and **THEN** portions of a rule are required; the other portions are optional.
2. When mixing **AND** and **OR** clauses, the **OR** operator has higher precedence than **AND**, i.e.,
   `IF A or B and C`

is equivalent to

`IF (A or B) and C`
If the interpretation was meant to be
`IF A or (B and C)`
then this can be expressed using two rules as in
`IF A THEN ...`
`IF B and C THEN ...`

3. The **PRIORITY** value is used to determine which rule applies when two or more rules require that conflicting actions be taken on a link. A rule without a priority value always has a lower priority than one with a value. For two rules with the same priority value, the rule that appears first is given the higher priority.

Example:

```
[RULES]
RULE 1
IF   TANK   1 LEVEL ABOVE 19.1
THEN PUMP 335 STATUS IS CLOSED
AND  PIPE 330 STATUS IS OPEN
RULE 2
IF   SYSTEM CLOCKTIME >= 8 AM
AND  SYSTEM CLOCKTIME < 6 PM
AND  TANK 1 LEVEL BELOW 12
THEN PUMP 335 STATUS IS OPEN
RULE 3
IF   SYSTEM CLOCKTIME >= 6 PM
OR   SYSTEM CLOCKTIME < 8 AM
AND  TANK 1 LEVEL BELOW 14
THEN PUMP 335 STATUS IS OPEN
```

7.1.24 Condition Clauses

A condition clause in a Rule-Based Control takes the form of:

object id attribute relation value

where

object	=	a category of network object
id	=	the object's ID label
attribute	=	an attribute or property of the object
relation	=	a relational operator
value	=	an attribute value

Some example conditional clauses are:
```
JUNCTION 23 PRESSURE > 20
TANK T200 FILLTIME BELOW 3.5
LINK 44 STATUS IS OPEN
SYSTEM CLOCKTIME = 7:30 AM
SYSTEM DEMAND >= 1500
```

Objects can be any of the following keywords:
NODE JUNCTION TANK RESERVOIR
LINK PIPE PUMP VALVE
SYSTEM

When **SYSTEM** is used in a condition no ID is supplied.

The following attributes can be used with Node-type objects:
DEMAND
HEAD
PRESSURE

The following attributes can be used with Tanks:
LEVEL
FILLTIME (hours needed to fill a tank)
DRAINTIME (hours needed to empty a tank)

These attributes can be used with Link-Type objects:
FLOW
STATUS (OPEN, CLOSED, or ACTIVE)
SETTING (pump speed or valve setting)

The **SYSTEM** object can use the following attributes:

DEMAND	(total system demand)
TIME	(hours from the start of the simulation expressed
	either as a decimal number or in hours:minutes format)
CLOCKTIME	(24-hour clock time with **AM** or **PM** appended)

Relation operators consist of the following:

=	**IS**
<>	**NOT**
<	**BELOW**
>	**ABOVE**
<=	
>=	

7.1.25 Action Clauses

An action clause in a Rule-Based Control takes the form of:

object id **STATUS / SETTING IS** *value*

where

object	=	**LINK, PIPE, PUMP, or VALVE** keyword
id	=	the object's ID label
value	=	a status condition (**OPEN or CLOSED**), pump speed setting, or valve setting

Some example action clauses are:
```
LINK 23 STATUS IS CLOSED
PUMP P100 SETTING IS 1.5
VALVE 123 SETTING IS 90
```

See the notes for the [STATUS] section for conventions used in specifying link status and setting, particularly for control valves.

7.1.26 [VALVES]

Purpose:

Defines all control valve links contained in the network.

Format:

One line for each valve containing:

- ID label
- ID of start node
- ID of end node
- Diameter, inches (mm)
- Valve type
- Valve setting
- Minor loss coefficient

Remarks:

1. Valve types and settings include:

Valve Type	Setting
PRV (pressure reducing valve)	Pressure, psi (m)
PSV (pressure sustaining valve)	Pressure, psi (m)
PBV (pressure breaker valve)	Pressure, psi (m)
FCV (flow control valve)	Flow (flow units)
TCV (throttle control valve)	Loss Coefficient
GPV (general purpose valve)	ID of head loss curve

2. Shutoff valves and check valves are considered to be part of a pipe, not a separate control valve component (see [PIPES]).

7.1.27 [DEMANDS]

Purpose:

Supplement to [JUNCTIONS] section for defining multiple water demands at junction nodes.

Format:

One line for each category of demand at a junction containing:

- Junction ID label
- Base demand (flow units)
- Demand pattern ID (optional)

- Name of demand category preceded by a semicolon (optional)

Remarks:

1. Only use for junctions whose demands need to be changed or supplemented from entries in **[JUNCTIONS]** section.
2. Data in this section replaces any demand entered in the **[JUNCTIONS]** section for the same junction.
3. An unlimited number of demand categories can be entered per junction.
4. If no demand pattern is supplied then the junction demand follows the **Pattern** entry in the [OPTIONS] section, or Pattern 1 if no such pattern is supplied. If the default pattern (or Pattern 1) does not exist, then the demand remains constant.

Example:
```
[DEMANDS]
;ID     Demand   Pattern   Category
;---------------------------------
J1      100       101      ;Domestic
J1      25        102      ;School
J256    50        101      ;Domestic
```

7.1.28 [EMITTERS]

Purpose:

Defines junctions modeled as emitters (sprinklers or orifices).

Format:

One line for each emitter containing:

- Junction ID label
- Flow coefficient, flow units at 1 psi (1 meter) pressure drop

Remarks:

1. Emitters are used to model flow through sprinkler heads or pipe leaks.
2. Flow out of the emitter equals the product of the flow coefficient and the junction pressure raised to a power.
3. The power can be specified using the **EMITTER EXPONENT** option in the [OPTIONS] section. The default power is 0.5, which normally applies to sprinklers and nozzles.
4. Actual demand reported in the program's results includes both the normal demand at the junction plus flow through the emitter.
5. An **[EMITTERS]** section is optional.

7.2 Report File

The Report file is the second file name supplied to the EN_open function (or the first file name to EN_init). It is used to log any error messages that occur when an Input file is being processed and to record all status messages that are generated during a hydraulic simulation. In addition, if the EN_report function is called the resulting report can also be written to this file as can user-generated lines of text using the EN_writeline function. The format of the report is controlled by statements placed in the [REPORT] section of the Input file and by similar statements included in calls to the EN_setreport function. Only results at a specified uniform reporting time interval are written to this file.

To suppress the writing of all error and warning messages to the Report file either include the command **`MESSAGES NO`** in the [REPORT] section of the Input file or call the Toolkit function **`EN_setreport("MESSAGES NO")`**.

To route a formatted report to a different file than the Report file either include the command **`FILE filename`** in the [REPORT] section of the Input file or call the Toolkit function **`EN_setreport("FILE filename")`**, where `filename` is the name of the file to use.

Toolkit clients will not be able to access the contents of a Report file until a project is closed. If access is needed before then, the EN_copyreport function can be used to copy its current contents to another file. A EN_clearreport function is also available to clear the current contents of the Report file.

7.3 Output File

The Output file is an unformatted binary file used to store both hydraulic and water quality results at uniform reporting intervals. It is the third file name supplied to the EN_open function (or second name to EN_init). If an empty string ("") is used as its name then a scratch temporary file will be used. Otherwise the Output file will be saved after the EN_close function is called. Saving this file is useful if further post-processing of the output results are needed.

The function EN_saveH will transfer hydraulic results to the Output file if no water quality analysis will be made. Using EN_solveQ to run a water quality analysis automatically saves both hydraulic and water quality results to this file. If the EN_initQ - EN_runQ - EN_nextQ set of functions is used to perform a water quality analysis, then results will be saved only if the **saveflag** argument of EN_initQ is set to **`EN_SAVE`**. Again, the need to save results to the Output file is application-dependent. If a formatted output report is to be generated using EN_report, then results must first be saved to the Output file.

The data written to the file is either 4-byte integers, 4-byte floats, or fixed-size strings whose size is a multiple of 4 bytes. This allows the file to be divided conveniently into 4-byte records. The file consists of four sections of the following sizes in bytes:

Section	Size in Bytes
Prolog	884 + 36∗Nnodes + 52∗Nlinks + 8∗Ntanks
Energy Usage	28∗Npumps + 4
Dynamic Results	(16∗Nnodes + 32∗Nlinks)∗Nperiods
Epilog	28

where:

- Nnodes = number of nodes (junctions + reservoirs + tanks),
- Nlinks = number of links (pipes + pumps + valves),
- Ntanks = number of tanks and reservoirs,

- Npumps = number of pumps,
- Nperiods = number of reporting periods.

All of these counts are themselves written to the file's Prolog or Epilog sections.

7.3.1 Prolog Section

The Prolog section of an EPANET binary output file contains the following data:

Item	Type	# Bytes
Magic Number = 516114521	Integer	4
Version (= 200)	Integer	4
Number of Nodes	Integer	4
Number of Reservoirs & Tanks	Integer	4
Number of Links	Integer	4
Number of Pumps	Integer	4
Number of Valves	Integer	4
Water Quality Option - see EN_QualityType	Integer	4
Traced Node Index	Integer	4
Flow Units Option	Integer	4
Pressure Units Option: 0 = psi 1 = meters 2 = kPa	Integer	4
Report Statistic Type - see EN_StatisticType	Integer	4
Reporting Start Time (sec)	Integer	4
Reporting Time Step (sec)	Integer	4
Simulation Duration (sec)	Integer	4
Project Title (1st line)	Char	80
Project Title (2nd line)	Char	80
Project Title (3rd line)	Char	80
Name of Input File	Char	260
Name of Report File	Char	260
Name of Quality Chemical	Char	32
Chemical Concentration Units	Char	32
ID String of Each Node	Char	32∗Nnodes
ID String of Each Link	Char	32∗Nlinks
Index of Head Node of Each Link	Integer	4∗Nlinks
Index of Tail Node of Each Link	Integer	4∗Nlinks
Type Code of Each Link (see EN_LinkType)	Integer	4∗Nlinks
Node Index of Each Tank	Integer	4∗Ntanks
Surface Area of Each Tank	Float	4∗Ntanks
Elevation of Each Node	Float	4∗Nnodes
Length of Each Link	Float	4∗Nlinks
Diameter of Each Link	Float	4∗Nlinks

7.3.2 Energy Usage Section

The Energy Usage section of an EPANET binary output file contains the following data:

Item (Repeated for Each Pump)	Type	# Bytes
Pump Index in list of links	Integer	4
Pump Utilization (%)	Float	4
Average Efficiency (%)	Float	4
Average kW/MGal (or kW/m^3)	Float	4
Average kW	Float	4
Peak kW	Float	4
Average Cost per Day	Float	4
Peak Energy Usage (kWh)	Float	4

7.3.3 Dynamic Results Section

The Dynamic Results section of an EPANET binary output file contains the following set of data for each reporting period (the reporting time step is written to the Output File's Prolog Section and the number of such steps is written to the Epilog Section):

Item	Type	# Bytes
Demand at Each Node	Float	4∗Nnodes
Head (Grade) at Each Node	Float	4∗Nnodes
Pressure at Each Node	Float	4∗Nnodes
Water Quality at Each Node	Float	4∗Nnodes
Flow in Each Link (negative for reverse flow)	Float	4∗Nlinks
Velocity in Each Link	Float	4∗Nlinks
Head Loss per 1000 Units of Length for Each Link (total head for pumps and head loss for valves)	Float	4∗Nlinks
Average Water Quality in Each Link	Float	4∗Nlinks
Status Code for Each Link: 0 = closed (pump shutoff head exceeded) 1 = temporarily closed 2 = closed 3 = open 4 = active (partially open) 5 = open (pump max. flow exceeded) 6 = open (FCV can't supply flow) 7 = open (PRV/PSV can't supply pressure)	Float	4∗Nlinks
Setting for Each Link	Float	4∗Nlinks
Reaction Rate for Each Link (mass/L/day)	Float	4∗Nlinks
Friction Factor for Each Link	Float	4∗Nlinks

7.3.4 Epilog Section

The Epilog section of an EPANET binary output file contains the following data:

Item	Type	# Bytes
Average bulk reaction rate (mass/hr)	Float	4
Average wall reaction rate (mass/hr)	Float	4
Average tank reaction rate (mass/hr)	Float	4
Average source inflow rate (mass/hr)	Float	4
Number of Reporting Periods	Integer	4
Warning Flag: 0 = no warnings 1 = warnings were generated	Integer	4
Magic Number = 516114521	Integer	4

7.4 Hydraulics File

The Hydraulics file is an unformatted binary file used to store the results of a hydraulic analysis. Results for all time periods are stored, including those at intermediate times when special hydraulic events occur (e.g., pumps and tanks opening or closing because control conditions have been satisfied).

Normally it is a temporary file that is deleted after the EN_deleteproject function is called. However, it will be saved if the EN_savehydfile function is called before that.

Likewise, a previously saved Hydraulics file can be used if the command **`HYDRAULICS USE`** filename appears in the [OPTIONS] section of the input file, or if the EN_usehydfile function is called.

When the Toolkit function EN_solveH is used to make a hydraulic analysis, results are automatically saved to the Hydraulics file. When the EN_initH - EN_runH - EN_nextH set of functions is used, the **initFlag** argument to E↵N_initH determines whether results are saved or not. The need to save hydraulic results is application-dependent. They must always be saved to the Hydraulics file if a water quality analysis will follow.

7.5 Header Files

The Toolkit provides several header files that are needed to develop C/C++ applications:

- **`epanet2.h`** contains declarations of the single-threaded version of the Toolkit (the ENxxx named functions).
- **`epanet2_2.h`** contains declarations of the multi-threaded version of the Toolkit (the EN_xxx named functions).
- **`epanet2_enums.h`** contains definitions of the symbolic constants used by the Toolkit.
- **`epanet2.lib`** must be linked in to any Toolkit application compiled for Windows using MS Visual C++. Developers need to issue an include directive for either `epanet2.h` or `epanet2_2.h` in their C/C++ code depending on whether they are building a single-threaded or multi-threaded application. There is no need to explicitly include `epanet2_enums.h` as it is automatically included by both of the other header files.

Several additional function declaration files that provide bindings for other programming languages are included with the Toolkit package:

- **`epanet2.bas`** for Visual Basic for Applications and Visual Basic 6
- **`epanet2.vb`** for Visual Basic .NET
- **`epanet2.pas`** for Delphi Pascal, Free Pascal or Lazarus.

These bindings only support the single-threaded version of the Toolkit.

Chapter 8

Measurement Units

The toolkit can use data expressed in either US Customary of SI Metric units. A project's unit system depends on the unit system used for its choice of flow units. If the EN_open function is used to supply data to a project from an Input File then its flow units are set in the [OPTIONS] section of the file. If the EN_init function is used to initialize a project then the choice of flow units is the fourth argument to the function. The following table lists the units used to express the various parameters in an EPANET model.

Parameter	US Customary	SI Metric
Concentration	mg/L or ug/L	mg/L or ug/L
Demand	(see Flow units)	(see Flow units)
Diameter (Pipes)	inches	millimeters
Diameter (Tanks)	feet	meters
Efficiency	percent	percent
Elevation	feet	meters
Emitter Coeff.	flow units @ 1 psi drop	flow units @ 1 meter drop
Energy	kwatt - hours	kwatt - hours
Flow	CFS (cubic feet / sec)	LPS (liters / sec)
	GPM (gallons / min)	LPM (liters / min)
	MGD (million gal / day)	MLD (megaliters / day)
	IMGD (Imperial MGD)	CMH (cubic meters / hr)
	AFD (acre-feet / day)	CMD (cubic meters / day)
Friction Factor	unitless	unitless
Head	feet	meters
Length	feet	meters
Minor Loss Coeff.	unitless	unitless
Power	horsepower	kwatts
Pressure	psi	meters
Reaction Coeff. (Bulk)	1/day (1st-order)	1/day (1st-order)
Reaction Coeff. (Wall)	mass/sq-ft/day (0-order)	mass/sq-m/day (0-order)
	ft/day (1st-order)	meters/day (1st-order)
Roughness Coeff.	millifeet (Darcy-Weisbach) unitless otherwise	mm (Darcy-Weisbach) unitless otherwise
Source Mass Injection	mass/minute	mass/minute
Velocity	ft/sec	meters/sec
Volume	cubic feet	cubic meters
Water Age	hours	hours

Chapter 9

Usage

The following topics briefly describe how to accomplish some basic tasks using the OWA-EPANET Toolkit in C/C++ code. See the Examples topic for code listings of complete applications of the Toolkit.

9.1 Creating a Project

Before any use is made of the Toolkit, a project and its handle must be created. After all processing is completed the project should be deleted. See the code snippet below:

```
EN_Project ph;  // a project handle
EN_createproject(&ph);
// Call functions that perform desired analysis
EN_deleteproject(ph);
```

9.2 Detecting Error Conditions

All of the Toolkit functions return an error/warning code. A 0 indicates that the function ran successfully. A number greater than 0 but less than 100 indicates that a warning condition was generated while a number higher than 100 indicates that the function failed.

The meaning of specific error and warning codes are listed in the Error Codes and Warning Codes sections of this guide. The Toolkit function EN_geterror can be used to obtain the text of a specific error/warning code. The following example uses a macro named `ERRCODE` along with a variable named `errcode` to execute Toolkit commands only if no fatal errors have already been detected:

```
#define ERRCODE(x) (errcode = ((errcode > 100) ? (errcode) : (x)))
void runHydraulics(EN_Project ph, char *inputFile, char *reportFile)
{
    int errcode = 0;
    char errmsg[EN_MAXMSG + 1];
    ERRCODE(EN_open(ph, inputFile, reportFile, ""));
    ERRCODE(EN_solveH(ph));
    ERRCODE(EN_saveH(ph));
    ERRCODE(EN_report(ph));
    EN_geterror(ph, errcode, errmsg);
    if (errcode) printf("\n%s\n", errmsg);
}
```

9.3 Providing Network Data

Once a project is created there are two ways in which it can be populated with data. The first is to use the EN_open function to load an EPANET-formatted Input File that provides a description of the network to be analyzed. This function should be called immediately after a project is created. It takes as arguments the name of the input file to open and the names of a report file and a binary output file, both of which are optional. Here is a code sample showing this approach:

```
EN_Project ph;
int  errcode;
EN_createproject(&ph);
errcode = EN_open(ph, "net1.inp", "net1.rpt", "");
if (errcode == 0)
{
    // Call functions that perform desired analysis
}
EN_deleteproject(ph);
```

After an input file has been loaded in this fashion the resulting network can have objects added or deleted, and their properties set using the various Toolkit functions .

The second method for supplying network data to a project is to use the Toolkit's functions to add objects and to set their properties via code. In this case the EN_init function should be called immediately after creating a project, passing in the names of a report and binary output files (both optional) as well as the choices of flow units and head loss formulas to use. After that the various **EN_add** functions, such as EN_addnode, EN_addlink, EN_addpattern, EN_addcontrol, etc., can be called to add new objects to the network. Here is a partial example of constructing a network from code:

```
int index;
EN_Project ph;
EN_createproject(&ph);
EN_init(ph, "net1.rpt", "", EN_GPM, EN_HW);
EN_addnode(ph, "J1", EN_JUNCTION, &index);
EN_addnode(ph, "J2", EN_JUNCTION, &index);
EN_addlink(ph, "P1", EN_PIPE, "J1", "J2", &index);
// additional function calls to complete building the network
```

See the Network Building Example for a more complete example. The labels used to name objects cannot contain spaces, semi-colons, or double quotes nor exceed EN_MAXID characters in length. While adding objects their properties can be set as described in the next section. Attemtping to change a network's structure by adding or deleting nodes and links while the Toolkit's hydraulic or water quality solvers are open will result in an error condition.

9.4 Setting Object Properties

The Toolkit contains several functions for retrieving and setting the properties of a network's objects and its analysis options. The names of retrieval functions all begin with **EN_get** (e.g., EN_getnodevalue, EN_getoption, etc.) while the functions used for setting parameter values begin with **EN_set** (e.g., EN_setnodevalue, EN_setoption, etc.).

Most of these functions use an index number to refer to a specific network component (such as a node, link, time pattern or data curve). This number is simply the position of the component in the list of all components of similar type (e.g., node 10 is the tenth node, starting from 1, in the network) and is not the same as the ID label assigned to the component. A series of functions exist to determine a component's index number given its ID label (see EN_getnodeindex, EN_getlinkindex, EN_getpatternindex, and EN_getcurveindex). Likewise, functions exist to retrieve a component's ID label given its index number (see EN_getlinkid, EN_getnodeid, EN_getpatternid, and EN_getcurveid). The EN_getcount function can be used to determine the number of different components in the network. Be aware that a component's index can change as elements are added or deleted from the network. The EN_addnode and EN_addlink functions return the index of the newly added node or link as a convenience for immediately setting their properties.

The code below is an example of using the property retrieval and setting functions. It changes all links with diameter of 10 inches to 12 inches.

```
void changeDiameters(EN_Project ph)
{
    int   i, nLinks;
```

```
    double diam;
    EN_getcount(ph, EN_LINKCOUNT, &nLinks);
    for (i = 1; i <= nLinks; i++)
    {
        EN_getlinkvalue(ph, i, EN_DIAMETER, &diam);
        if (diam == 10) EN_setlinkvalue(ph, i, EN_DIAMETER, 12);
    }
}
```

9.5 Computing Hydraulics

There are two ways to use the Toolkit to run a hydraulic analysis:

1. Use the EN_solveH function to run a complete extended period analysis, without having access to intermediate results.
2. Use the EN_openH - EN_initH - EN_runH - EN_nextH - EN_closeH series of functions to step through the simulation one hydraulic time step at a time.

Method 1 is useful if you only want to run a single hydraulic analysis, perhaps to provide input to a water quality analysis. With this method hydraulic results are always saved to an intermediate hydraulics file at every time step.

Method 2 must be used if you need to access results between time steps or if you wish to run many analyses efficiently. To accomplish the latter, you would make only one call to **EN_openH** to begin the process, then make successive calls to **EN_initH - EN_runH - EN_nextH** to perform each analysis, and finally call **EN_closeH** to close down the hydraulics system. An example of this is shown below (calls to **EN_nextH** are not needed because we are only making a single period analysis in this example).

```
void runHydraulics(EN_Project ph, int nRuns)
{
    int  i;
    long t;
    EN_openH(ph);
    for (i = 1; i <= nRuns; i++)
    {
        // user-supplied function to set parameters
        setparams(ph, i);
        // initialize hydraulics; don't save them to file
        EN_initH(ph, EN_NOSAVE);
        // solve hydraulics
        EN_runH(ph, &t);
        // user-supplied function to process results
        getresults(ph, i);
    }
    EN_closeH(ph);
}
```

9.6 Computing Water Quality

As with a hydraulic analysis, there are two ways to carry out a water quality analysis:

1. Use the EN_solveQ function to run a complete extended period analysis, without having access to intermediate results. A complete set of hydraulic results must have been generated either from running a hydraulic analysis or from importing a saved hydraulics file from a previous run.
2. Use the EN_openQ - EN_initQ - EN_runQ - EN_nextQ - EN_closeQ series of functions to step through the simulation one hydraulic time step at a time. (Replacing EN_nextQ with EN_stepQ will step through one water quality time step at a time.)

The second option can either be carried out after a hydraulic analysis has been run or simultaneously as hydraulics are being computed. Example code for these two alternatives is shown below:

```
int runSequentialQuality(EN_Project ph)
{
    int err;
    long t, tStep;
    err = EN_solveH(ph);
    if (err > 100) return err;
    EN_openQ(ph);
    EN_initQ(ph, EN_NOSAVE);
    do {
        EN_runQ(ph, &t);
        // Access quality results for time t here
        EN_nextQ(ph, &tStep);
    } while (tStep > 0);
    EN_closeQ(ph);
    return 0;
}
int runConcurrentQuality(EN_Project ph)
{
    int err = 0;
    long t, tStep;
    EN_openH(ph);
    EN_initH(ph, EN_NOSAVE);
    EN_openQ(ph);
    EN_initQ(ph, EN_NOSAVE);
    do {
        err = EN_runH(ph, &t);
        if (err > 100) break;
        EN_runQ(ph, &t);
        // Access hydraulic & quality results for time t here
        EN_nextH(ph, &tStep);
        EN_nextQ(ph, &tStep);
    } while (tStep > 0);
    EN_closeH(ph);
    EN_closeQ(ph);
    return err;
}
```

9.7 Retrieving Computed Results

The EN_getnodevalue and EN_getlinkvalue functions can also be used to retrieve the results of hydraulic and water quality simulations. The computed parameters (and their Toolkit codes) that can be retrieved are as follows:

For Nodes:	For Links:
EN_DEMAND (demand)	**EN_FLOW** (flow rate)
EN_DEMANDDEFICIT (demand deficit)	**EN_VELOCITY** (flow velocity)
EN_HEAD (hydraulic head)	**EN_HEADLOSS** (head loss)
EN_PRESSURE (pressure)	**EN_STATUS** (link status)
EN_TANKLEVEL (tank water level)	**EN_SETTING** (pump speed or valve setting)
EN_TANKVOLUME (tank water volume)	**EN_ENERGY** (pump energy usage)
EN_QUALITY (water quality)	**EN_PUMP_EFFIC** (pump efficiency)
EN_SOURCEMASS (source mass inflow)	

The following code shows how to retrieve the pressure at each node of a network after each time step of a hydraulic analysis (`writetofile` is a user-defined function that will write a record to a file):

```
void getPressures(EN_Project ph)
{
    int    i, numNodes;
    long   t, tStep;
    double p;
    char   id[EN_MAXID + 1];
    EN_getcount(ph, EN_NODECOUNT, &numNodes);
    EN_openH(ph);
    EN_initH(ph, EN_NOSAVE);
    do {
        EN_runH(ph, &t);
        for (i = 1; i <= NumNodes; i++) {
            EN_getnodevalue(ph, i, EN_PRESSURE, &p);
```

```
            EN_getnodeid(ph, i, id);
            writetofile(t, id, p);
        }
        EN_nextH(&tStep);
    } while (tStep > 0);
    EN_closeH(ph);
}
```

9.8 Producing a Report

The Toolkit has some built-in capabilities to produce formatted output results saved to a file. More specialized reporting needs can always be handled by writing custom code.

The EN_setreport function is used to define the format of a report while the EN_report function actually writes the report. The latter should be called only after a hydraulic or water quality analysis has been made. An example of creating a report that lists all nodes where the pressure variation over the duration of the simulation exceeds 20 psi is shown below:

```
void reportPressureVariation(EN_Project ph)
{
    // Compute ranges (max - min)
    EN_settimeparam(ph, EN_STATISTIC, EN_RANGE);
    // Solve and save hydraulics
    EN_solveH(ph);
    EN_saveH(ph);
    // Define contents of the report
    EN_resetreport(ph);
    EN_setreport(ph, "FILE myfile.rpt");
    EN_setreport(ph, "NODES ALL");
    EN_setreport(ph, "PRESSURE PRECISION 1");
    EN_setreport(ph, "PRESSURE ABOVE 20");
    // Write the report to file
    EN_report(ph);
}
```

Chapter 10

Module Index

10.1 API Reference

These topics describe the Toolkit's functions, enumerations, and error/warning codes.

Chapter 11

Module Documentation

11.1 Project Functions

These functions are used to manage a project.

These functions are used to manage a project.

11.2 Hydraulic Analysis Functions

These functions are used to perform a hydraulic analysis.

These functions are used to perform a hydraulic analysis.

11.3 Water Quality Analysis Functions

These functions are used to perform a water quality analysis.

These functions are used to perform a water quality analysis.

11.4 Reporting Functions

These functions are used to report simulation results.

These functions are used to report simulation results.

11.5 Analysis Options Functions

These functions are used to get and set analysis options.

These functions are used to get and set analysis options.

11.6 Network Node Functions

These functions are used for working with network nodes.

These functions are used for working with network nodes.

11.7 Nodal Demand Functions

These functions are used for managing nodal demands.

These functions are used for managing nodal demands.

11.8 Network Link Functions

These functions are used for working with network links.

These functions are used for working with network links.

11.9 Time Pattern Functions

These functions are used for working with time patterns.

These functions are used for working with time patterns.

11.10 Data Curve Functions

These functions are used for working with data curves.

These functions are used for working with data curves.

11.11 Simple Control Functions

These functions are used for working with simple conditional controls.

These functions are used for working with simple conditional controls.

11.12 Rule-Based Control Functions

These functions are used for working with rule-based controls.

These functions are used for working with rule-based controls.

11.13 Enumerated Types

These are the toolkit's enumerated types whose members are used as function arguments.

These are the toolkit's enumerated types whose members are used as function arguments.

11.14 Error Codes

Code	Meaning
0	No error
101	Insufficient memory available
102	No network data available
103	Hydraulic solver not opened
104	No hydraulics for water quality analysis
105	Water quality solver not opened
106	No results saved to report on
107	Hydraulics supplied from external file
108	Cannot use external file while hydraulics solver is open
110	Cannot solve network hydraulic equations
120	Cannot solve water quality transport equations
200	One or more errors in an input file
201	Syntax error
202	Function call contains an illegal numeric value
203	Function call refers to an undefined node
204	Function call refers to an undefined link
205	Function call refers to an undefined time pattern
206	Function call refers to an undefined curve
207	Function call attempts to control a check valve pipe or a GPV valve
208	Function call contains illegal PDA pressure limits
209	Function call contains an illegal node property value
211	Function call contains an illegal link property value
212	Function call refers to an undefined Trace Node
213	Function call contains an invalid option value
214	Too many characters in a line of an input file
215	Function call contains a duplicate ID label
216	Function call refers to an undefined pump
217	Invalid pump energy data
219	Illegal valve connection to tank node
220	Illegal valve connection to another valve
221	Mis-placed clause in rule-based control
222	Link assigned same start and end nodes
223	Not enough nodes in network
224	No tanks or reservoirs in network
225	Invalid lower/upper levels for tank
226	No head curve or power rating for pump
227	Invalid head curve for pump
230	Nonincreasing x-values for curve
233	Network has unconnected node
240	Function call refers to nonexistent water quality source
241	Function call refers to nonexistent control
250	Function call contains invalid format (e.g. too long an ID name)
251	Function call contains invalid parameter code
253	Function call refers to nonexistent demand category
254	Function call refers to node with no coordinates
257	Function call refers to nonexistent rule

Code	Meaning
258	Function call refers to nonexistent rule clause
259	Function call attempts to delete a node that still has links connected to it
260	Function call attempts to delete node assigned as a Trace Node
261	Function call attempts to delete a node or link contained in a control
262	Function call attempts to modify network structure while a solver is open
301	Identical file names used for different types of files
302	Cannot open input file
303	Cannot open report file
304	Cannot open output file
305	Cannot open hydraulics file
306	Hydraulics file does not match network data
307	Cannot read hydraulics file
308	Cannot save results to binary file
309	Cannot save results to report file

11.15 Warning Codes

Code	Description
1	System hydraulically unbalanced - convergence to a hydraulic solution was not achieved in the allowed number of trials
2	System may be hydraulically unstable - hydraulic convergence was only achieved after the status of all links was held fixed
3	System disconnected - one or more nodes with positive demands were disconnected from all supply sources
4	Pumps cannot deliver enough flow or head - one or more pumps were forced to either shut down (due to insufficient head) or operate beyond the maximum rated flow
5	Valves cannot deliver enough flow - one or more flow control valves could not deliver the required flow even when fully open
6	System has negative pressures - negative pressures occurred at one or more junctions with positive demand

11.16 OutFileFormat

The Toolkit uses an unformatted binary output file to store both hydraulic and water quality results at uniform reporting intervals. Data written to the file is either 4-byte integers, 4-byte floats, or fixed-size strings whose size is a multiple of 4 bytes. This allows the file to be divided conveniently into 4-byte records. The file consists of four sections of the following sizes in bytes:

Section	Size in Bytes
Prolog	884 + 36∗Nnodes + 52∗Nlinks + 8∗Ntanks
Energy Usage	28∗Npumps + 4
Dynamic Results	(16∗Nnodes + 32∗Nlinks)∗Nperiods
Epilog	28

where:

- Nnodes = number of nodes (junctions + reservoirs + tanks),
- Nlinks = number of links (pipes + pumps + valves),
- Ntanks = number of tanks and reservoirs,
- Npumps = number of pumps,
- Nperiods = number of reporting periods.

All of these counts are themselves written to the file's Prolog or Epilog sections.

11.16.0.1 Prolog Section

The Prolog section of an EPANET binary output file contains the following data:

Item	Type	# Bytes
Magic Number = 516114521	Integer	4
Version (= 200)	Integer	4
Number of Nodes	Integer	4
Number of Reservoirs & Tanks	Integer	4
Number of Links	Integer	4
Number of Pumps	Integer	4
Number of Valves	Integer	4
Water Quality Option - see EN_QualityType	Integer	4
Traced Node Index	Integer	4
Flow Units Option	Integer	4
Pressure Units Option	Integer	4
0 = psi		
1 = meters		
2 = kPa		
Report Statistic Type - see EN_StatisticType	Integer	4
Reporting Start Time (sec)	Integer	4
Reporting Time Step (sec)	Integer	4
Simulation Duration (sec)	Integer	4

Item	Type	# Bytes
Project Title (1st line)	Char	80
Project Title (2nd line)	Char	80
Project Title (3rd line)	Char	80
Name of Input File	Char	260
Name of Report File	Char	260
Name of Quality Chemical	Char	32
Chemical Concentration Units	Char	32
ID String of Each Node	Char	32∗Nnodes
ID String of Each Link	Char	32∗Nlinks
Index of Head Node of Each Link	Integer	4∗Nlinks
Index of Tail Node of Each Link	Integer	4∗Nlinks
Type Code of Each Link (see EN_LinkType)	Integer	4∗Nlinks
Node Index of Each Tank	Integer	4∗Ntanks
Surface Area of Each Tank	Float	4∗Ntanks
Elevation of Each Node	Float	4∗Nnodes
Length of Each Link	Float	4∗Nlinks
Diameter of Each Link	Float	4∗Nlinks

11.16.0.2 Energy Usage Section

The Energy Usage section of an EPANET binary output file contains the following data:

Item (Repeated for Each Pump)	Type	# Bytes
Pump Index in list of links	Integer	4
Pump Utilization (%)	Float	4
Average Efficiency (%)	Float	4
Average kwatts/MGal (or kwatts/cu m)	Float	4
Average kwatts	Float	4
Peak kwatts	Float	4
Average Cost per Day	Float	4
Peak Energy Usage (kw-hrs)	Float	4

11.16.0.3 Dynamic Results Section

The Dynamic Results section of an EPANET binary output file contains the following set of data for each reporting period (the reporting time step is written to the Output File's Prolog Section and the number of such steps is written to the Epilog Section):

Item	Type	# Bytes
Demand at Each Node	Float	4∗Nnodes
Head (Grade) at Each Node	Float	4∗Nnodes
Pressure at Each Node	Float	4∗Nnodes
Water Quality at Each Node	Float	4∗Nnodes
Flow in Each Link	Float	4∗Nlinks
(negative for reverse flow)		
Velocity in Each Link	Float	4∗Nlinks
Headloss per 1000 Units of Length for Each Link	Float	4∗Nlinks

Item	Type	# Bytes
(total head for pumps and head loss for valves)		
Average Water Quality in Each Link	Float	4∗Nlinks
Status Code for Each Link	Float	4∗Nlinks
0 = closed (pump shutoff head exceeded)		
1 = temporarily closed		
2 = closed		
3 = open		
4 = active (partially open		
5 = open (pump max. flow exceeded)		
6 = open (FCV can't supply flow		
7 = open (PRV/PSV can't supply pressure)		
Setting for Each Link	Float	4∗Nlinks
Reaction Rate for Each Link (mass/L/day)	Float	4∗Nlinks
Friction Factor for Each Link	Float	4∗Nlinks

11.16.0.4 Epilog Section

The Epilog section of an EPANET binary output file contains the following data:

Item	Type	# Bytes
Average bulk reaction rate (mass/hr)	Float	4
Average wall reaction rate (mass/hr)	Float	4
Average tank reaction rate (mass/hr)	Float	4
Average source inflow rate (mass/hr)	Float	4
Number of Reporting Periods	Integer	4
Warning Flag:	Integer	4
0 = no warnings		
1 = warnings were generated		
Magic Number = 516114521	Integer	4

Index