June 8, 2017

WASP8 Stream Transport - Model Theory and User's Guide

Supplement to Water Quality Analysis Simulation Program (WASP) User Documentation

Robert B. Ambrose, Jr., P.E.
U.S. EPA, Office of Research and Development
National Exposure Research Laboratory
Ecosystems Research Division
Athens, Georgia

Tim A. Wool
U.S. EPA, Region 4
Water Management Division
Atlanta, Georgia

U.S. Environmental Protection Agency
Office of Research and Development
Washington, DC 20460

NOTICE

The U.S. Environmental Protection Agency (EPA) through its Office of Research and Development (ORD) funded and managed the research described herein. It has been subjected to the Agency's peer and administrative review and has been approved for publication as an EPA document. Mention of trade names or commercial products does not constitute endorsement or recommendation for use.

Abstract

The standard WASP8 stream transport model calculates water flow through a branching stream network that may include both free-flowing and ponded segments. This supplemental user manual documents the hydraulic algorithms, including the transport and hydrogeometry equations, the model input and output, and a series of model verification tests.

For one-dimensional, branching streams or rivers, flow routing can be calculated for free-flowing stream reaches, for ponded reaches, and for backwater or tidally-influenced reaches. Kinematic wave flow routing is a simple but realistic option to drive advective transport through free-flowing segments. The kinematic wave equation calculates flow wave propagation and resulting variations in flows, volumes, depths, and velocities resulting from variable upstream inflow. This well-known equation is a solution of the one-dimensional continuity equation and a simplified form of the momentum equation that considers the effects of gravity and friction. Advective transport through ponded segments is controlled by a sharp-crested weir equation. This equation calculates outflow based on water elevation above the weir. Ponded reaches can include a flowing surface water segment (epilimnion) as well as stagnant underlying segments (hypolimnion).

For dynamic flow through backwater segments, the momentum equations from DYNHYD provide a simple solution for calculating outflows and resultant changes in velocity, surface elevation, depth and volume. Driven by variable upstream flows and downstream heads, the dynamic flow routine solves one-dimensional equations describing the propagation of a long wave through a shallow water system while conserving both momentum and volume. This approach considers the effects of gravity, friction, and convective inertia, assuming that flow is predominantly one-dimensional, that accelerations normal to the direction of flow are negligible, that channels can be adequately represented by a constant top width with a variable hydraulic depth (i.e., "rectangular"), and that bottom slopes are moderate. This option can be used to represent simple two-dimensional (x-y) water bodies with a branching link-node network.

To run the WASP8 stream transport module, the user must supply segment information and flow information. Required segment information includes lengths, widths, and depths for average flow conditions, as well as bottom slopes and Manning friction coefficients. The hydrogeometric depth exponents may also be specified to control the channel shape. Minimum channel depths for zero-flow conditions may be specified, with a default value of 0.001 m. Bottom slopes less than 10^{-6} signify ponded or backwater segments. For ponded segments, the minimum channel depth is interpreted as the outlet weir height. Required flow information includes the flow pathways for the main channel and each of the tributaries, as well as the inflow time functions for the simulation period.

Transport variables are provided as output for each of the WASP8 kinetic modules. Standard output includes segment outflows, depths, velocities, and widths.

Table of Contents

List of Figures

List of Tables

1 - Introduction

The Water Quality Analysis Simulation Program, WASP8 (Di Toro et al. 1983, Ambrose et al. 1988, Wool et al. 2001, Wool et al. 2006) is a general dynamic mass balance framework for modeling contaminant fate and transport in surface waters. Based on the flexible compartment modeling approach, WASP can be applied in one, two, or three dimensions with advective and dispersive transport between discrete physical compartments, or "segments." WASP provides a selection of modules to allow the simulation of conventional water quality variables as well as toxicants.

The WASP kinetic models are based on a set of transport and transformation equations. Advective transport is driven by water flow through a specified computational network (e.g. Figure 1). Inflows bring boundary concentrations into the network, and internal flows advect most constituents along specified flow paths through the network and out the downstream boundaries.

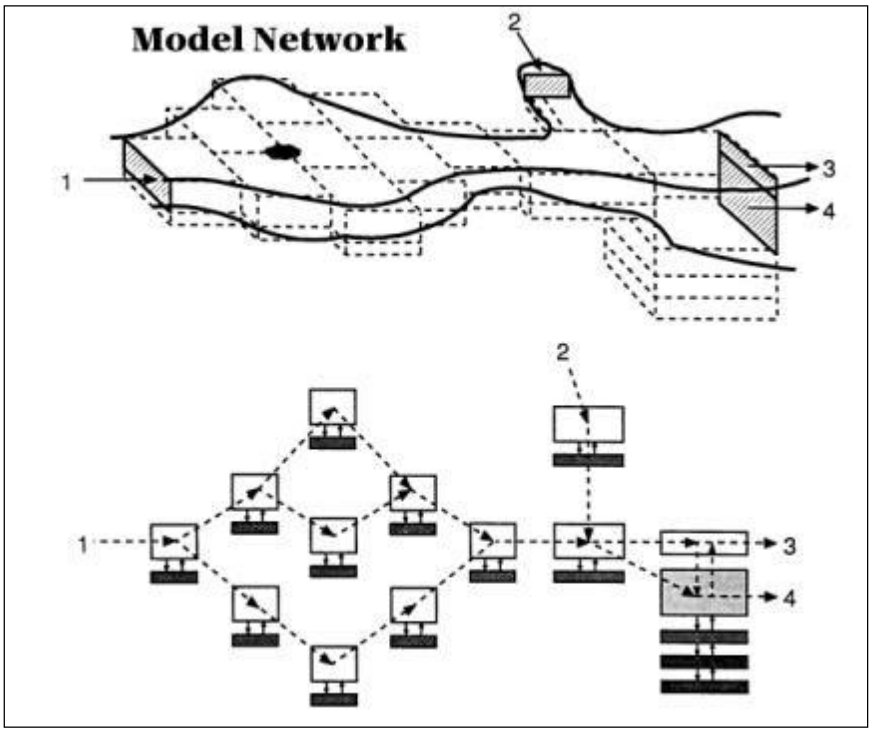

Figure 1 - Model network with advective transport pathways

2 - Background

2.1 *WASP Transport Fields*

Advective transport in WASP is divided into six distinct types, or "fields." The first transport field is advective flow in the water column. Advective flow carries water quality constituents "downstream" with the water and accounts for instream dilution. The second transport field specifies the movement of pore water in the sediment bed. Dissolved water

quality constituents are carried through the bed by pore water flow. The third, fourth, and fifth transport fields specify the transport of particulate pollutants by the settling, resuspension, and burial of solids (descriptive approach). WASP8 allows the user to specify settling and resuspension rates more directly using segment parameters. WASP8 also implements a mechanistic approach for sediment transport (see Sediment Transport User's Manual). Water quality constituents sorbed onto solid particles are transported between the water column and the sediment bed. The sixth transport field represents evaporation or precipitation from or to surface water segments.

2.2 *WASP Surface Water Flow*

Advective water column flows directly control the transport of dissolved and particulate pollutants in many water bodies. In addition, changes in velocity and depth resulting from variable flows can affect such kinetic processes as reaeration, volatilization, and photolysis. In WASP, water column flow is input via transport field 1. Circulation patterns may be described using 1 of the available 6 flow/transport options (Table 1)

Table 1 WASP8 Flow Transport Options

Flow Option	Description
Flow Routing	Uses specified flow for segment; volume, depth, and velocity are not adjusted with flow.
Stream Routing	Uses specified flow for segment; volume, depth, and velocity are adjusted for variable flow based on hydrogeometry
Kinematic Wave	Use kinematic wave routing for segment based on slope and bottom roughness
Ponded Weir	Uses weir height to control flow through assumed flat surface
Hydrodynamic Linkage	Detailed transport is provided by external hydrodynamic model
Dynamic Wave	Uses water surface elevation and surface slope to calculate flow

2.2.1 WASP Surface Water Descriptive Flow Options

Two descriptive flow options are available in WASP – Flow Routing and Stream Routing. The outflow from a descriptive flow segment is equal to the sum of the inflows to that segment. For Flow Routing, there are no adjustments in volume, depth, and velocity if inflows vary. For Stream Routing, the volume, depth, and velocity varies with flow based on specified hydrogeometry coefficients.

2.2.2 WASP Kinematic Wave Stream Flow Option

The kinematic wave stream flow option was implemented to provide a more realistic simulation of flow dynamics in branching, one-dimensional networks. Kinematic flow is controlled by bottom slope and bottom roughness. The kinematic wave formulation can be used for most stream and small river systems. WASP simulates downstream flow through the network in response to time-variable inflows and withdrawals.

As in the descriptive flow options, the user must supply both a continuity function and a time function for each inflow (or withdrawal). Flow paths may diverge (branch) and then re-join. For surface water segments, the user must specify bottom slopes and roughness factors, as well as widths and depths for average flow conditions and a depth hydraulic exponent for non-rectangular channels. The model uses the inflow and flow path functions, along with specified channel geometry and hydraulic coefficients to calculate time-variable water movement (flows and velocities) and channel hydrogeometry (top widths, cross-sectional average depths, and volumes).

Beginning with version 7.3, the stream network can include ponded flow segments along with kinematic flow segments. Flow through ponded segments is controlled by a downstream low-head dam, weir, or natural sill. For these segments, the user must specify bottom slope of 0 (or less than 10^{-6}) and the downstream weir height. Ponded segments may include stagnant underlying water layers.

Beginning with version 8.0, the stream network can include dynamic flow (or "backwater") segments along with kinematic flow and ponded flow segments. Dynamic flow is controlled by gradients in surface elevation and velocity, as well as bottom roughness. For these segments, the user must set bottom slope to 0 and specify bottom elevation in reference to a downstream control point.

3 - Development of Equations

3.1 *Hydrogeometry*

A good description of segment hydrogeometry as a function of flow can be important in properly using WASP to simulate streams and rivers. For the hydrodynamic linkage flow option, velocities and depths computed by the hydrodynamic model are used in WASP. For the internal flow options (Net Flow, Gross Flow, Kinematic Wave), a set of user-specified hydraulic discharge coefficients defines the relationship between velocity, depth, and stream flow in surface water segments. This method follows the implementation in QUAL2E (Brown and Barnwell, 1987). For the descriptive flow options (Net Flow, Gross Flow), segment velocities and depths do not influence the transport scheme; they are only used in calculations of reaeration and volatilization rates. For the Kinematic Wave flow option, segment velocities, widths, and depths are integral to the transport calculations.

Discharge coefficients giving depth and velocity from stream flow are based on empirical observations of the stream flow relationship with velocity and depth (Leopold and Maddox, 1953). The equations relate velocity, channel width, and depth to stream flow through power functions:

Equation 1

$$v = vmult \cdot Q^{v\exp}$$

Equation 2

$$R = dmult \cdot Q^{dxp}$$

Equation 3

$$B = bmult \cdot Q^{b\exp}$$

where v is velocity [m/sec], R is hydraulic radius, or cross-sectional average depth [m], B is top width [m], *vmult*, *dmult*, and *bmult* are empirical coefficients, and *vexp*, *dx*p, and *bexp* are empirical exponents. Cross-sectional area, A is the product of top width and average depth, and from continuity, flow is given by:

Equation 4

$$Q = v \cdot A = v \cdot R \cdot B = (vmult\, Q^{v\exp}) \cdot (dmult\, Q^{dxp}) \cdot (bmult\, Q^{b\exp})$$
$$= (vmult \cdot dmult \cdot bmult)\, Q^{v\exp + dxp + b\exp}$$

From inspection, the following hydraulic relationships hold:

Equation 5

$$vmult \cdot dmult \cdot bmult = 1$$

Equation 6

$$v\exp + dxp + b\exp = 1$$

The Net Flow and Gross Flow options in WASP require specification of the hydraulic relationships for velocity and depth; the width coefficients are calculated internally from Equation 5 and Equation 6. The Kinematic Wave Flow option requires specification of the hydraulic depth exponent *dxp*, along with depth D_m and width B_m under average flow conditions. Manning's equation (rearranged) is used to calculate velocity v_m under average flow conditions, and then average flow Q_m from depth, width, and velocity:

Equation 7

$$v_m = \frac{D_m^{2/3} \cdot S_f^{1/2}}{n}$$

Equation 8

$$Q_m = v_m \cdot D_m \cdot B_m$$

A consistent set of hydraulic exponents are set (see Appendix, Section 6.1):

Equation 9

$$v\exp = \left(2/3\right) \cdot dxp$$

Equation 10

$$b\exp = 1 - dxp - v\exp$$

Finally, a consistent set of hydraulic multipliers are then derived from mean flow width, the hydraulic geometry equations, and Manning's equation:

Equation 11

$$bmult = B_m \cdot Q_m^{-b\exp}$$

Equation 12

$$vmult = 1/\alpha$$

Equation 13

$$dmult = \alpha / bmult$$

Equation 14

$$\alpha = \left(\frac{n \cdot bmult^{2/3}}{S_f^{1/2}} \right)^{3/5}$$

Table 2 - Hydraulic Exponents for Figure 3

Channel	Velocity	Depth	Width
1. Rectangular	0.40	0.60	0.00
2. U-Shape	0.32	0.48	0.20
3. V-Shape	0.26	0.39	0.35
4. Shallow	0.20	0.30	0.50

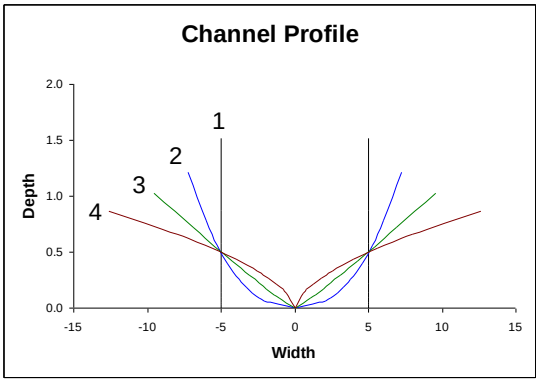

Figure 2 - Channel hydraulic cross-sections

Channel cross-sections for representative hydraulic geometry coefficients in Table 2are illustrated in Figure 2. Under mean flow, these channels are 10 m wide and 0.5 m deep. Leopold et al. (1964) have noted that stream channels in humid regions tend towards a rectangular cross-section because cohesive soils promote steep side slopes whereas noncohesive soils encourage shallow sloped, almost undefined banks.

Table 3 - Comparison of Empirical Hydraulic Exponents

Channel	Velocity	Depth	Width
Rectangular	0.40	0.60	0.00
Average of 158 U.S. Gaging Stations	0.43	0.45	0.12
Average of 10 Gaging Stations on Rhine River	0.43	0.41	0.13
Ephemeral Streams in Semiarid U.S.	0.34	0.36	0.29

Table 3 compares hydraulic exponents for a rectangular channel with data reported by Leopold et al. (1964). Note that the average velocity exponent is relatively constant for all channel cross sections. The major variation occurs as a decrease in the depth exponent and concomitant increase in the width exponent as channel cross-sections change from the steep side slopes characteristic of cohesive soils to the shallow slopes of arid regions with noncohesive soils.

For site-specific river or stream simulations, hydraulic coefficients and exponents must be estimated. Brown and Barnwell (1987) recommended estimating the exponents (b and d) and then calibrating the coefficients (a and c) to observed velocity and depth. The exponents may be chosen based on observations of channel shape noted in a reconnaissance survey. If cross sections are largely rectangular with vertical banks, the first set of exponents shown should be useful. If channels have steep banks typical of areas with cohesive soils, then the second set of exponents is appropriate. If the stream is in an arid region with typically noncohesive soils and shallow sloping banks, then the last set of exponents is recommended.

The key property of the channel that should be noted in a reconnaissance survey is the condition of the bank slopes or the extent to which width would increase with increasing stream flow. Clearly the bank slopes and material in contact with the stream flow at the flow rate(s) of interest are the main characteristics to note in a reconnaissance. This gives general guidance but it should be noted that values are derived for bankful flows. Even in streams with vertical banks, the low flows may be in contact with a sand bed having shallow sloped, almost nonexistent banks that are more representative of ephemeral streams in semi-arid areas.

3.2 *Governing Flow Equations*

The WASP stream flow model consists of a set of one-dimensional equations solving water flow and water volume in a branching stream or shallow river network. This network can include free-flowing stream reaches (kinematic wave flow), ponded reaches (weir overflow), and backwater or tidally influenced reaches (dynamic flow). The equation of motion, based on the conservation of momentum, predicts water velocities and flows. The equation of continuity, based on the conservation of volume, predicts water heights (heads) and volumes.

The one-dimensional continuity equation is given by:

Equation 15

$$\frac{\partial Q}{\partial x} + \frac{\partial A}{\partial t} = 0$$

where Q is volumetric flow, [m^3/sec] and A is cross-sectional area [m^2]. For rectangular channels, where width is constant, Equation 15 becomes:

Equation 16

$$\frac{\partial Q}{\partial x} + B\frac{\partial H}{\partial t} = 0$$

where B is channel width [m] and H is water surface elevation [m]. As presently implemented in WASP, kinematic flow reaches have shapes described by the hydrogeometric relationships described in Section 3.1, while ponded reaches and dynamic flow reaches have rectangular channel shapes.

The equations of motion implemented in the three reach types are described in the following sections.

3.2.1 Kinematic Wave Flow

For one-dimensional, free-flowing stream reaches, kinematic wave flow routing is a simple but realistic option to drive advective transport. The kinematic wave equation calculates flow wave propagation and resulting variations in flows, velocities, widths, and depths throughout a stream network. This well-known equation is a solution of the one-dimensional continuity equation and a simplified form of the momentum equation that considers the effects of gravity and friction:

Equation 17

$$g\left(S_0 - S_f\right) = 0$$

where g is acceleration of gravity [m/sec^2], S_0 is the bottom slope, and S_f is the friction slope. Manning's equation expresses the friction force as a function of water velocity and hydraulic radius:

Equation 18

$$S_f = \frac{n^2 v^2}{R^{4/3}}$$

where n is the Manning friction factor, v is water velocity [m/sec], and R is hydraulic radius [m], which is equivalent to the cross-sectional average depth, D. From the simplified momentum equation, S_0 can be equated to S_f. Hydraulic radius can be expressed as cross-sectional area divided by width, B [m]. Substituting these into the Manning's equation and rearranging terms gives flow as a function of bottom slope, cross-sectional area, and width:

Equation 19

$$Q = \frac{1}{n}\frac{A^{5/3}}{B^{2/3}} S_0^{1/2}$$

Substituting this expression into the continuity equation and differentiating *A* with respect to time gives the kinematic wave differential equation:

Equation 20

$$\frac{\partial Q}{\partial x} + \alpha \beta Q^{\beta-1} \frac{\partial Q}{\partial t} = 0$$

where, for rectangular channels:

Equation 21

$$\beta = 3/5, \qquad \alpha = \left(\frac{n B^{2/3}}{S_f^{1/2}} \right)^{3/5}$$

For channels where width varies with flow, α and β are functions of hydraulic coefficients:

Equation 22

$$\beta = 0.6 + 0.4 \cdot dxp, \qquad \alpha = \left(\frac{n \cdot bmult^{2/3}}{S_f^{1/2}} \right)^{3/5}$$

where the hydraulic coefficients *dxp* and *bmult* are defined in Section 3.1.

3.2.2 Ponded Weir Flow

For flow through ponded segments controlled by a downstream low-head dam or natural sill (Figure 3), the sharp-crested weir overflow equation is a simple solution for calculating outflows and resultant changes in depth and volume. Weir height H_w [m] and width B_w [m] are specified by the user, and hydraulic head H_h is the difference between ponded depth H and H_w.

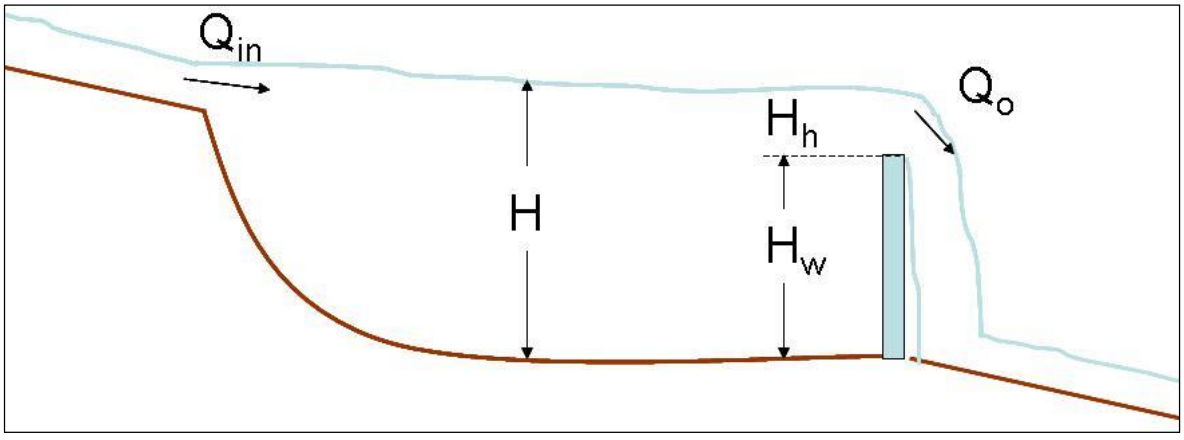

Figure 3 - Definition sketch for ponded flow

For a sharp-crested weir where $H_h/H_w < 0.4$, velocity and flow are related to head by (Finnemore and Franzini 2002):

Equation 23

$$v_o = 1.83\ H_h^{1/2}$$

$$Q_o = v_o\ A_h = 1.83\ B_w\ H_h^{3/2}$$

$$= 0 \qquad when\ H \le H_w$$

where v_o is the velocity of flow over the weir [m/sec], and A_h is the cross-sectional area of flow over the weir [m^2], given by the product of H_h and B_h.

3.2.3 Dynamic Flow

For dynamic flow through backwater segments, the momentum equations from DYNHYD provide a simple solution for calculating outflows and resultant changes in velocity, surface elevation, depth and volume. Bottom elevation H_b [m], width B [m], initial depth D [m], and initial velocity v [m/sec] are specified by the user. Surface elevation Y is the sum of D and H_b. Driven by variable upstream flows and downstream heads, simulations typically proceed at 1- to 5-minute intervals.

The dynamic flow routine solves one-dimensional equations describing the propagation of a long wave through a shallow water system while conserving both momentum and volume. The equation of motion calculates water velocities and flows. The equation of continuity calculates surface elevations, along with associated depths and volumes. This approach assumes that flow is predominantly one-dimensional, that accelerations normal to the direction of flow are negligible, that channels can be adequately represented by a constant top width with a variable hydraulic depth (i.e., "rectangular"), that the wave length is significantly greater than the depth, and that bottom slopes are moderate. Reaches with steep bottom slopes are best solved with the kinematic wave equations.

Equation 16 gives the equation of continuity implemented for dynamic flow reaches. The equation of motion calculates local acceleration, the velocity rate of change with respect to time [m/sec^2]:

Equation 24

$$\frac{dv}{dt} = -v\frac{dv}{dx} + a_g + a_f$$

where v is water velocity [m/sec], a_g is gravitational acceleration along the axis of the channel [m/sec^2], and a_f is frictional acceleration [m/sec^2]. The first term, convective inertia or Bernoulli acceleration, represents the rate of momentum change by mass transfer, [m/sec^2]. The second term, gravitational acceleration, is driven by the slope of the water surface:

Equation 25

$$a_g = -g\frac{\partial Y}{\partial x}$$

where Y is surface elevation [m], and g is the acceleration of gravity [9.81 m/sec^2]. The third term, frictional acceleration, can be expressed using the Manning equation for steady uniform flow:

Equation 26

$$a_f = -g \frac{n^2 v^2}{R^{4/3}}$$

where n is the Manning friction factor, v is water velocity [m/sec], and R is hydraulic radius [m], which is equivalent to the cross-sectional average depth, D.

Local wind acceleration is not included in this implemented of the dynamic flow equations.

3.3 *Implementation of Equations*

WASP8 solves the kinematic flow, ponded flow, and dynamic flow equations for appropriate surface water segments in a stream network using finite-difference formulations for flow and for continuity.

For each segment, a maximum numerical time step DT_{max} is calculated from the segment length and characteristic velocity, as described in the sections below. The overall time step is the product of the minimum DT_{max} in the network and a user-specified fraction, DTF (default = 0.9) that is set to ensure stability:

Equation 27

$$DT = DTF \cdot \min\left(DT_{\max}\right)$$

This time step DT is divided into two half time steps. For kinematic wave reaches and ponded weir reaches, flows are calculated sequentially for each half time step and then averaged for subsequent use by the water quality module. Final velocities, depths, volumes, and surface elevations at the end of the full time step are passed along to the water quality module.

For dynamic flow reaches, the two half time steps are used in a predictor-corrector scheme as described in Section 3.3.3 below. Flows, velocities, volumes, depths, and surface elevations are updated following each of the numerical steps. Final flows, velocities, volumes, depths, and surface elevations at the end of the full time step are passed along to the water quality module.

3.3.1 Kinematic Wave Flow

To solve for flow in kinematic wave reaches, Equation 20 is expressed in finite difference form:

Equation 28

$$\frac{Q_{Dt} - Q_{D0}}{DT} = \frac{Q_{Dt}^{1-\beta}}{\alpha \beta L}\left(Q_U - Q_{D0}\right)$$

where Q_U is the upstream inflow, Q_{D0} is the outflow from the preceding time step, Q_{Dt} is the outflow for this time step, and DT is the time step [days]. Equation 28 is solved using a Newton-Raphson approach:

Equation 29

$$f(Q_{Dt}) \quad = \quad Q_{Dt} + c_1 Q_{Dt}^{1-\beta} + c_2 \quad = \quad 0, \qquad c_1 = -\frac{DT}{\alpha \beta L}\left(Q_U - Q_{D0}\right), \quad c_2 = -Q_{D0}$$

Equation 30

$$f'(Q_{Dt}) \quad = \quad 1 + c_1 (1-\beta) Q_{Dt}^{-\beta}$$

Given an initial estimate of Q_{Dt}, an updated estimate, Q_{Dt2}, is calculated by:

Equation 31

$$Q_{Dt2} \quad = \quad Q_{Dt} - \frac{f\ (Q_{Dt})}{f'(Q_{Dt})}, \qquad err = \left| \frac{Q_{Dt2} - Q_{Dt}}{Q_{Do}} \right|$$

Equation 29, Equation 30, and Equation 31 are solved in an iterative loop where Q_{Dt} it set equal to Q_{Dt2} until *err* is less than 10^{-5}. Given the new set of flows for the water column network $Q_{Dt,ij}$, volumes for all water segments "*i*" are updated using the continuity equation:

Equation 32

$$DV_i \quad = \quad \sum_j Q_{Dt,ij}\ DT, \qquad V_{t,i} \quad = \quad V_{0,i} + DV_i$$

Segment widths are updated with the new flows using Equation 3. Associated cross-sectional areas and depths are then calculated:

Equation 33

$$DA_i \quad = \quad \frac{DV_i}{L_i}, \qquad A_{t,i} \quad = \quad A_{0,i} + DA_i$$

Equation 34

$$D_{t,i} \quad = \quad \frac{A_{t,i}}{B_i}$$

To prevent slow numerical drift in calculated volumes during lengthy simulations, small adjustments are made to flows based on differences between hydraulic radius calculated from Equation 2 and cross-sectional average depth calculated through continuity. Applying Equation 19:

Equation 35

$$Q_{err,i} \quad = \quad Q_{Dt,i} \quad - \quad Q_{R,i} \quad = \quad \frac{B_i}{n_i} S_{0,i}^{1/2} \left(D_i^{5/3} - R_i^{5/3} \right)$$

Equation 36

$$Q_{Dt,i} \quad = \quad Q_{Dt,i} \quad + \quad Q_{err,i}$$

For each kinematic flow segment, a maximum stable numerical time step DT_{max} [days] is calculated from the segment length *L* [m] and celerity *c* [m/s]:

Equation 37

$$c = v / \beta$$

Equation 38

$$DT_{\max} = \frac{0.5}{86400} \cdot \left(L \diagup c \right)$$

where 0.5 is a safety factor.

3.3.2 Ponded Weir Flow

For ponded reaches "*i*", weir overflow (Equation 23) must be solved along with continuity such that:

Equation 39

$$Q_{Dt,i} = 1.83\, B_{w,i}\, H_{ht,i}^{3/2}$$

and

Equation 40

$$H_{ht,i} = H_{h0,i} + \frac{Q_{Ut,i} - Q_{Dt,i}}{B_i\, L_i} DT$$

where $Q_{Ut,i}$ and $Q_{Dt,i}$ are the upstream inflow and outflow for this time step [m^3/sec], $H_{ht,i}$ is the head for this time step [m], $H_{h0,i}$ is the head from the previous time step [m], and DT is the time step [days]. These equations are solved using a Newton-Raphson approach where:

Equation 41

$$f(Q_{Dt}) = Q_{Dt} - 1.83\, B\, H_{ht}^{3/2} = 0$$

Equation 42

$$f'(Q_{Dt}) = 1 - 1.83\, B \left(\frac{3}{2}\, H_{ht,i}^{1/2}\ \frac{DT}{B_i L_i} \right)$$

$$= 1 + \frac{1.5 \cdot 1.83 \cdot DT}{L_i} H_{ht,i}^{1/2}$$

Given an initial estimate of $Q_{Dt,i}$, a consistent value for $H_{ht,}$is calculated using Equation 40. An updated estimate Q_{Dt2}, is then calculated by:

Equation 43

$$Q_{Dt2} = Q_{Dt} - \frac{f\ (Q_{Dt})}{f'(Q_{Dt})}, \qquad err = \left| \frac{Q_{Dt2} - Q_{Dt}}{Q_{Do}} \right|$$

Equation 41, Equation 42, Equation 43, and Equation 40 are solved in an iterative loop where Q_{Dt} and H_{ht} are set equal to Q_{Dt2} and H_{ht2} until *err* is less than 10^{-5}.

WASP8 solves this weir overflow equation for each ponded segment in a stream network. In each of the numerical steps, calculated values of dQ/dt and Q_o are used to update volumes, depths, and H_h, which are used in the next numerical step to calculate Q_o.

For each weir overflow segment, a maximum stable numerical time step DT_{max} [days] is calculated from the segment length and overflow velocity v_o:

Equation 44

$$DT_{\max} = \frac{0.5}{86400} \cdot \left(L \Big/ 1.5 v_o \right)$$

where 0.5 and 1.5 are safety factors.

3.3.3 Dynamic Flow

Equation 16 and Equation 24 form the basis of the hydrodynamic model DYNHYD5, which is implemented within this version of WASP. These equations are integrated numerically on a flexible, computationally efficient "link-node" network (Feigner and Harris, 1970), which solves the equations of motion and continuity at alternating grid points. At each time step, the equation of motion is solved at the links (or "channels"), giving velocities for mass transport calculations, and the equation of continuity is solved at the nodes (or "junctions"), giving heads for pollutant concentration calculations. Link-node networks can treat fairly complex branching flow patterns and irregular shorelines with acceptable accuracy for many studies. They cannot handle stratified water bodies, small streams, or rivers with a large bottom slope. Link-node networks can be set up for wide, shallow water bodies if primary flow directions are well defined.

In WASP, nodes correspond to segments, and links correspond to segment interfaces. For every dynamic flow segment in a WASP network, a distinct channel number "*ich*" is defined for each of its downstream segments. Channel *ich* is defined by upstream segment *j* and downstream segment *i*. Positive flow in channel *ich* is outflow from segment *j* and inflow to segment *i*. Negative flow in channel *ich* is outflow from segment *i* to segment *j*.

In finite difference form, Equation 24 is given by:

Equation 45

$$\frac{v_{t,ich} - v_{ich}}{DT} = v_{ich} \frac{\Delta v_{ich}}{\Delta x_{ich}} - g \frac{\Delta H_{ich}}{\Delta x_{ich}} - \frac{g n_{ich}^2}{R_{ich}^{4/3}} v_{ich} \left| v_{ich} \right|$$

where $v_{t,ich}$ is the velocity for this time step [m/sec], v_{ich} is the velocity from the preceding time step [m/sec], Δx_{ich} is channel *ich* length [m], $\Delta v_{ich} / \Delta x_{ich}$ is the velocity gradient in channel *ich* with respect to distance [sec^{-1}], $\Delta H_{ich} / \Delta x_{ich}$ is the water surface gradient in channel *ich* with respect to distance [m/m], and DT is the time step [days]. All values on the right hand side of equation 20 are referenced to the previous time step.

The water surface gradient can be computed from the junction heads at either end of the channel:

Equation 46

$$\frac{\Delta H_{ich}}{\Delta x_{ich}} = \frac{H_j - H_i}{(L_j + L_i)/2}$$

where H_j and L_j are the water surface elevation and length of the upstream segment[m], and H_i and L_i are the water surface elevation and length of the downstream segment[m].

The velocity gradient cannot be computed directly from upstream and downstream channel velocities because of possible branching in the network. If branching does occur, there would be several upstream and downstream channels, and any computed velocity gradient would be ambiguous. An expression for the velocity gradient within a channel can be derived by applying the continuity equation to the channel and substituting $v \times A$ for Q:

Equation 47

$$\frac{\partial A}{\partial t} = -\frac{\partial Q}{\partial x} = -v\frac{\partial A}{\partial x} - A\frac{\partial v}{\partial x}$$

Rearranging terms gives the channel velocity gradient:

Equation 48

$$\frac{\partial v}{\partial x} = -\frac{1}{A}\frac{\partial A}{\partial t} - \frac{v}{A}\frac{\partial A}{\partial x}$$

Writing this in finite difference form and substituting $B \times R$ for A and $B \times \Delta H$ for $M\!A$ gives the velocity gradient term:

Equation 49

$$\frac{\Delta v_{ich}}{\Delta x_{ich}} = -\frac{1}{R_{ich}}\frac{\Delta H_{ich}}{\Delta t} - \frac{v_{ich}}{R_{ich}}\frac{\Delta H_{ich}}{\Delta x_{ich}}$$

The term $\Delta H_{ich}/\Delta t$ is computed as the average water surface elevation change between the segments at each end of channel *ich* during time step *t*. Substituting Equation 49 into Equation 45 and rearranging gives the explicit finite difference equation of motion applied to each channel:

Equation 50

$$v_{t,ich} = v_{ich} + DT\left[\frac{v_{ich}}{R_{ich}}\frac{\Delta H_{ich}}{\Delta t} + \left(\frac{v_{ich}^2}{R_{ich}} - g\right)\frac{\Delta H_{ich}}{\Delta x_{ich}} - \frac{g n_{ich}^2}{R_{ich}^{4/3}} v_{ich}|v_{ich}|\right]$$

Writing the equation of continuity in finite difference form and rearranging terms gives:

Equation 51

$$H_{t,j} = H_j - DT\frac{\sum_i Q_{ij}}{B_j L_j}$$

Equation 50 and Equation 51 are solved using a 2-step predictor-corrector routine. Based on initial velocities, surface elevations, and depths from the previous time step, new

velocities and flows are solved for the half time step, along with new surface elevations and depths. Using these predicted half-time step values, velocity and flow derivatives are recalculated for the half time step. These corrected derivatives are then used with the initial velocities, depths, and surface elevations to calculate velocities and flows for the full time step. Finally, surface elevations, depths, and volumes are calculated for the full time step.

For each dynamic flow segment, a maximum stable numerical time step DT_{max} [days] is calculated from the segment length L [m] and celerity c [m/s]:

Equation 52

$$c = \sqrt{g\,D}$$

Equation 53

$$DT_{\max} = \frac{0.5}{86400} \cdot \left(L/c\right)$$

where 0.5 is a safety factor.

4 - Stream Transport Model Inputs

To implement stream flow routing, the user must specify information in the Data set screen, the Segments screen, and the Flows screen, accessed from the gears, the cube, and the faucet on the main WASP toolbar (Figure 4). Each of these is briefly described in the sections below.

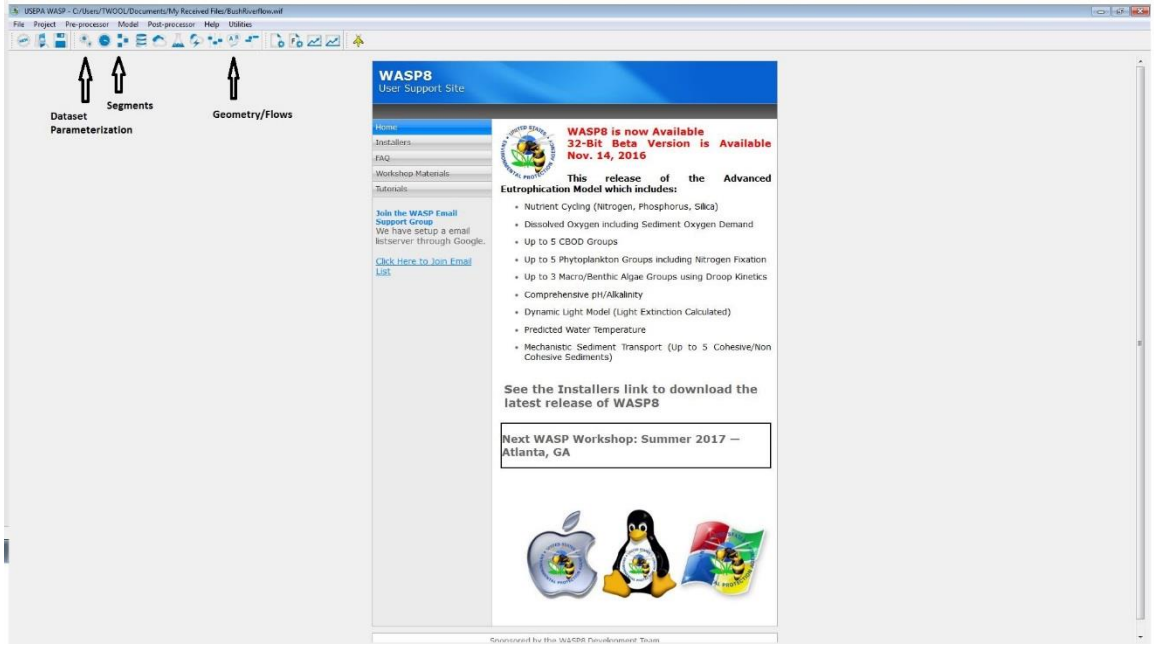

Figure 4 - WASP Main Screen Toolbar, data input buttons

4.1 *Data set screen*

In the Data set screen (Figure 5), the user must select the Model Type. In the Time Range section, the user must specify the simulation Start Date, Start Time, End Date and End

Time. In the hydrodynamics section, the user must select the flow option. Finally, the Time Step information should be specified. Default values are supplied for Fraction of max time step (0.9), Max time step (1.0 day), and Min time step (0.0001 day). These values should work well for most cases, but if numerical instability is encountered, lowering the Fraction of max time step (to 0.5 or even 0.1) could help. In some cases, the user may want to specify a maximum times step of less than a day. If diurnal output is desired, then a maximum time step of 0.1 or 0.05 days should give the necessary precision.

When creating a new input dataset the input parameterization data entry form is the first one that needs to be completed. This form provides basic information that is needed by the program to parameterize the other data entry forms that follow. This screen informs the program what type of WASP file you are going to be creating.

4.1.1 Restart Options

The methods used by WASP to read and create restart files have changed substantially in this version. In previous versions the user would have selected Create Restart File, for WASP to write the final conditions of the simulation to an output file. This is true for the current version as well. If the user wants to restart a simulation with the final conditions of previous simulation this radio must be set. At the end of the WASP simulation a restart file with the same name as the WIF except with the extension *.RST will be saved. With the current release of WASP if the user wants to use a restart file they simple click on the Load Restart File button, this will allow the user to browse to whatever restart file they want to use. Once the file is selected and the user clicks on the Okay button, the restart file is opened up and segment volumes and state variable initial conditions are reset to the values in the user selected *.RST file.

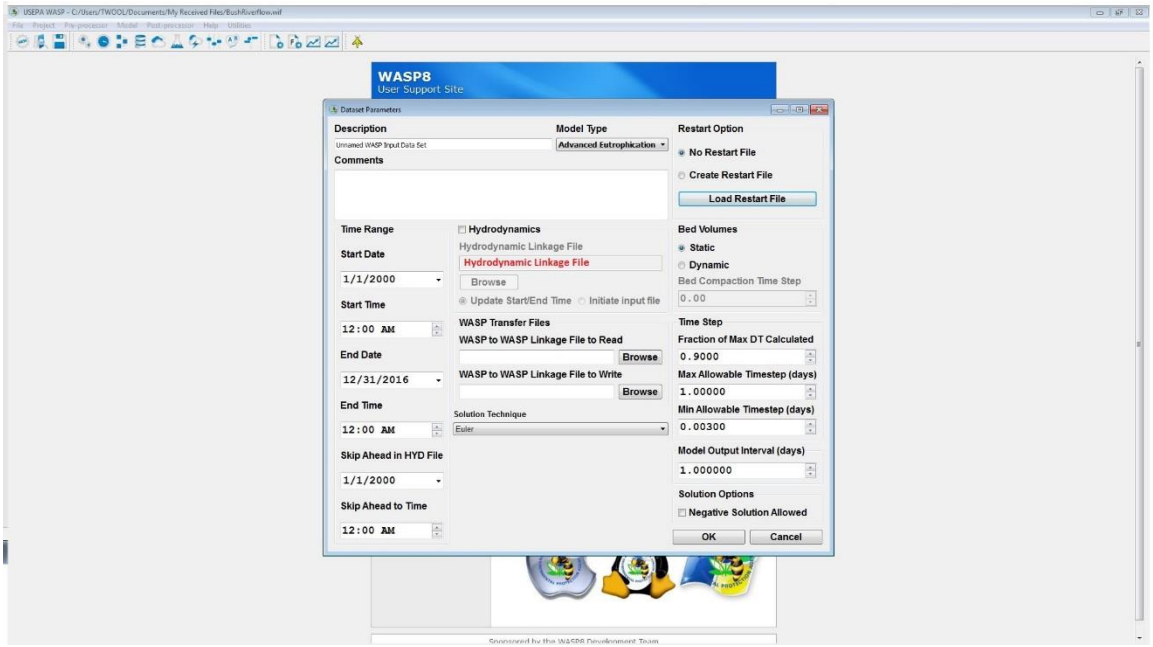

Figure 5 Dataset Parameterization

4.1.2 Date and Times

The previous versions of WASP did not require that the model time functions be represented in Gregorian date format. WASP requires all time functions be represented in Gregorian fashion (mm/dd/year hh:mm:ss).

Start Date and Start Time - The start time dialog is used to define the date and time for the start of the simulation or time period being considered in the model input files. This date and time correspond to time zero within the model.

End Date and End Time - The end time dialog is used to define the date and time when the simulation will end.

Skip Ahead to Date and Time - This new addition to the WASP interface allows the user to skip to any portion of the simulation and/or the selected loaded hydrodynamic linkage file. When the user selects a hydrodynamic linkage file the start time and end time of the file is read and the interface automatically sets the beginning and end time to these values. It is best that the user build all of the time series (environmental, boundary and dispersion) to cover this full range of time. Once the WIF is built the user can set a date and time to skip the simulation from the start time set in the hydrodynamic file. This is handy for using the whole hydrodynamic linkage file for calibration and verification, and then using a small portion of the hydrodynamic linkage file for scenario analysis. It could be the critical time period that will be used for the waste load allocation or TMDL. The start time of the simulation should still be set the beginning time in the hydrodynamic linkage file. The user can change the end time of the simulation by changing the last date/time pair in the Time Step screen.

4.1.3 Hydrodynamics

Hydrodynamic Linkage -- Realistic simulations of unsteady transport in rivers, reservoirs, and estuaries can be accomplished by linking WASP8 to a compatible hydrodynamic simulation. This linkage is accomplished through an external "hyd" file chosen by the user at simulation time. The hydrodynamic file contains segment volumes at the beginning of each time step, and average segment interfacial flows during each time step. WASP8 uses the interfacial flows to calculate mass transport, and the volumes to calculate constituent concentrations. Segment depths and velocities may also be contained in the hydrodynamic file for use in calculating reaeration and volatilization rates. Before using hydrodynamic linkage files with WASP, a compatible hydrodynamic model must be set up for the water body and run successfully, creating a hydrodynamic linkage file with the extension of *.hyd. This is an important step in the development of the WASP input file because the hydrodynamic linkage file contains all necessary network and flow information. When Hydrodynamic Linkage is selected in the Data Set Parameters screen, the user cannot provide any additional surface flow information. When you are ready to begin the development of a WASP input deck, simply open the hydrodynamic linkage file from within the data preprocessor. The hydrodynamic linkage dialog box allows the user to browse and select a hydrodynamic linkage file. The data preprocessor will open the hydrodynamic interface file and extract the number of segments, the starting and ending time. Once a hydrodynamic linkage file is selected in the data preprocessor, the user will

have to add the model systems to use. User can check the numerical stability of the hydrodynamic linkage by inspecting the Mass Check system on the runtime grid and in model output. If the simulation is run for a sufficient duration, mass check concentrations should approach 1.0 mg/L throughout the network. If you are getting a number other than 1 mg/L, you may have to use a different time step in the hydrodynamic model. This is especially true if the concentrations are oscillating between large and small numbers, a clear indication of numerical instability. WASP has the ability to get hydrodynamic information from a host of hydrodynamic models. If a hydrodynamic model does not support the WASP linkage it is relative straightforward to create a hydrodynamic linkage file (see Appendix 3 for file format). The hydrodynamic models that currently support the WASP8 file format are: EFDC (three dimensions), DYNHYD (one dimension branching), RIVMOD (one dimension no branching, CE-QUAL-RIV1 (one dimension branching), SWMM/Transport (one dimension branching, SWMM/Extran (one dimension branching)

4.1.4 Solution Technique

The user now has the ability to select the model solution technique to be used by the water quality module during the simulation. Currently there are 3 solution techniques that can be selected: 1) Euler – which is the traditional solution technique that has been in WASP since its inception, 2) COSMIC Flux Limiting – this solution technique is typically used when WASP is linked to multi-dimensional hydrodynamic models like EFDC, 3) Runge-Kutta 4 step solution technique used for diurnal simulations.

4.1.5 Time Step Definition

Starting with WASP Version 7.3 the user no longer has control over the computational time step. Time step optimization routines have been refined to the point where the model can determine what the most appropriate time step should be used next. This assures the most efficient run time as well as minimizing numerical dispersion caused by too small of a time step. While the user can not set the time step directly, they do have some control over what would be an acceptable time step.

Fraction of Maximum Time Step - This dialog box specifies what fraction of the model calculated time step will be used for the next time step. Its primary purpose is aid the user in keeping the model stable. The default is 0.9 (or 90%) of the optimal time step.

Maximum Time Step - This specifies the maximum time step that will be used. If the time step optimizer calculates a time step larger than this value, this value will be used. This could be important in constraining the time step for diurnal or daily calculations.

Minimum Time Step - This specifies the minimum time step that will be used. The default minimum time step is defined in the model as 0.0001 days. Use this dialog to raise the minimum time step.

4.2 *Segments screen*

In the Segments screen, the user must enter a row for each segment in the model network (e.g., by pressing "Insert," by pressing the down arrow from the bottom row or by copying

from spreadsheet). Provide segment type, orientation and transport option to use for segment.

4.2.1 Segment Name

User must provide a unique segment name for each segment. These segment names are used to identify the segment in post processor. If the user wants to paste segment descriptions/names that include spaces, the segment description must be placed in quotes (i.e. "I 20 Bridge").

4.2.2 Segment Type

Segment type is entered using a pick list. Four segment types are available: "Surface," "Subsurface," "Surface Benthic," and "Subsurface Benthic." The default segment type is "Surface," which represents upper water column segments in contact with the atmosphere. "Subsurface" represents underlying water column segments. "Surface Benthic" represents the upper benthic sediment segments in contact with the water column. "Subsurface Benthic" represents underlying benthic segments.

4.2.3 Bottom Segment

Bottom segment is the segment immediately underneath the current segment. The bottom segment is entered by typing in the segment number or by using a pick list. If no segments are underneath the current segment, then the bottom segment is designated "none."

4.2.4 Transport Mode

There are currently six surface flow options available for WASP. The user has the ability select different transport modes for individual or groups of segments. The only restriction when using a hydrodynamic linkage file, is if linked to a hydrodynamic model all segments has be first two options pertain to how WASP will calculate the exchange of mass between adjoining segments with flow in both directions across a segment interface. The three flow options available for surface water flow are:

1. Stream Routing -- WASP will calculate net transport across a segment interface that has opposing flow. WASP will net the flows and move the mass from the segment that has the higher flow leaving. If the opposed flows are equal no mass is moved.
2. Flow Routing -- Pertains to mass and water being moved without regard to net flow.
3. Kinematic Wave -- For one-dimensional, branching streams or rivers, kinematic wave flow routing is a simple but realistic option to drive advective transport. The kinematic wave equation calculates flow wave propagation and resulting variations in flows, volumes, depths, and velocities throughout a stream network.
4. Dynamic Flow – For one-dimensional flow controlled by surface water slope. Capable of calculating backwater flow.

5. Ponded Weir – For one-dimensional flow where water surface is flat and controlled by height of weir.
6. Hydrodynamic Linkage – For multi-dimensional flow where all transport information is provided via a hydrodynamic model. This option utilizes the Hydrodynamic Application Program Interface (API), described in Appendix 3: Hydrodynamic Linkage File API.

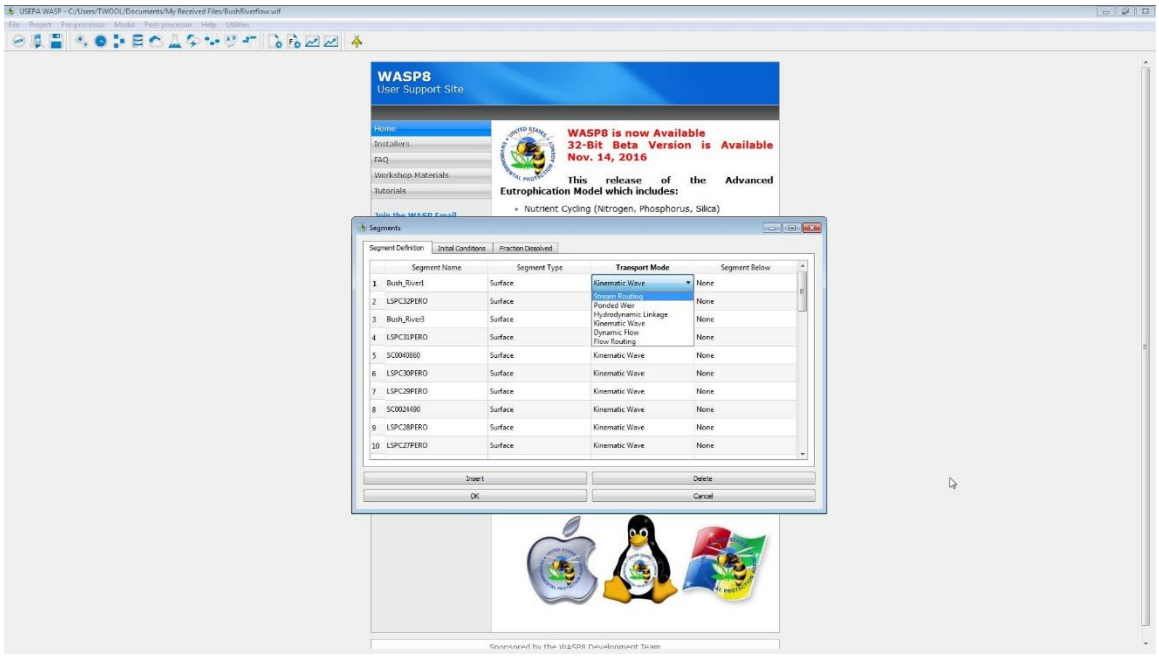

Figure 6 Segment Definition Screen

4.3 *Channel Geometry*

Depending upon the Transport Option selected for a given segment, the data entry screen will highlight the columns where data is not needed. The user should provide information for the columns that are not shaded. Segment volumes [m^3] should be specified when using the Flow and Stream routing. Channel geometry information is assumed to be at average flow conditions. Much of this geometry information can be obtained from the National Hydrography Dataset (NHDPlus). If a segment volume is not entered (or is 0), then WASP will calculate that volume from specified segment length, width, and depth.

For the 1-D Network Kinematic Wave option, input volumes are only used for benthic segments. WASP calculates initial water column segment volumes from length, width, and depth under initial flow conditions.

When using the hydrodynamic linkage flow option, initial water column volumes are read from the external hydrodynamic file. Only benthic segment volumes must be entered.

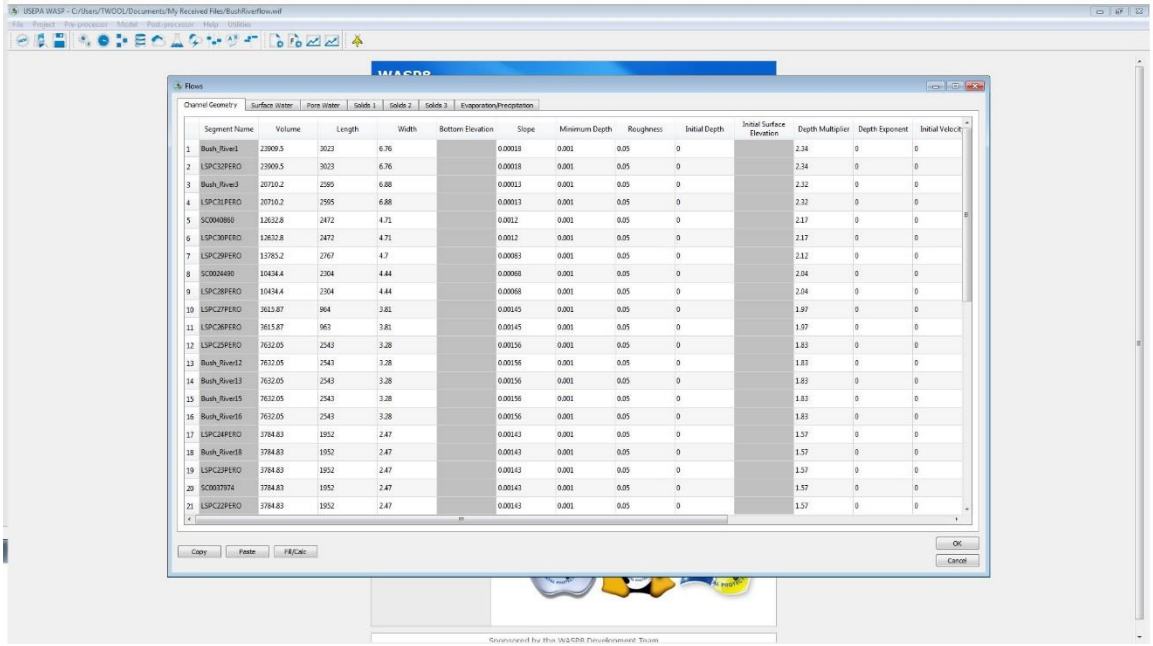

Figure 7 - WASP Segment Definition Screen

4.3.1 Volume

Segment volume (cubic meters) required for the Stream and Flow Routing option. Will be recalculated for all other options.

4.3.2 Length

Segment length [m] is the bottom length along the center of the flow line from the upstream end to the downstream end of the segment.

4.3.3 Width

Segment width [m] is the top width averaged along the length of the segment. If no input volume is specified, then width is used along with length and depth (multiplier) to calculate an initial volume.

For the 1-D Network Kinematic Wave option, width should be specified for average flow conditions. Average widths are used along with average depths, depth exponents, slopes, and roughness coefficients to back-calculate a consistent set of hydraulic coefficients.

4.3.4 Bottom Elevation

Bottom elevations values represent the vertical distance from the segment bottom (cross-sectional average) to the bottom of the downstream control segment, which is either a weir or a boundary.

4.3.5 Slope

Segment slope [m/m] is the elevation drop divided by length averaged over the segment length. This is usually calculated as the upstream elevation minus the downstream elevation divided by segment length. Slope is used only in the 1-D Network Kinematic Wave option.

4.3.6 Minimum Depth

Minimum depth [m] is the average segment depth under zero-flow conditions, used only in the 1-D Network Kinematic Wave option. If this cell is left blank, a default minimum depth of 0.001 m is assigned internally. Total depth is hydraulic depth plus minimum depth.

4.3.7 Segment Roughness

Segment roughness is the Manning's roughness coefficient *n*. Roughness is used in the 1-D Network Kinematic Wave option for kinematic wave flow and dynamic flow segments. Roughness coefficients should usually be between 0.01 and 0.15. If a coefficient of 0 is input for a free-flowing segment, WASP will reset the coefficient to 0.05 and issue a message to the screen.

4.3.8 Initial Depth

Represents the initial depth (m) of the segment at average flow.

4.3.9 Initial Surface Elevation

Represents the initial surface elevation of the segment for the dynamic wave option. Initial depth and surface slope is calculated from Bottom Elevation and initial Surface Elevation.

4.3.10 Depth (multiplier and exponent)

The depth hydraulic multipliers [m / (m^3/sec)] and exponents should be specified when using the Net Flow, Gross Flow, or Kinematic Wave options. Depth multipliers are required for all segments. For benthic segments, the depth multipliers are interpreted as segment depths [m].

For the Net Flow and Gross Flow options, the depth multipliers and exponents are used along with initial segment flows to calculate initial segment depths. If a depth multiplier is left at 0, it is reset internally to 1.0 and a message is issued to the screen. If a depth exponent is left at 0, then the depth multiplier is equal to the initial segment depth [m]. During simulations using these descriptive flow options, changing flows do not directly change segment depths, even if the hydraulic exponent is nonzero. Depths are recalculated along with volumes based on flow continuity. If a segment outflow continuity multiplier is equal to the inflow continuity multiplier, then changing flows will not alter that segment's volume or depth.

For free-flowing segments in the 1-D Network Kinematic Wave option, the depth multiplier is taken to be the cross-sectional average segment depth under average flow conditions [m]. The depth exponent is a value generally between 0.3 and 0.6. If a segment depth exponent is left at 0, a rectangular cross-section is assumed and the exponent is reset internally to 0.6. The average depths and depth exponents are used along with segment

widths, slopes, and roughness factors to calculate consistent hydraulic coefficients, which are then used to calculate segment depths under initial flow conditions. During simulations, changing flows directly change hydraulic depths based on the hydraulic coefficients. Total segment depth is equal to the hydraulic depth plus the user-input zero-flow minimum depth.

4.3.11 Velocity (multiplier and exponent)

The velocity hydraulic multipliers [(m/sec) / (m^3/sec)] and exponents should be specified only when using the Net Flow or Gross Flow options. For the Kinematic Wave option, the velocity multipliers and exponents are internally calculated from the input depth multipliers and exponents, and the input width. Any input velocity multiplier or coefficient will be ignored when using this option.

4.4 *Flows screen*

The Flows screen is used to define advective transport, including surface water and pore water flow, as well as solids settling and resuspension, precipitation and evaporation. The Flows screen is also used to define downstream boundary elevations and two-dimensional channel networks for the Dynamic Flow option.

The flow input screen is a complex screen that contains four tables (Figure 8). The upper left quadrant is used to select the transport field, such as "Surface Water" flow. For each transport field selected, the upper right quadrant is used to define a set of transport functions, including upstream and tributary inflows. For each transport function, the bottom two quadrants are used to define the flow path and the flow time function.

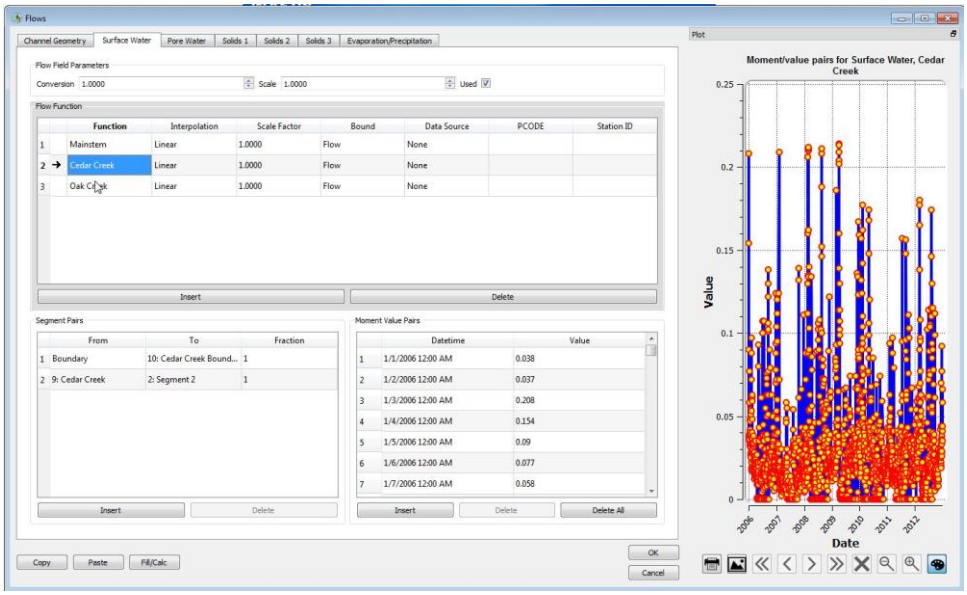

Figure 8 - Flows screen

For surface water and pore water transport, the upstream inflow, each tributary inflow, pore water inflow, and any flow withdrawals must be described by continuity path functions and inflow time functions. An example is shown in Figure 9.

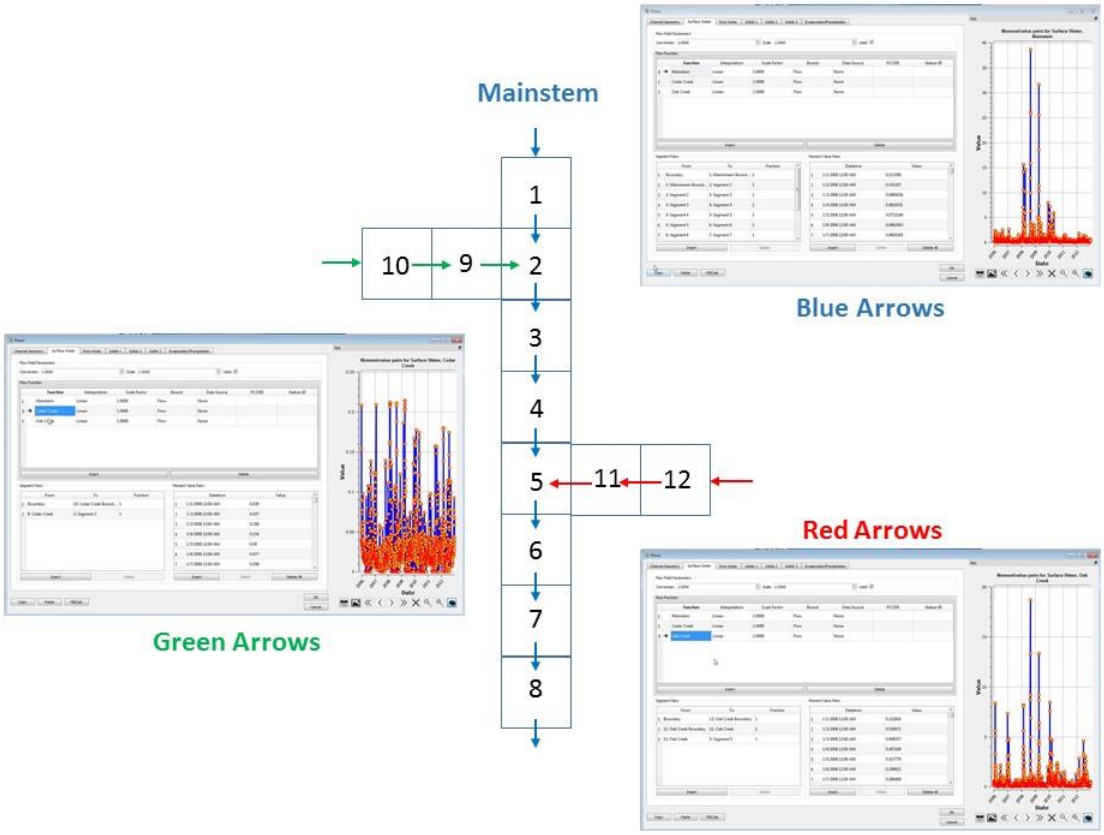

Figure 9 - Example WASP flow input

4.4.1 Flow Field

The transport field must be selected. Six transport fields are available:

1. Surface Water – This transport field is used to describe surface water flows. These flows transport both the particulate and dissolved fractions of a constituent. If the user has selected the hydrodynamic linkage option they will not be able to enter information here.

2. Pore Water – This transport field is used to describe pore water flows. These flows transport only the dissolved fraction of a constituent.

3. Solids 1 – This transport field is used to describe solids type 1 settling and resuspension. These flows transport only the particulate fraction of a constituent that is mapped to solid type 1 in the Systems screen.

4. Solids 2 – This transport field is used to describe solids type 2 settling and resuspension. These flows transport only the particulate fraction of a constituent that is mapped to solid type 2 in the Systems screen.

5. Solids 3 – This transport field is used to describe solids type 3 settling and resuspension. These flows transport only the particulate fraction of a constituent that is mapped to solid type 3 in the Systems screen.

6. Evaporation/Precipitation – This transport field subtracts/adds water from the model network. No constituent mass is added, removed, or transported.

Scale Factor – The scale factor for a transport field multiplies all flows associated with that field by the input value. This is generally used to scale flows in sensitivity tests. The default value is 1.0.

Conversion Factor – The conversion factor for a transport field multiplies all flows associated with that field by the input value. This is generally used to adjust input flow units to the internal units of m^3/sec. If flows are specified in ft^3/sec, the conversion factor should be 0.02832. The default value is 1.0.

4.4.2 Flow Function

The user can define several flow functions for the selected transport field. Each flow function must have its own flow path function (lower left table) and flow time function (lower right table). Normally, a Flow Function defines a discrete inflow, such as upstream flow, tributary flow, or pore water flow. Special flow functions are also used in conjunction with the Dynamic Flow option to define downstream boundary elevations or two-dimensional (x-y) channel networks.

To insert a flow function, first highlight the Surface Water flow field in the upper left table, then move the cursor to the upper right quadrant and click on the insert button. The resulting flow function cell, labeled "Flow Function," can be edited to provide a descriptive name.

To insert additional flow functions, either click on "insert" or highlight the last flow function and press the down arrow. To delete a flow function, select the function by highlighting the row and click on the delete button. Deleting a flow function will delete the corresponding flow path function (lower left table) and flow time function (lower right table).

Function Name – When a flow function is inserted, it is given the default name "Flow Function." The Function cell can be edited to provide a descriptive name, such as "upstream inflow," "tributary inflow," "downstream elevation," or "channel network."

Interpolation Option – The default interpolation option for the flow time function associated with a flow function is "Linear." This can be changed to "Step" to provide for a step function. To change options, click in the Interpolation cell, press the down arrow, and select the interpolation option for this flow function.

Bound Option – The default boundary type for the time function is Flow (cms). The user can also specify water surface elevation used in the dynamic wave transport option. User can also specify an internal flow boundary where water is moved between the defined segments without using a transport option.

Once a flow function is selected and named, the user must define the associated flow path function and flow time function. Be sure that the correct flow field and flow function are highlighted before entering these next screens.

4.4.3 Flow Path Function

The flow path function traces this flow from its point of entry into the model network to its point of exit either from the model network or to an alternate pathway associated with another flow function. The flow path consists of a set of rows, corresponding to segment interfaces. Each row will have a set of segment pairs and a fraction of flow multiplier.

Segment Pairs - The segment pairs consist of a "From" segment and a "To" segment, and define the direction of flow across this segment interface. Either the "From" or the "To" segment can be defined as "Boundary." Normally the first row will define the inflow from "Boundary" to the upstream segment and the last row will define the outflow from the downstream segment to "Boundary." If this flow path is a tributary, then the last row will define the outflow from the downstream tributary segment to a segment in another tributary or the main stem of the river.

Positive values of flow transport water and constituent mass in the defined direction from the first segment to the second segment. Negative flows transport water and constituent mass from the second segment to the first segment. For example, if "From" is segment 1 and "To" is segment 2, then negative values of flow in the time function will cause transport from 2 to 1. Note: While the kinematic wave option checks to make sure that all flow paths are ultimately connected to outflows, neither the preprocessor nor the model can assure that the segments are connected properly. Connectivity is the responsibility of the user.

Fraction of Flow - The fraction of flow column defines what fraction of the total flow in this pathway moves between these segment pairs. For surface water flow, the fraction of flow is normally 1.0. This allows the user to split flows from one segment into two or more downstream directions. This can be used to define diverging and converging flows, but must be used carefully. The sum of all fractions entering each segment must normally equal the sum of all fractions leaving. If the sum is greater than 1.0, then that segment's volume will continually increase. If the sum is less than 1.0, then that segment's volume will continually decrease; if the volume reaches 0, the simulation will end badly.

Note for downstream boundary elevation – If a downstream boundary elevation function is being defined for a dynamic flow network, then the flow path should consist of a single segment pair from "Boundary" to the downstream segment number. The fraction of flow should be set to 0.

Note for channel network – If a two-dimensional channel network is being defined for the Dynamic Flow option, then a channel network must be defined by one or more special Flow Functions. In a channel network flow function, each segment pair defines a unique flow channel. The fraction of flow multipliers must be set to 0. Channel lengths and cross-

sectional areas for each segment pair are read from the Exchanges screen (Section 4.5). Channel hydraulic radius is calculated internally as the average of the upstream and the downstream segment depths. Channel width is calculated internally as the cross-sectional area divided by hydraulic radius. The channel network must not include any boundaries. Channels connect two segments within the model network. WASP distinguishes channel networks from traditional flow paths by the absence of boundaries and flow path multipliers of 0. The flow time function associated with a channel network is not used, and flow values can be left at 0.

4.4.4 Flow Time Function

The flow time function is a table consisting of dates, times, and inflow values [m^3/sec]. Each row in the table represents a single point in time. During a simulation, inflows are interpolated between these points based on the flow function interpolation option selected (see Section 4.4.2). At least two rows must be entered in the flow time function to allow for interpolation.

Date – As with other WASP time functions, the date must be entered as mm/dd/year (e.g., 01/01/2004). The first date in the time function should correspond with the Start Date specified in the Data Set Screen (Section 4.1.2). The last date in the time function normally will correspond with the End Date.

Time – The time must be entered as hh:mm (e.g., 14:30).

Value – The inflow for this date and time is specified in units of [m^3/sec]. Different units can be used if a conversion factor is provided with the Flow Field (Section 4.4.1). Note that if a downstream boundary elevation function is being defined, then the value entered will be surface elevation [m].

The time function table allows the user to enter time variable flow information. For constant flows, two rows should be specified with the simulation start and end dates, and the constant flow value. The user can enter the information by hand, paste in from a spreadsheet, or query in from database/spreadsheets.

4.5 *Exchanges screen*

The Exchanges screen is used to define dispersive transport, including surface water and pore water mixing. The Exchanges screen is also used to define channel lengths and initial cross-sectional areas for channel networks.

The Exchanges input screen is a complex screen that contains four tables (Figure 10). The upper left quadrant is used to select the transport field, such as “Surface Water” flow. For each transport field selected, the upper right quadrant is used to define a set of transport functions, including lateral and longitudinal dispersion functions that provide data for any channel networks defined in the Flow functions (Section 4.4.3). For each exchange

function, the bottom two quadrants are used to define the exchange path and the exchange time function.

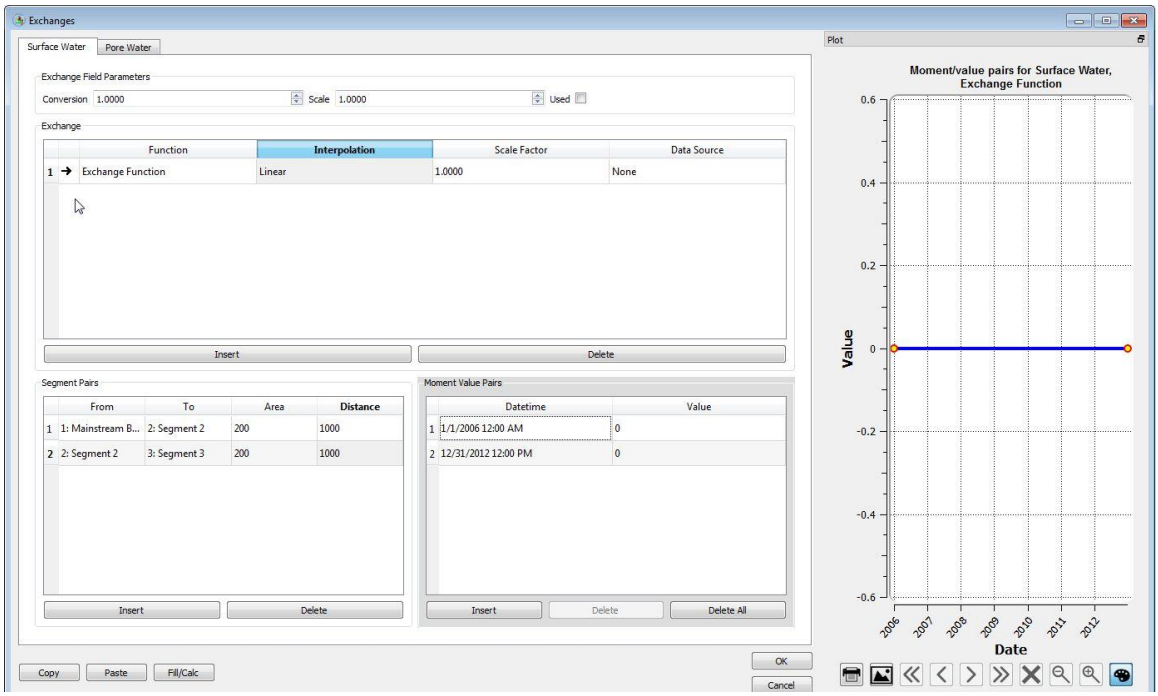

Figure 10 - Exchanges Screen

4.5.1 Exchange Field

The exchange field must be selected. Two transport fields are available:

1. Surface Water – This transport field is used to describe surface water mixing. These turbulent flows transport both the particulate and dissolved fractions of a constituent. Pore Water – This transport field is used to describe pore water mixing. These flows transport only the dissolved fraction of a constituent.

Scale Factor – The scale factor for a transport field multiplies all exchanges associated with that field by the input value. This is generally used to scale exchanges in sensitivity tests. The default value is 1.0.

Conversion Factor – The conversion factor for a transport field multiplies all exchanges associated with that field by the input value. This is generally used to adjust input dispersion coefficient units to the internal units of m^2/sec. If dispersion coefficients are specified in cm^3/sec, the conversion factor should be 0.0001. The default value is 1.0.

4.5.2 Exchange Function

The user can define several exchange functions for the selected transport field. Each exchange function must have its own exchange path function (lower left table) and dispersion time function (lower right table). Normally, an Exchange Function defines a type of exchange, such as lateral or longitudinal dispersion or surface water-pore water exchange. Exchange functions are also used in conjunction with the Dynamic Flow option

to define channel lengths and initial cross-sectional areas for two-dimensional (x-y) channel networks.

To insert an exchange function, first highlight the Surface Water field in the upper left table, then move the cursor to the upper right quadrant and click on the insert button. The resulting flow function cell, labeled "Exchange Function," can be edited to provide a descriptive name.

To insert additional exchange functions, either click on "insert" or highlight the last flow function and press the down arrow. To delete an exchange function, select the function by highlighting the row and click on the delete button. Deleting an exchange function will delete the corresponding exchange path function (lower left table) and disperson time function (lower right table).

Function Name – When an exchange function is inserted, it is given the default name "Exchange Function." The Function cell can be edited to provide a descriptive name, such as "lateral dispersion," "longitudinal dispersion," "pore water exchange," or "channel network."

Interpolation Option – The default interpolation option for the exchange time function associated with an exchange function is "Linear." This can be changed to "Step" to provide for a step function. To change options, click in the Interpolation cell, press the down arrow, and select the interpolation option for this exchange function.

Once an exchange function is selected and named, the user must define the associated exchange path function and exchange time function. Be sure that the correct exchange field and exchange function are highlighted before entering these next screens.

4.5.3 Exchange Path Function

The exchange path function specifies a set of dispersive exchange flows. The function consists of a set of rows, corresponding to segment interfaces (or "channels" in a channel network). Each row will have a set of segment pairs, a cross-sectional area, and a characteristic mixing length.

Segment Pairs – Each discrete exchange pathway is defined by a set of two segments between which exchange flows occur. Either "Segment One" or "Segment Two" can be defined as "Boundary." Neither the preprocessor nor the model can assure that the segments are connected properly. Connectivity is the responsibility of the user.

Cross-Sectional Area, m^2 - Cross-sectional areas are specified for each dispersion coefficient, reflecting the area through which mixing occurs. These can be surface areas for vertical exchange, such as in lakes or in the benthos. Areas are not modified during the simulation to reflect flow changes.

Mixing Length, m – Mixing lengths or distance are specified for each dispersion coefficient, reflecting the characteristic length over which mixing occurs. These are typically the lengths between the center points of adjoining segments. A single segment may have three or more mixing lengths for segments adjoining longitudinally, laterally, and vertically. For surficial benthic segments connecting water column segments, the depth of the benthic layer is a more realistic mixing length than half the water depth.

Note for channel network – If a two-dimensional channel network is being defined for the Dynamic Flow option, then cross-sectional areas and mixing lengths entered here will be assigned to corresponding channels defined in the Flow Path screen (Section 4.5.3).

4.5.4 Exchange Time Function

The exchange time function is a table consisting of dates, times, and dispersion coefficient values [m^2/sec]. Each row in the table represents a single point in time. During a simulation, dispersion coefficients are interpolated between these points based on the dispersion function interpolation option selected (see Section 4.5.2). At least two rows must be entered in the flow time function to allow for interpolation.

Date – As with other WASP time functions, the date must be entered as mm/dd/year (e.g., 01/01/2004). The first date in the time function should correspond with the Start Date specified in the Data Set Screen (Section 4.1.2). The last date in the time function normally will correspond with the End Date.

Time – The time must be entered as hh:mm (e.g., 14:30).

Value – The inflow for this date and time is specified in units of [m^2/sec]. Different units can be used if a conversion factor is provided with the Exchange Field (Section 4.5.1). Dispersive mixing coefficients may represent pore water diffusion in benthic segments, vertical diffusion in lakes, and lateral and longitudinal dispersion in large water bodies. Values can range from 10^{-10} m^2/sec for molecular diffusion to 5×10^2 m^2/sec for longitudinal mixing in some estuaries.

The time function table allows the user to enter time variable exchange information. For constant exchanges, two rows should be specified with the simulation start and end dates, and the constant dispersion coefficient value. The user can enter the information by hand, paste in from a spreadsheet, or query in from database/spreadsheets.

5 - References

Ambrose, R. B., J. L. Martin, and T. A. Wool, 2006. Wasp7 Benthic Algae - Model Theory and User's Guide. U.S. Environmental Protection Agency, Washington, DC, EPA/600/R-06/106 (NTIS PB2007-100139), 2006.

Wool T.A., R.B. Ambrose, J.L. Martin, and E.A. Comer, 2001. The Water Quality Analysis Simulation Program, WASP6; Part A: Model Documentation. U.S. Environmental Protection Agency, Center for Exposure Assessment Modeling, Athens, GA.

Ambrose, R.B., Jr., T.A. Wool, and J.L. Martin, 1993. The Water Quality Analysis Simulation Program, WASP5; Part A: Model Documentation. Internal Report Distributed by USEPA Center for Exposure Assessment Modeling, U.S. Environmental Protection Agency, Athens, GA.

Ambrose R.B., T.A. Wool, J.P. Connolly, and R.W. Schanz, 1988. WASP4, A Hydrodynamic and Water Quality Model—Model Theory, User's Manual, and

Programmer's Guide. EPA/600/3-87-039, U.S. Environmental Protection Agency, Athens, GA.

Di Toro DM, J.J. Fitzpatrick, and R.V. Thomann, 1983. Water Quality Analysis Simulation Program (WASP) and Model Verification Program (MVP) - Documentation. Contract No. 68-01-3872, Hydroscience Inc., Westwood, NY, for U.S. EPA, Duluth, MN.

Chapra, S.C. 1997. *Surface Water-Quality Modeling*, McGraw-Hill, New York, New York, 844 pp.

Chapra, S.C. 2003. QUAL2K: A Modeling Framework for Simulating River and Stream Water Quality (Beta Version): Documentation and Users Manual. Civil and Environmental Engineering Dept., Tufts University, Medford, MA.

Chapra, S.C. and G.J. Pelletier. 2004. QUAL2K: A Modeling Framework for Simulating River and Stream Water Quality, Version 1.3: Documentation and Users Manual. Civil and Environmental Engineering Dept., Tufts University, Medford, MA.

Feigner, K.D. and H.S. Harris, 1970. Documentation Report – FWQA Dynamic Estuary Model. U.S. Department of Interior, Federal Water Quality Administration.

Wool, T.A., R.B. Ambrose, and J. L. Martin. 2001. "The Water Analysis Simulation Program, User Documentation for Version 6.0," Distributed by USEPA Watershed and Water Quality Modeling Technical Support Center, Athens, GA.

6 - Appendix 1: Derivation of Equations

6.1 *Hydraulic Exponents for Kinematic Wave Flow*

Manning's formula (Equation 19) provides the basis for deriving relationships among the hydraulic exponents. Rearranging terms gives cross-sectional area as a function of flow:

Equation 54

$$A = n^{3/5} S_0^{-3/10} B^{2/5} Q^{3/5}$$

Cross-sectional average depth (hydraulic radius) is given by:

Equation 55

$$d = \frac{A}{B} = n^{3/5} S_0^{-3/10} B^{-3/5} Q^{3/5}$$

Velocity is given by:

Equation 56

$$v = \frac{Q}{A} = n^{-3/5} S_0^{3/10} B^{-2/5} Q^{2/5}$$

Substituting Equation 3 for width as a function of flow gives:

Equation 57

$$d = n^{3/5} S_0^{-3/10} bmult^{-3/5} Q^{3/5(1-b\exp)}$$

Equation 58

$$v = n^{-3/5} S_0^{3/10} bmult^{-2/5} Q^{2/5(1-b\exp)}$$

Equation 57 and Equation 58 give the depth and velocity hydraulic exponents as a function of the width exponent:

Equation 59

$$dxp = 0.6\,(1-b\exp) \qquad v\exp = 0.4\,(1-b\exp)$$

Comparing the flow exponents confirms that the velocity exponent is 2/3 of the depth exponent, confirming Equation 9.

7 - Appendix 2: Model Verification Tests

Model verification tests were designed to assure that the equations are implemented correctly in the model code. Results are stored in separate folders at:

- \WASP7\QA\Stream Transport\ 4-Stream Kinematic Wave Flows\

Verification tests are outlined below. Results are detailed in a companion document.

7.1 *Kinematic Wave Tests*

7.1.1 Stream Transport Test 1

This series tests the kinematic wave flow routines in WASP using the Simple Toxicant module with intermediate slope and steady inflow with an example problem taken from Chapra, Example 14.6, pp. 253, 254. Results are compared with analytical solutions implemented in a spreadsheet, and by simple hand calculations.

- Test 1a – Upstream Inflow Only
- Test 1b – Upstream and Pore Water Inflow
- Test 1c – Upstream and Precipitation Inflow
- Test 1d – Upstream Inflow and Evaporation Outflow

7.1.2 Stream Transport Test 2

This series tests the kinematic wave flow routines in WASP using the Heat module with shallow slope and step changes in inflow. Results for each flow step are compared with analytical calculations calculated in a spreadsheet.

- Test 2a – Rectangular Cross-Section
- Test 2b – U Cross-Section
- Test 2c – V Cross-Section

7.1.3 Stream Transport Test 3

This series tests the kinematic wave flow routines in WASP using the Eutrophication module with shallow slope, U-shape cross-section, step changes in upstream inflow, and inflow or withdrawal at Segment 3.

- Test 3a – Constant Tributary Inflow
- Test 3b - Constant Flow Withdrawal

7.1.4 Stream Transport Test 4

This series tests the kinematic wave flow routines in WASP using the Mercury module with a branching stream system. A medium stream with moderate slope is connected to a small tributary with shallow slope.

- Test 4a – Step Inflows
- Test 4b - Long-Term, Variable Inflows

Test 4a specifies step changes in upstream and tributary inflows. Results for each flow step are compared graphically and to analytical solutions from a spreadsheet. Test 4b uses variable upstream and tributary inflows repeating in a 2-year pattern over a long simulation period. Results are examined graphically for stationary (repeating) output.

7.1.5 Stream Transport Test 5

This tests the kinematic wave flow routines in WASP using the Heat module with a diverging-converging stream system. A small river with steep slope diverges into two branches receiving 40% and 60% of the upstream flow. These branches converge downstream. This test uses step changes in the upstream inflow. Results for each flow step are compared graphically and to analytical solutions from a spreadsheet.

7.2 *Weir Overflow Verification Tests*

Model verification tests were designed to assure that the equations are implemented correctly in the model code. Results are stored in separate folders at:

- \WASP7\QA\Stream Transport\4-Ponded Weir Flows\

Tests are outlined below.

7.2.1 Weir Overflow Test 1 – Steady Flow

This tests the ponded weir overflow routine in WASP using the Mercury module with steady inflow and sequentially increasing weir heights. Results are compared with analytical calculations.

7.2.2 Weir Overflow Test 2 – Variable Flow

This tests the ponded weir overflow routine in WASP using the Mercury module with sequentially increasing inflow and sequentially increasing weir heights. Results are compared with analytical calculations.

7.2.3 Weir Overflow Test 3 – Long Term Dynamics

This tests weir overflow routines in WASP for long-term performance with variable inflows repeating in a 2-year pattern. Results are examined for long-term drift in depths, volumes, and velocities.

7.3 *Dynamic Flow Verification Tests*

Model verification tests were designed to assure that the equations are implemented correctly in the model code. Results are stored in separate folders at:

- \WASP7\QA\Stream Transport\4-Dynamic Flows\

Tests are outlined below.

7.3.1 Dynamic Flow Test 1 - Steady Backwater

This tests the dynamic flow routine in WASP with mild slope, steady inflow, and downstream pond with weir. Weir height is set to provide downstream ponded depth equal to the kinematic flow depth. Results are compared with output from an equivalent kinematic wave simulation.

7.3.2 Dynamic Flow Test 2 – Steady Flow, Elevation

This tests the dynamic flow routine in WASP with mild slope, steady inflow, and constant downstream boundary elevation. Results are compared with output from an equivalent DYNHYD simulation linked with WASP.

7.3.3 Dynamic Flow Test 3 – Variable Stream Flow

This tests the dynamic flow routine in WASP using the Mercury module with sequentially increasing flow. Results are compared with output from a DYNHYD simulation linked with WASP.

7.3.4 Dynamic Flow Test 4 – EFDC Stream

This tests the dynamic flow routine in WASP with mild slope, steady inflow, and constant downstream boundary elevation. Results are compared with output from an equivalent EFDC simulation linked with WASP.

7.3.5 Dynamic Flow Test 5 – Tidal Stream

This tests the dynamic flow routine in WASP with flat slope, no inflow, and sinusoidal tidal downstream boundary elevation. Results are compared with output from equivalent DYNHYD and EFDC simulations linked with WASP.

7.3.6 Dynamic Flow Test 6 – 2-D Tidal Stream

This tests the dynamic flow routine in WASP on a 2-dimensional network with flat slope, no inflow, and sinusoidal tidal downstream boundary elevation. Results are compared with output from an equivalent EFDC simulation linked with WASP.

8 Appendix 3: Hydrodynamic Linkage File API

his application program interface function was developed to efficiently allow multi-dimensional hydrodynamic and sediment transport models to pass simulation information to WASP. The WASP developers believe it is best to separate the hydrodynamic and water quality models and facilitate the soft linkage of the algorithms through this hydrodynamic linkage file. This allows hydrodynamic model developers an easy method for linking their models with the WASP kinetics. The linkage allows hydrodynamic models provide transport information to all of the WASP modules.

The purpose of this document is to give an overview of the API and describe the various functions that are available. The details of the format and storage of the hydrodynamic linkage file is totally controlled by the API

8.1 *General Concept*

Figure 11 provides a schematic of how a hydrodynamic model would interact with the API to create a hydrodynamic linkage file and how WASP interacts with the same API to read the information. There are two methods that could be employed by the hydrodynamic model developer:

In-Line Code – embed the calls to the API that creates the hydrodynamic model. This allows the model to create the hydrodynamic linkage file as the hydrodynamic model simulates through time.

Post process and output file. Basically, the hydrodynamic model writes a file using its own structure. Then a utility program like HYDROLINK.EXE that is distributed with WASP will read this file and create the hydrodynamic linkage file.

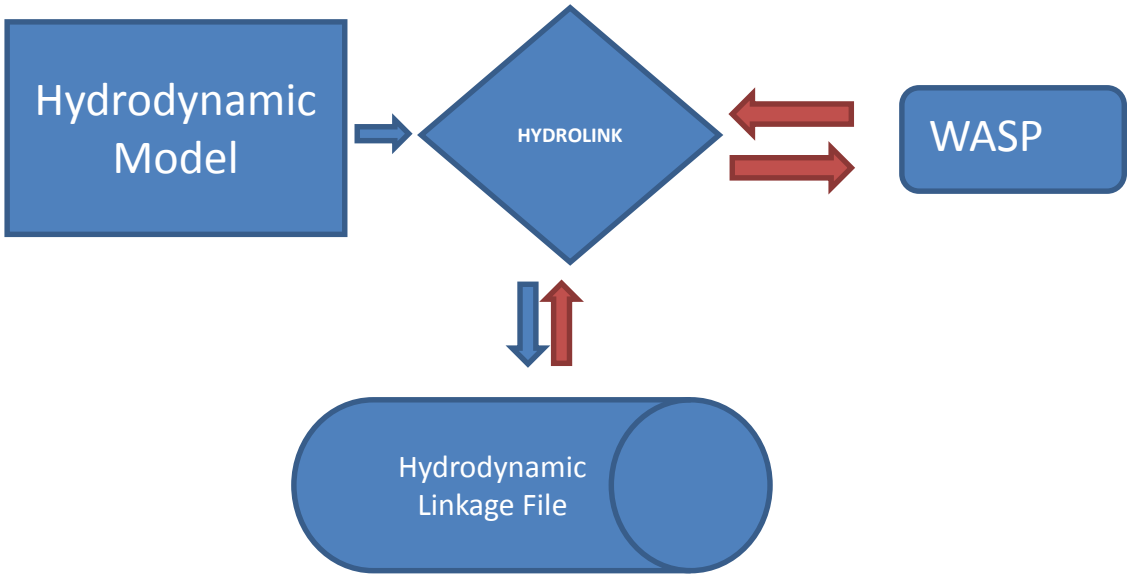

Figure 11 HYDROLINK Overview

8.2 *Application Program Interface (API) Overview*

The API allows other model developers to link their hydrodynamic models with WASP without worrying about format or changes made within the WASP framework. The API when implemented correctly will allow the creation of a correctly formatted linkage file that WASP can read.

There are several steps that need to be completed to initialize and ultimately write a correct file. Figure 12 illustrates the various stages in building the file. The initialization block controls the creation of the file, specifying the time range, number of segment, number of flow paths, and assorted switches which control the options of the file. All options will be discussed below. The main portion of the API is the writing of time variable segment information (volumes, depths, velocities, salinity, and temperature) from the hydrodynamic model for use by WASP. After segment information is written, flows and dispersion are written for each of the flow paths simulated in the hydrodynamic model.

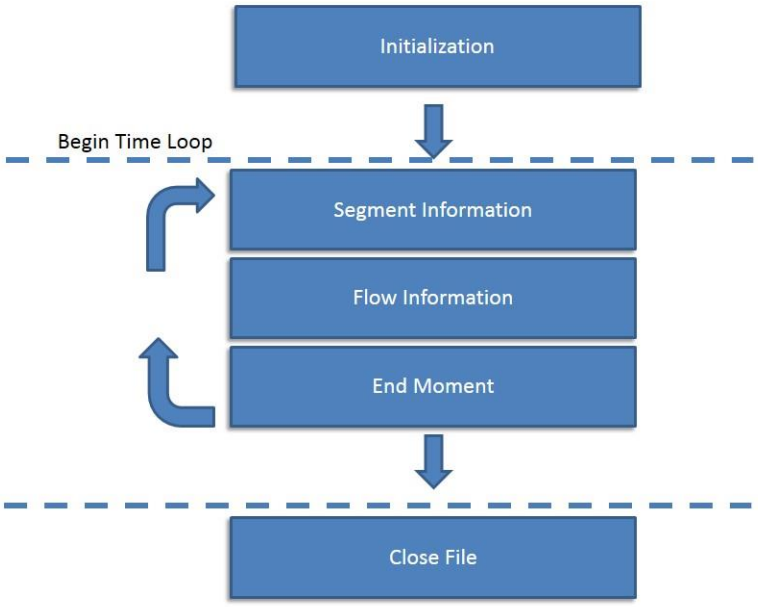

Figure 12 API Components

8.3 *Initialization*

This initialization block must be completed prior to saving information in the time loop. Many of the initialization calls will control how the information will be stored, in particular which language type will be used to create or read the information. C++ array indices range from 0 to number of values, while FORTRAN indices range from 1 to number of values, it is critical that these be set correctly because the API uses pointers to save and send array data. It is recommended that you use the order as described below.

8.3.1 Call Hlopen (Hlfile, Ihl_mode, Ihl_handle,Ierror)

This routine is used to initialize a hydrodynamic linkage file or open one for reading depending on how ihl_mode is set (see option in use). If this call is successful (Ierror =0), the files is either created (open for writing) or open for reading an important variable is set that is needed in virtually every other call to the API. Hlopen will return a file handle (Ihl_handle), this integer will be to be stored.

Where:

- Hlfile – this is a character string that contains the path and filename of the hydrodynamic linkage file.
- Ihl_mode – is an integer (I4 or Short integer) that specifies writing (create) or reading. 0 = Read, 1 = write.
- Ihl_handle – this value is assigned once the call to Hlopen is successful. This is an integer value that is used to reference the file that was open or created in the call to Hl_open. Note that you can have more than one file open in the API.

- Ierror – virtually all of the calls to the API will return an error code. If there is no error this integer will return a 0, a number greater than 0 means there was an error. The user can get a description of the error by calling Hlgetlasterror (described below).

8.3.2 Call Hlsetlanguage (Ihl_handle, Ilanguage, Ierror)

This routine sets the language type that will be writing to the hydrodynamic linkage file. Currently the switches are between C and FOTRAN. For C array indices start at 0 and two/three dimensional arrays vary from left to right. For FORTRAN arrays start at 1, and arrays vary right to left. The default is C.

Where:

- Ihl_handle – the file handle assigned by Hlopen (integer)
- Ilanguage – integer value that sets the language of the writing or reading program (0 = C++, 1 = FORTRAN)
- Ierror -- If there is no error this integer will return a 0, a number greater than 0 means there was an error.

8.3.3 Call Hlgetlasterror (ErrorString)

This method is used to return a string describing the last error encounter by the API. After ever call to an API function the user should check the status of error returned by the calling function. If the error status is greater than 0 there was an error. To get an error message Call Hlgetlasterror.

Where:

- Errstring – this is a character string that will receive the error message

8.3.4 call Hladdescription (Ihl_handle, 0 ,Description(I) , Ierror)

This function allows the user to add descriptions to the hydrodynamic linkage file that can be displayed at run time in WASP. There is no limit to the number of description lines that you can add.

Where:

- Ihl_handle– the file handle assigned by Hlopen (integer)
- Iline – this is the indices of the description being saved.
- Description(I) – description (string)
- Ierror -- If there is no error this integer will return a 0, a number greater than 0 means there was an error.

8.3.5 Call Hlsetcreator (Ihl_handle, Modtype, Ierror)

This function is used to inform WASP what the of hydrodynamic model was used to create the linkage file. Currently WASP recognizes four linkage types: 1) Environmental Fluids

Dynamic Code (EFDC), 2) 1-Dimensional Dynamic Flow Model (DYNHYD), 3) EPD-RIV1 Model, 4) WASP to WASP linkage

Where:

- Ihl_handle -- the file handle assigned by Hlopen (integer
- Modtype – this is an integer designated for the type of hydrodynamic model that created the linkage file.
- Ierror -- If there is no error this integer will return a 0, a number greater than 0 means there was an error.

8.3.6 call hlsetseedmoment(Ihl_handle, istartmonth, istartday, istartyear, istarthour, istartminute, istartsecond, ierror)

This function is used to set the initial time and date for th hydrodynamic information in the linkage file. WASP will automatically set the start and end date of the water quality model simulation. The API will increment time from the seed time as information is added to the hydrodynamic linkage file.

Where:

- Ihl_handle -- the file handle assigned by Hlopen (integer
- istartmonth – this is an integer month designation.
- Istartday – this is an integer day designation.
- Istartyear – this is an integer year designation
- Istarthour – this is an integer hour designation
- Istartminute – this is an integer minute designation
- Istartsecond – this is an integer second designation
- Ierror -- If there is no error this integer will return a 0, a number greater than 0 means there was an error.

8.3.7 call hlsetnumlayers(Ihl_handle,num_layer,ierror)

This function is used to determine the number of layers that will be passed in the hydrodynamic linkage file. If the hydrodynamic model is a sigma stretch grid (constant number of layers), when WASP is initially linked to the hydrodynamic file it will automatically determine segment orientation for the light path and determine which segments have an air interface. If the hydrodynamic model does not have a constant number of layers an auxiliary file will need to be created (SEE SECTION).

Where:

- Ihl_handle -- the file handle assigned by Hlopen (integer)
- Num_layer – this is an integer that specifies the number of layers contained in the hydrodynamic model.
- Ierror -- If there is no error this integer will return a 0, a number greater than 0 means there was an error.

8.3.8 call hlsetnumsegments(Ihl_handle, noseg, ierror)

This function defines the number of cells/segments that will be transferred from the hydrodynamic model. This number is constant throughout development of the hydrodynamic linkage. This number defines the number of segment that segment constituents will be saved. WASP uses this number to set the number of segments when initially linked to the hydrodynamic linkage file.

Where:

- Ihl_handle -- the file handle assigned by Hlopen (integer)
- Noseg – this is an integer that specifies the number of segments that information will be saved.
- ierror -- If there is no error this integer will return a 0, a number greater than 0 means there was an error.

8.3.9 call hlsetsegname(ihl_handle,i,segname,ierror)

This function allows a segment name be assigned to each cell. This segment name will be imported into WASP during the initial linkage process. Typically segment names could be the cell designation in the hydrodynamic model, such as I=1, J=1, K=4.

Where:

- Ihl_handle -- the file handle assigned by Hlopen (integer)
- Iseg – is the segment number from 1 to noseg that name is being defined.
- Segname – this is a string no larger than 30 characters that can specify a name associated with each cell.
- ierror -- If there is no error this integer will return a 0, a number greater than 0 means there was an error.

8.3.10 call hlsetnumflowpaths(Ihl_handle, numflow, ierror)

This function sets the number of flow paths that will be defined in the hydrodynamic linkage file. See figure xx for what defines a flow path. Once the time loop starts, this is the number of flows that will need to be written.

Where:

- Ihl_handle – the file handle assigned by Hlopen (integer)
- Numflow – the number of flow paths that will be passed to WASP
- ierror -- If there is no error this integer will return a 0, a number greater than 0 means there was an error.

8.3.11 call hlsetnumsegconsts(lhl_handle, inumsegconsts, ierror)

This function is used to set the number of segment constituents that will be written to the hydrodynamic linkage file. The current version of the HYDROLINK API assumes a particular order. To get to a particular constituent you must define the earlier ones. Segment constituents are: volume, depth, velocity, temperature and salinity.

Where:

- lhl_handle – the file handle assigned by Hlopen (integer)
- Inumsegconsts – defines the number of segment constituents that will be written to hydrodynamic linkage file (integer)
- ierror -- If there is no error this integer will return a 0, a number greater than 0 means there was an error.

8.3.12 Hlsetnumfpconsts (lhl_handle, NumFlowPathConst, ierror)

This function is used to specify the number of flow path constituents. The number of flow path constituents that are passed by the hydrodynamic model is typically a function of the dimensionality of the model. For models like EFDC the number of flow path constituents is three: 1) Flow 2) Dispersion/residual flow, 3) Direction of Flow. For simple 1 dimensional models like DYNHYD the number of flow path constituents is one, Flow.

Where:

- Ihandle – the handle assigned by Hlopen (integer)
- NumFlowPathConst – Number of Flow path constituent
- ierror -- If there is no error this integer will return a 0, a number greater than 0 means there was an error.

8.3.13 Hlsetfpconsttype (lhl_handle, IconType, Index, ierror)

This function is from 1 to the Number of Flow Path Constituents, setting the characteristic of individual flow path constituent.

Where:

For Index = 1, NumFlowPathConst

- Ihandle – the handle assigned by Hlopen (integer)
- IconType – Integer value to define flow constituent
- 1 – Flow
- 2 – dispersion
- 3 – Flow Direction
- Index – 1 to NumFlowPathConst
- ierror -- If there is no error this integer will return a 0, a number greater than 0 means there was an error.

End Do

8.3.14 call hlsetvartimestep(lhl_handle,IdtOpt,ierror)

This function is used to specify the timestep option. If the hydrodynamic model has the ability to take dynamic timestep, additional information will need to be written to the hydrodynamic linkage file so that WASP can take the same timestep.

Where:

- Ihl_handle – the handle assigned by Hlopen (integer)
- IdtOpt – sets the timestep option (integer)
- 0 – Constant Timestep, the hydrodynamic model timestep will be constant
- 1 – Time variable timestep, note additional information needs to be written to the hydrodynamic linkage
- ierror -- If there is no error this integer will return a 0, a number greater than 0 means there was an error.

8.3.15 call hlsettimestep(lhl_handle,hdt,ierror)

This function is used to set the timestep that WASP will use. This value is typically set to the timestep that is used by the hydrodynamic model. If the time variable timestep option is not used, this value only has to be specified once. If the time variable timestep is selected, it needs to be specified each time a data frame of data is written.

Where:

- Ihl_handle -- the handle assigned by Hlopen (integer)
- Hdt --- timestep to set in seconds (integer)
- ierror -- If there is no error this integer will return a 0, a number greater than 0 means there was an error.

8.3.16 call hlsethydtimestep(lhl_handle,hdt,ierror)

This function specifies the timestep that is used for the hydrodynamic model simulation. This value only needs to be set once.

Where:

- Ihl_handle -- the handle assigned by Hlopen (integer)
- Hdt --- timestep to set in seconds (integer)
- ierror -- If there is no error this integer will return a 0, a number greater than 0 means there was an error.

8.3.17 call hlsetupdateint(lhl_handle,rinterval,ierror)

This function specifies the time interval in which data is written to the hydrodynamic linkage file. WASP does not require information at every hydrodynamic model timestep. The time interval in which information is written is:
timestep * numdht
It is t this interval that WASP will read a new set of data from the hydrodynamic linkage file.

Where:

- lhl_handle -- the handle assigned by Hlopen (integer)
- rinterval—this specifies the time interval between reading new information from the hydrodynamic linkage file in seconds (integer)
- ierror -- If there is no error this integer will return a 0, a number greater than 0 means there was an error.

8.3.18 call hlsethydtowaspratio(lhl_handle,numdht,ierror)

This function specifies the number of hydrodynamic model timesteps that will be taken before information is saved to the hydrodynamic linkage file. This value is used to calculate the simulation time interval between writes/reads to/from of the hydrodynamic linkage file.

Where:

- lhl_handle -- the handle assigned by Hlopen (integer)
- numdht --- specifies the number of timesteps the hydrodynamic model takes between writing information to the hydrodynamic linkage file (integer)
- ierror -- If there is no error this integer will return a 0, a number greater than 0 means there was an error.

8.3.19 call hlsetflowpath(lhl_handle,k,jq(k),iq(k),iflowdir(k),ierror)

This function sets the physical flow paths so that information can be mapped into WASP. This is a critical step in setting up a linkage file for WASP. The order of the flow path information that is entered here dictates the order that the actual flow information has to written later.
Where:
DO K=1, NumFlow

- lhl_handle – is the handle thst is assigned by Hlopen (integer)
- K – is the index number of the flow path being defined (integer)
- JQ – is the upstream cell (has to be a number 1 to number os segments)
- IQ – is thedownstream cell (has to be a number 1 to number of segments)
- Iflowdir(k) –is the direction of the flow path
- Ierror -- If there is no error this integer will return a 0, a number greater than 0 means there was an error.

End DO

8.4 *Segment Information*

Segment information is sent to the API for 3 segment components that are needed by WASP. All of the data that will be sent to WASP must be loaded in the array prior to the API call. The same call is used for all segment information. The variable ISegInfo determines which data is currently being sent.

8.4.1 call hlsetseginfo(lhl_handle,IsegInfo,SegVolume,ierror)

where:

- lhl_handle – is the handle that is assigned by Hlopen (integer)
- ISegInfo = 1 Segment Volume (m^3)
- ISegInfo = 2 Segment Depth (m)
- IsegInfo = 3 Segment Velocity (m/sec)
- SegVolume = is a real array dimension 1 to number of segments, containing the corresponding volume.
- Ierror -- If there is no error this integer will return a 0, a number greater than 0 means there was an error.

8.5 *Flow Information*

The user has the ability to send two types of flows to the API that could be used by WASP. The first type is advective flow and dispersive flow. All of the data that will be sent to WASP must be loaded in the array prior to the API call. The same call is used for all flow information. The second variable in the call controls whether it is advective flow (1) or dispersive flow (2).

8.5.1 call hlsetflowinfo(lhl_handle,1,Flow,ierror)

8.5.2 call hlsetflowinfo(lhl_handle,2,brintt,ierror)

Where:

- lhl_handle – is the handle that is assigned by Hlopen (integer)
- 1 or 2 – designates advective or dispersive flow
- Flow/Brintt = is a real array dimensioned 1 to number of flows, corresponding to flow paths defined above.
- Ierror -- If there is no error this integer will return a 0, a number greater than 0 means there was an error.

9 End Moment

Once a complete data frame (segment and flow information) has been saved for the current timestep, the API needs to be informed to advance to the next data frame. The call below gives this instruction. When called the data frame is flushed out of memory and compressed in the hydrodynamic linkage file.

call hlmomentcomplete(Ihl_Handle,ierror)

Where:

- Ihl_handle – is the handle that is assigned by Hlopen (integer)
- Ierror -- If there is no error this integer will return a 0, a number greater than 0 means there was an error.

10 Close File

Before exiting the program that is using this API, an instruction has to be given for the API to close the file. Failing to make this call will make the hydrodynamic linkage file unusable.
call hlclose(Ihl_handle,ierror)

Where:

- Ihl_handle – is the handle that is assigned by Hlopen (integer)
- Ierror -- If there is no error this integer will return a 0, a number greater than 0 means there was an error.

11 Compilation Guidance

When compile code to link with the HYDROLINK API the user will have to be aware that you are linking to external functions. Depending upon the compiler/linker you may be required to take additional steps. The HYDROLINK API has been successfully called from the following development environments:

- Absoft Fortran
- Intel Fortran *
- GNU Fortran
- Microsoft Visual C++
- Microsoft Visual Basic

The Intel compiler as others may require interface files for functions contained in the API.
Interface Files

12 Interface Files

If your compiler requires interface files for the external calls, they are listed below.

```
interface
  subroutine hlsetdebug(hl_debug)
    !ms$attributes c,dllimport,alias:'__hlsetdebug'::hlsetdebug
    Integer hl_debug
    !ms$attributes reference :: hl_debug
  end subroutine
end interface
```

```
!-----------------------------------------------------------------------------------------------------------------------
----
    interface
      subroutine hlgetlasterror(message)
        !ms$attributes c,dllimport,alias:'__hlgetlasterror'::hlgetlasterror
        character*(*) message
        !ms$attributes reference :: message
      end subroutine
    end interface
!-----------------------------------------------------------------------------------------------------------------------
----
    interface
      subroutine hlopen(FName, hl_mode, hl_handle, ierror)
        !ms$attributes c,dllimport,alias:'__hlopen'::hlopen
        character*(*) Fname
        Integer hl_mode, hl_handle, ierror
        !ms$attributes reference :: FName, hl_mode, hl_handle, ierror
      end subroutine
    end interface
!-----------------------------------------------------------------------------------------------------------------------
----
    interface
      subroutine hlsetlanguage(hl_handle, hl_language, ierror)
        !ms$attributes c,dllimport,alias:'__hlsetlanguage'::hlsetlanguage
        Integer hl_handle, hl_language, ierror
        !ms$attributes reference :: hl_handle, hl_language, ierror
      end subroutine
    end interface
!-----------------------------------------------------------------------------------------------------------------------
----
    interface
      subroutine hlsetcreator(hl_handle, hl_creator, ierror)
        !ms$attributes c,dllimport,alias:'__hlsetcreator'::hlsetcreator
        Integer hl_handle, hl_creator, ierror
        !ms$attributes reference :: hl_handle, hl_creator, ierror
      end subroutine
    end interface
!-----------------------------------------------------------------------------------------------------------------------
----
    interface
      subroutine hladddescription(hl_handle, id, string, ierror)
        !ms$attributes c,dllimport,alias:'__hladddescription'::hladddescription
```

```
        Integer hl_handle, id, ierror
        character *(*) string
        !ms$attributes reference :: hl_handle, id, string, ierror
      end subroutine
    end interface
!-----------------------------------------------------------------------------------------------------------------------
----
    interface
      subroutine hlsetseedmoment(hl_handle, month, day, year, hour, minute,second,ierror)
        !ms$attributes c,dllimport,alias:'__hlsetseedmoment'::hlsetseedmoment
        Integer hl_handle, month, day, year, hour, minute,second,ierror
        !ms$attributes reference :: hl_handle, month, day, year, hour
        !ms$attributes reference :: minute,second,ierror
      end subroutine
    end interface
!-----------------------------------------------------------------------------------------------------------------------
----
    interface
      subroutine hlsetnumsegments(hl_handle, numsegs, ierror)
        !ms$attributes c,dllimport,alias:'__hlsetnumsegments'::hlsetnumsegments
        Integer hl_handle, numsegs, ierror
        !ms$attributes reference :: hl_handle, numsegs,ierror
      end subroutine
    end interface
!-----------------------------------------------------------------------------------------------------------------------
----
    interface
      subroutine hlsetsegname(hl_handle,index, segname, ierror)
        !ms$attributes c,dllimport,alias:'__hlsetsegname'::hlsetsegname
        Integer hl_handle, index, ierror
        Character *(*) segname
        !ms$attributes reference :: hl_handle, index, segname,ierror
      end subroutine
    end interface
!-----------------------------------------------------------------------------------------------------------------------
----
    interface
      subroutine hlsetnumflowpaths(hl_handle,numfp, ierror)
        !ms$attributes c,dllimport,alias:'__hlsetnumflowpaths'::hlsetnumflowpaths
        Integer hl_handle, numfp, ierror
        !ms$attributes reference :: hl_handle, numfp, ierror
      end subroutine
```

```
    end interface
!-----------------------------------------------------------------------------------------------------------------------
----
    interface
      subroutine hlsetnumsegconsts(hl_handle,numsc, ierror)
        !ms$attributes c,dllimport,alias:'__hlsetnumsegconsts'::hlsetnumsegconsts
        Integer hl_handle, numsc, ierror
        !ms$attributes reference :: hl_handle, numsc, ierror
      end subroutine
    end interface
!-----------------------------------------------------------------------------------------------------------------------
----
    interface
      subroutine hlsetnumfpconsts(hl_handle,numfpc, ierror)
        !ms$attributes c,dllimport,alias:'__hlsetnumfpconsts'::hlsetnumfpconsts
        Integer hl_handle, numfpc, ierror
        !ms$attributes reference :: hl_handle, numfpc, ierror
      end subroutine
    end interface
!-----------------------------------------------------------------------------------------------------------------------
----
    interface
      subroutine hlsetsegconsttype(hl_handle,sc_index, sc_type,ierror)
        !ms$attributes c,dllimport,alias:'__hlsetsegconsttype'::hlsetsegconsttype
        Integer hl_handle, sc_index, sc_type, ierror
        !ms$attributes reference :: hl_handle, sc_index, sc_type, ierror
      end subroutine
    end interface
!-----------------------------------------------------------------------------------------------------------------------
----
    interface
      subroutine hlsetfpconsttype(hl_handle,fp_index, fp_type,ierror)
        !ms$attributes c,dllimport,alias:'__hlsetfpconsttype'::hlsetfpconsttype
        Integer hl_handle, fp_index, fp_type, ierror
        !ms$attributes reference :: hl_handle, fp_index, fp_type, ierror
      end subroutine
    end interface
!-----------------------------------------------------------------------------------------------------------------------
----
    interface
      subroutine hlsetvartimestep(hl_handle,vardt,ierror)
        !ms$attributes c,dllimport,alias:'__hlsetvartimestep'::hlsetvartimestep
```

```
        Integer hl_handle, vardt, ierror
        !ms$attributes reference :: hl_handle,vardt, ierror
      end subroutine
    end interface
!-------------------------------------------------------------------------------------------------------------------------
----
    interface
      subroutine hlsethydtimestep(hl_handle,timestep,ierror)
        !ms$attributes c,dllimport,alias:'__hlsethydtimestep'::hlsethydtimestep
        Integer hl_handle, ierror
        Real timestep
       !ms$attributes reference :: hl_handle,timestep, ierror
      end subroutine
    end interface
!-------------------------------------------------------------------------------------------------------------------------
----
    interface
      subroutine hlsetupdateint(hl_handle,updateinterval,ierror)
        !ms$attributes c,dllimport,alias:'__hlsetupdateint'::hlsetupdateint
        Integer hl_handle, ierror
        Real updateinterval
       !ms$attributes reference :: hl_handle,updateinterval, ierror
      end subroutine
    end interface
!-------------------------------------------------------------------------------------------------------------------------
----
    interface
      subroutine hlsethydtowaspratio(hl_handle,iratio,ierror)
        !ms$attributes c,dllimport,alias:'__hlsethydtowaspratio'::hlsethydtowaspratio
        Integer hl_handle, iratio, ierror
       !ms$attributes reference :: hl_handle,iratio, ierror
      end subroutine
    end interface
!-------------------------------------------------------------------------------------------------------------------------
----
    interface
      subroutine hlsetnumlayers(hl_handle,numlayers,ierror)
        !ms$attributes c,dllimport,alias:'__hlsetnumlayers'::hlsetnumlayers
        Integer hl_handle, numlayers, ierror
        !ms$attributes reference :: hl_handle,numlayers, ierror
      end subroutine
    end interface
```

```
!-----------------------------------------------------------------------------------------------------------------------
----
    interface
      subroutine hlsetflowpath(hl_handle,flow_index,from_seg, to_seg, direction, ierror)
        !ms$attributes c,dllimport,alias:'__hlsetflowpath'::hlsetflowpath
        Integer hl_handle, flow_index,from_seg, to_seg,direction,ierror
        !ms$attributes reference :: hl_handle,flow_index,from_seg
        !ms$attributes reference :: to_seg,direction,ierror
      end subroutine
    end interface
!-----------------------------------------------------------------------------------------------------------------------
----
    interface
      subroutine hlsetflowinfo(hl_handle,index,value,ierror)
        !ms$attributes c,dllimport,alias:'__hlsetflowinfo'::hlsetflowinfo
        Integer hl_handle, index, ierror
        Real value
        !ms$attributes reference :: hl_handle,index, value, ierror
      end subroutine
    end interface
!-----------------------------------------------------------------------------------------------------------------------
----
    interface
      subroutine hlsetseginfo(hl_handle,index,value,ierror)
        !ms$attributes c,dllimport,alias:'__hlsetseginfo'::hlsetseginfo
        Integer hl_handle, index, ierror
        Real value
       !ms$attributes reference :: hl_handle,index, value, ierror
      end subroutine
    end interface
!-----------------------------------------------------------------------------------------------------------------------
----
    interface
      subroutine hlsettimestep(hl_handle,value,ierror)
        !ms$attributes c,dllimport,alias:'__hlsettimestep'::hlsettimestep
        Integer hl_handle, ierror
        Real value
        !ms$attributes reference :: hl_handle,value, ierror
      end subroutine
    end interface
!-----------------------------------------------------------------------------------------------------------------------
----
```

```
    interface
      subroutine hlmomentcomplete(hl_handle,ierror)
        !ms$attributes c,dllimport,alias:'__hlmomentcomplete'::hlmomentcomplete
        Integer hl_handle, ierror
        !ms$attributes reference :: hl_handle,ierror
      end subroutine
    end interface
!-----------------------------------------------------------------------------------------------------------------------
----
    interface
      subroutine hlclose(hl_handle,ierror)
        !ms$attributes c,dllimport,alias:'__hlclose'::hlclose
        Integer hl_handle, ierror
        Real value
        !ms$attributes reference :: hl_handle,ierror
      end subroutine
    end interface
!-----------------------------------------------------------------------------------------------------------------------
----
```

13 Example Program

The following is source code to a utility program that is distributed with WASP. It converts ASCII and binary outputs from several hydrodynamic models into the format required by WASP. This would be a good starting point for your development.

```
     Program HydroLink
!-----------------------------------------------------------------------------------------------------------------------
----
    Integer, Allocatable, Dimension (:):: IQ, JQ, IFLOWDIR
    Real, Allocatable, Dimension (:):: SegVolume, SegDepth, SegVel, Flow, crnu, brintt
    integer*4 Ihl_handle
    character*1 ANS
    character*30 segname
    character*256 HLFILE,INFILE,segfile
    character*256 DESCRIPTION(10)
    character*256 MODELERNAME
    character*256 errstring
    logical binary, ConfigFile,EFDC,DYNHYD, EPDRIV1, HECRAS
```

```
!-----------------------------------------------------------------------------------------------------------------------
----
!     Opening Message
!-----------------------------------------------------------------------------------------------------------------------
----
!     write(6,6120)
!6120  format(3(/),62('-'),/,
!    'The purpose of this program is to convert a previously created',/ &
!    'hydrodynamic linkage file from EFDC, DYNHYD, EPDRIV1 to the   ',/ &
!    'new HYDROLINK method.  This is required for the latest version',/ &
!    'of WASP.  The user must specify the path and filename of all  ',/ &
!    'files to be created, must specify the file type and from which',/ &
!    'hydrodynamic model created the old file.  The user must also  ',/ &
!    'specify the start time (Gregorian Format) of the hydrodynamic ',/ &
!    'linkage file.                                  ',/ &
!     62('-'),2(/))
!-----------------------------------------------------------------------------------------------------------------------
----
!     hlopen parameters
!-----------------------------------------------------------------------------------------------------------------------
----
     Ihl_language    = 1
     Ihl_creator     = 1
     Ihl_handle      = 0
     Ihl_debug       = 1
     Ihl_mode        = 1
     inumsegconsts   = 3
     inumfpconsts    = 3
     binary          = .true.
     ConfigFile      = .false.
     EFDC            = .false.
     DYNHYD          = .false.
     EPDRIV1         = .false.
     HECRAS          = .false.
!-----------------------------------------------------------------------------------------------------------------------
----
!    Open the Control File
!-----------------------------------------------------------------------------------------------------------------------
----
     open (unit=10,file='hydrolink.ctl',status='old',iostat=istat)
     if (istat .eq. 0) then
       write(6,*)'Previous Control File Found'
```

```
      write(6,*)'Do you want to read from File (Y=Yes, N=No)'
      read(5,*)ANS
      If(ANS .eq. 'Y' .or. ANS .eq. 'y')ConfigFile=.true.
      If(ANS .eq. 'A' .or. ANS .eq. 'a')ConfigFile=.false.
      if (ConfigFile) then
        write(6,*)'Reading Information from HYDROLINK.CTL'
      else
        close(unit=10)
      endif
    endif
!-------------------------------------------------------------------------------------------------------------------------
----
    if (ConfigFile)then
      read(10,*)ANS
      If(ANS .eq. 'A' .or. ANS .eq. 'a')binary=.false.
      If(ANS .eq. 'B' .or. ANS .eq. 'b')binary=.true.
!-------------------------------------------------------------------------------------------------------------------------
----
      read(10,*)ANS
      If(ANS .eq. 'E' .or. ANS .eq. 'e')EFDC=.true.
      If(ANS .eq. 'D' .or. ANS .eq. 'd')DYNHYD=.true.
      If(ANS .eq. 'R' .or. ANS .eq. 'r')EPDRIV1=.true.
      If(ANS .eq. 'H' .or. ANS .eq. 'h')HECRAS=.true.
!-------------------------------------------------------------------------------------------------------------------------
----
      read(10,1000)INFILE
      if (binary) then
        open(unit=1,file=INFILE,form='unformatted',status='old',  &
           iostat=istat)
        if (istat .gt. 0) then
          write(6,*)'HYD File not Found: '
          stop
        endif
      else
        open(unit=1,file=INFILE,status='old',iostat=istat)
        if (istat .gt. 0) then
          write(6,*)'HYD File not Found: '
          stop
        endif
      endif
!-------------------------------------------------------------------------------------------------------------------------
----
```

```
      read(10,1000)HLFILE
      HLFILE=trim((HLFILE)//CHAR(0))
      read(10,*)istartmonth
      read(10,*)istartday
      read(10,*)istartyear
      read(10,*)istarthour
      read(10,*)istartminute
      read(10,*)istartsecond
      read(10,1000)segfile
      READ(10,*)NUM_DESCRIPTIONS
      do i=1,NUM_DESCRIPTIONS
        READ(10,1111)DESCRIPTION(I)
1111      format(A256)
      end do
1000    format(A64)
    else
!-------------------------------------------------------------------------------------------------------------------------------
----
      open (unit=10,file='hydrolink.ctl',status='unknown')
      write(6,*)'Is Old HYD ASCII or Binary (A=ASCII, B=Binary)'
      read(5,*)ANS
      write(10,1060)ANS
1060    format(A1)
      If(ANS .eq. 'A' .or. ANS .eq. 'a')binary=.false.
      If(ANS .eq. 'B' .or. ANS .eq. 'b')binary=.true.
!-------------------------------------------------------------------------------------------------------------------------------
----
      write(6,*)'Enter File Type (E=EFDC, D=DYNHYD, R=EPDRIV1, H=HECRAS)'
      read(5,*)ANS
      write(10,1060)ANS
      If(ANS .eq. 'E' .or. ANS .eq. 'e')EFDC=.true.
      If(ANS .eq. 'D' .or. ANS .eq. 'd')DYNHYD=.true.
      If(ANS .eq. 'R' .or. ANS .eq. 'r')EPDRIV1=.true.
      If(ANS .eq. 'H' .or. ANS .eq. 'h')HECRAS=.true.
!-------------------------------------------------------------------------------------------------------------------------------
----
      write(6,*)'Enter Name of Previous Version HYD File to Convert'
      read(5,*)INFILE
      write(10,1000)INFILE
      if (binary) then
        open(unit=1,file=INFILE,form='unformatted',status='old', iostat=istat)
        if (istat .gt. 0) then
```

```
          write(6,*)'HYD File not Found: '
          stop
        endif
      else
        open(unit=1,file=INFILE,status='old',iostat=istat)
        if (istat .gt. 0) then
          write(6,*)'HYD File not Found: '
          stop
        endif
      endif
!-------------------------------------------------------------------------------------------------------------------------
----
      write(6,*)'Enter Name of HYDROLINK HYD File to Create'
      read(5,*)HLFILE
      write(10,1000)HLFILE
!-------------------------------------------------------------------------------------------------------------------------
----
      Write(6,6600)
6600    format('Seed Time Information Needed for the Start of  ',/'  Hydrodynamic Linkage File
mm/dd/yyyy hh:mm:ss')
      write(6,*)'Enter Start Month (mm)'
      read(5,*)istartmonth
      write(6,*)'Enter Start Day (dd)'
      read(5,*)istartday
      write(6,*)'Enter Start Year (yyyy)'
      read(5,*)istartyear
      write(6,*)'Enter Start Hour (hh)'
      read(5,*)istarthour
      write(6,*)'Enter Start Minute (mm)'
      read(5,*)istartminute
      write(6,*)'Enter Start Second (ss)'
      read(5,*)istartsecond
      write(10,*)istartmonth
      write(10,*)istartday
      write(10,*)istartyear
      write(10,*)istarthour
      write(10,*)istartminute
      write(10,*)istartsecond
      write(6,*)'Enter Name of Segment Name File (Type NONE)'
      read(5,1000)segfile
      write(10,1000)segfile
      write(6,6610)
```

```
6610    format('How Many Description Lines would you like to add to the file (0=None)')
        read(5,*)NUM_DESCRIPTIONS
        write(10,*) NUM_DESCRIPTIONS
        do i=1, NUM_Descriptions
          write(6,*)'Enter Description:',I
          read(5,1111)description(i)
          write(10,1111)description(i)
        end do
        close(unit=10)
      End If
      if(EFDC)MODTYPE    =1
      if(DYNHYD)MODTYPE  =2
      if(EPDRIV1)MODTYPE =3
      if(HECRAS)MODTYPE  =2
!-----------------------------------------------------------------------------------------------------------------------
----
      HLFILE=(TRIM(HLFILE)//CHAR(0))
      write(6,*)'About to Create Hydrolink File'
      call hlopen(HLFILE, Ihl_mode, Ihl_handle, ierror)
      if(ierror .gt. 0)then
         call hlgetlasterror(errstring)
         write(6,6000) ierror, errstring
         stop
      end if
      write(6,*)'Hydrolink File Created'
!-----------------------------------------------------------------------------------------------------------------------
----
!     Set the language to FORTRAN
!-----------------------------------------------------------------------------------------------------------------------
----
      call hlsetlanguage(Ihl_handle, 1, ierror)
      if(ierror .gt. 0)then
         call hlgetlasterror(errstring)
         write(6,6000) ierror, errstring
         stop
      end if
!-----------------------------------------------------------------------------------------------------------------------
----
!     Store a description string
!-----------------------------------------------------------------------------------------------------------------------
----
      write(6,*)'Storing Descriptions'
```

```
    do i =1,NUM_DESCRIPTIONS
      call hladddescription(Ihl_handle,0,DESCRIPTION(I),ierror)
      if(ierror .gt. 0)then
        call hlgetlasterror(errstring)
        write(6,6000) ierror, errstring
        stop
      end if
    end do
!-----------------------------------------------------------------------------------------------------------------------
----
!    Store the modeler name
!-----------------------------------------------------------------------------------------------------------------------
----
!     call hladddescription(Ihl_handle,1,MODELERNAME,ierror)
!     if(ierror .gt. 0)then
!       call hlgetlasterror(errstring)
!       write(6,6000) ierror, errstring
!       stop
!     end if
!-----------------------------------------------------------------------------------------------------------------------
----
!    Set the creator
!-----------------------------------------------------------------------------------------------------------------------
----
    write(6,*)'Setting Creator'
    call hlsetcreator(Ihl_handle, MODTYPE, ierror)
    if(ierror .gt. 0)then
      call hlgetlasterror(errstring)
      write(6,6000) ierror, errstring
      stop
    end if
!-----------------------------------------------------------------------------------------------------------------------
----
!    Set the seed moment
!-----------------------------------------------------------------------------------------------------------------------
----
    write(6,*)'Setting Seed Moment'
    IF (.not. HECRAS) THEN
    call hlsetseedmoment(Ihl_handle, istartmonth, istartday,istartyear, istarthour, istartminute,
istartsecond, ierror)
    if(ierror .gt. 0)then
      call hlgetlasterror(errstring)
```

```
        write(6,6000) ierror, errstring
        stop
      end if
      ENDIF
!-----------------------------------------------------------------------------------------------------------------------
----
! Read the Header from Hydrodynamic Linkage File
!-----------------------------------------------------------------------------------------------------------------------
----
      write(6,*)'Starting to Process Hydrodynamic File'
      IF(BINARY)THEN
        IF (EFDC)READ(1)NOSEG,NUMFLOW, NUMDHT,HDT, START,END,NUM_LAYER
        IF (DYNHYD)READ(1)NOSEG,NUMFLOW, HDT, START,END,NUM_LAYER
      ELSE
        IF(EFDC)READ(1,*)NOSEG,NUMFLOW, NUMDHT, HDT, START,END, NUM_LAYER
        IF(DYNHYD)READ(1,*)NOSEG,NUMFLOW, HDT, START,END,NUM_LAYER
        IF(HECRAS)Then
          READ(1,*)NOSEG,NUMFLOW, HDT, IMON, IDAY, IYEAR, IHour, Imin,NUM_LAYER
          call hlsetseedmoment(Ihl_handle, IMON, IDAY, IYEAR, IHour,Imin, 0, ierror)
          NUMDHT = 1
      Endif
      END IF
!-----------------------------------------------------------------------------------------------------------------------
----
      Allocate (IQ(NUMFLOW))
      Allocate (JQ(NUMFLOW))
      Allocate (IFLOWDIR(NUMFLOW))
      Allocate (SEGVOLUME(NOSEG))
      Allocate (SEGDEPTH(NOSEG))
      Allocate (SEGVEL(NOSEG))
      Allocate (Flow(NUMFLOW))
      Allocate (CRNU(NUMFLOW))
      Allocate (BRINTT(NUMFLOW))
!-----------------------------------------------------------------------------------------------------------------------
----
      if (num_layer .lt. 1) then
        num_layer=1
        numdht=1
      endif
      if (numdht .lt. 1)numdht=1
!-----------------------------------------------------------------------------------------------------------------------
----
```

```
    call hlsetnumlayers(Ihl_handle,num_layer,ierror)
    if(ierror .gt. 0)then
       call hlgetlasterror(errstring)
       write(6,6000) ierror, errstring
       stop
    end if
!-------------------------------------------------------------------------------------------------------------------------
----
!    Set the number of segments
!-------------------------------------------------------------------------------------------------------------------------
----
    call hlsetnumsegments(Ihl_handle, noseg, ierror)
    if(ierror .gt. 0)then
       call hlgetlasterror(errstring)
       write(6,6000) ierror, errstring
       stop
    end if
!-------------------------------------------------------------------------------------------------------------------------
----
!            Get Segment Name Map File if exists
!-------------------------------------------------------------------------------------------------------------------------
----
    open(unit=15,file=segfile,status='old', iostat=istat)
    write(6,*)'Opening  segment map file; iostat = ',istat          !3/11/08,rba                    !
    if (istat.eq.0) then
      write(6,*)'Segment Map Text file is: ',segfile              !
      do i=1, NOSEG
        read(15,4000)SEGNAME
        SEGNAME =(TRIM(SEGNAME)//CHAR(0))                    !
        call hlsetsegname(ihl_handle,i,segname,ierror)
        write(6,*)'Segment ',i,' name: ',segname,' err code ',ierror              !
4000      format(A30)
      end do
    Else
      do i=1, NOSEG
        call hlsetsegname(ihl_handle,i,'WASP-Seg',ierror)
      end do

    endif
!-------------------------------------------------------------------------------------------------------------------------
----
!    Set the number of flow paths
```

```
!-----------------------------------------------------------------------------------------------------------------------------
----
    call hlsetnumflowpaths(Ihl_handle, numflow, ierror)
    if(ierror .gt. 0)then
       call hlgetlasterror(errstring)
       write(6,6000) ierror, errstring
       stop
    end if
!-----------------------------------------------------------------------------------------------------------------------------
----
!    Set the number of segment constituents
!-----------------------------------------------------------------------------------------------------------------------------
----
    call hlsetnumsegconsts(Ihl_handle, inumsegconsts, ierror)
    if(ierror .gt. 0)then
       call hlgetlasterror(errstring)
       write(6,6000) ierror, errstring
       stop
    end if
!-----------------------------------------------------------------------------------------------------------------------------
----
!    Set the number of flow path constituents
!-----------------------------------------------------------------------------------------------------------------------------
----
    if(EFDC)call hlsetnumfpconsts (Ihl_handle, 3, ierror)
!     if(DYNHYD)call hlsetnumfpconsts (Ihl_handle, 1, ierror)
    if(DYNHYD)call hlsetnumfpconsts (Ihl_handle, 2, ierror)         !6/10/08,rba
    if(HECRAS)call hlsetnumfpconsts (Ihl_handle, 2, ierror)         !6/10/08,rba
    if(ierror .gt. 0)then
       call hlgetlasterror(errstring)
       write(6,6000) ierror, errstring
       stop
    end if
!-----------------------------------------------------------------------------------------------------------------------------
----
!    Now we will set all the constitituent types
!-----------------------------------------------------------------------------------------------------------------------------
----
    call hlsetsegconsttype(Ihl_handle, 1, 0, ierror)
    if(ierror .gt. 0)then
       call hlgetlasterror(errstring)
       write(6,6000) ierror, errstring
```

```
        stop
      end if
!-------------------------------------------------------------------------------------------------------------------------------
----
      if(EFDC)then
        call hlsetsegconsttype(Ihl_handle, 2, 1, ierror)
        if(ierror .gt. 0)then
          call hlgetlasterror(errstring)
          write(6,6000) ierror, errstring
          stop
        end if
        call hlsetsegconsttype(Ihl_handle, 3, 2, ierror)
        if(ierror .gt. 0)then
         call hlgetlasterror(errstring)
         write(6,6000) ierror, errstring
         stop
        end if
      endif
!-------------------------------------------------------------------------------------------------------------------------------
----
      call hlsetfpconsttype(Ihl_handle, 1, 0, ierror)
      if(ierror .gt. 0)then
         call hlgetlasterror(errstring)
         write(6,6000) ierror, errstring
         stop
      end if
      if (DYNHYD)then                                  !6/10/08,rba
        call hlsetfpconsttype(Ihl_handle, 2, 0, ierror)          !
        if(ierror .gt. 0)then
          call hlgetlasterror(errstring)
          write(6,6000) ierror, errstring
          stop
        endif
      end if
      if (HECRAS)then                                  !6/10/08,rba
        call hlsetfpconsttype(Ihl_handle, 1, 0, ierror)          !
        if(ierror .gt. 0)then
          call hlgetlasterror(errstring)
          write(6,6000) ierror, errstring
          stop
        endif
      end if
```

```
    if (EFDC)then
      call hlsetfpconsttype(Ihl_handle, 1, 1, ierror)          !shouldn't this be 2,1
      if(ierror .gt. 0)then
        call hlgetlasterror(errstring)
        write(6,6000) ierror, errstring
        stop
      endif
      call hlsetfpconsttype(Ihl_handle, 1, 2, ierror)          !shouldn't this be 3,1
      if(ierror .gt. 0)then
        call hlgetlasterror(errstring)
        write(6,6000) ierror, errstring
        stop
      end if
    end if
!-------------------------------------------------------------------------------------------------------------------------------
----
    call hlsetvartimestep(Ihl_handle,0,ierror)
    if(ierror .gt. 0)then
       call hlgetlasterror(errstring)
       write(6,6000) ierror, errstring
       stop
    end if
    timestep=(hdt/86400.)
!-------------------------------------------------------------------------------------------------------------------------------
----
    IF (HECRAS)then
       call hlsettimestep(Ihl_handle,timestep,ierror)
    Else If (DYNHYD) then
       call hlsettimestep(Ihl_handle,timestep,ierror)
    Else
       call hlsettimestep(Ihl_handle,hdt,ierror)
    End If
!-------------------------------------------------------------------------------------------------------------------------------
----
    call hlsethydtimestep(Ihl_handle,hdt,ierror)
    if(ierror .gt. 0)then
       call hlgetlasterror(errstring)
       write(6,6000) ierror, errstring
       stop
    end if
    rinterval=(hdt*numdht)/86400.
    IF (DYNHYD) THEN
```

```
      rinterval=(hdt*numdht)
    ENDIF
!-------------------------------------------------------------------------------------------------------------------------------
----
    call hlsetupdateint(Ihl_handle,rinterval,ierror)
    if(ierror .gt. 0)then
      call hlgetlasterror(errstring)
      write(6,6000) ierror, errstring
      stop
    end if
!-------------------------------------------------------------------------------------------------------------------------------
----
    call hlsethydtowaspratio(Ihl_handle,numdht,ierror)
    if(ierror .gt. 0)then
      call hlgetlasterror(errstring)
      write(6,6000) ierror, errstring
      stop
    end if
    idt=1
!-------------------------------------------------------------------------------------------------------------------------------
----
    do i=1,numflow
      if (binary) then
        read(1)JQ(I), IQ(I)
      else
        IF(EFDC)read(1,1001)JQ(I), IQ(I)
        IF(DYNHYD)read(1,1001)JQ(I), IQ(I)
        IF(HECRAS)read(1,*)JQ(I), IQ(I)
1001      format(2(I5))
      endif
    end do
!-------------------------------------------------------------------------------------------------------------------------------
----
    NOCYCLES= END-START
    NOCYCLES=NOCYCLES/(HDT*numdht)
    IF(HECRAS)nocycles=1000000
    Do itime=1,nocycles
      do k=1,Noseg
        if (itime .eq. 1) then
          if (binary) then
            if(EFDC)Read(1,end=999)SegVolume(k),SegDepth(k), SegVel(k)
            if(DYNHYD)read(1,end=999)SegVolume(k),rjunk, SegDepth(k),SegVel(k)
```

```
          else
            if(EFDC)read(1,*,end=999)SegVolume(k),SegDepth(k),SegVel(k)
            if(DYNHYD)read(1,*,end=999)SegVolume(k),rjunk, SegDepth(k),SegVel(k)
            if(HECRAS)read(1,*,end=999)SegVolume(k),SegDepth(k), SegVel(k)
          endif
1020        Format(5x,F15.0,5x,F15.0,5x,F15.0)
        else
          if (binary) then
            if(EFDC)Read(1,end=999)SegVolume(k),SegDepth(k), SegVel(k)
            if(DYNHYD)read(1,end=999)SegVolume(k),rjunk,SegDepth(k),SegVel(k)
          else
            if(EFDC)read(1,*,end=999)SegVolume(k),SegDepth(k), SegVel(k)
            if(DYNHYD)read(1,*,end=999)SegVolume(k),rjunk, SegDepth(k),SegVel(k)
            if(HECRAS)read(1,*,end=999)SegVolume(k),SegDepth(k),SegVel(k)
            Write(6,*)'Segment = ',k
          endif
1011        format(4(F17.0))
        endif
      end do
!-------------------------------------------------------------------------------------------------------------------------
----
      do k=1,numflow
        if (binary) then
          if(EFDC)read(1)Flow(k),crnu(k),iflowdir(k)
          IF(DYNHYD)READ(1)Flow(k),brintt(k)                    !6/10/08,rba
        else
          if (EFDC)read(1,*)Flow(k),crnu(k),brintt(k),iflowdir(k)
          IF (DYNHYD)read(1,1012)Flow(k),brintt(k)              !6/10/08,rba
          IF (HECRAS)read(1,*)Flow(k)
        endif
1012      format(2F20.0)                                  !6/10/08,rba
1010      format(3(F17.0),I5)
      end do
!-------------------------------------------------------------------------------------------------------------------------
----
      if(itime .eq. 1)then
        do k=1,numflow
          if(EFDC)call hlsetflowpath(Ihl_handle,k,jq(k),iq(k),iflowdir(k),ierror)
          if(DYNHYD)call hlsetflowpath(Ihl_handle,k,jq(k),iq(k),1,ierror)
          if(HECRAS)call hlsetflowpath(Ihl_handle,k,jq(k),iq(k),1,ierror)
          if(ierror .gt. 0)then
            call hlgetlasterror(errstring)
```

```
            write(6,6000) ierror, errstring
            stop
          end if
        end do
      end if
!-------------------------------------------------------------------------------------------------------------------------
----
      call hlsetseginfo(Ihl_handle,1,SegVolume,ierror)
      if(ierror .gt. 0)then
        call hlgetlasterror(errstring)
        write(6,6000) ierror, errstring
        stop
      end if
      call hlsetseginfo(Ihl_handle,2,SegDepth,ierror)
      if(ierror .gt. 0)then
        call hlgetlasterror(errstring)
        write(6,6000) ierror, errstring
        stop
      end if
      call hlsetseginfo(Ihl_handle,3,SegVel,ierror)
      if(ierror .gt. 0)then
        call hlgetlasterror(errstring)
        write(6,6000) ierror, errstring
        stop
      end if
!-------------------------------------------------------------------------------------------------------------------------
----
      call hlsetflowinfo(Ihl_handle,1,Flow,ierror)
      if(ierror .gt. 0)then
        call hlgetlasterror(errstring)
        write(6,6000) ierror, errstring
        stop
      end if
      if(DYNHYD)Then                                  !6/10/08,rba
        call hlsetflowinfo(Ihl_handle,2,brintt,ierror)
        if(ierror .gt. 0)then
          call hlgetlasterror(errstring)
          write(6,6000) ierror, errstring
          stop
        end if
      end if
      if(HECRAS)Then
```

```
        DO Itemp=1,numflow
         brintt(Itemp)=0.00
        End Do                                  !6/10/08,rba
        call hlsetflowinfo(Ihl_handle,2,brintt,ierror)
        if(ierror .gt. 0)then
          call hlgetlasterror(errstring)
          write(6,6000) ierror, errstring
          stop
        end if
      end if
      if(EFDC)Then
        call hlsetflowinfo(Ihl_handle,2,crnu,ierror)
        if(ierror .gt. 0)then
          call hlgetlasterror(errstring)
          write(6,6000) ierror, errstring
          stop
        end if
        call hlsetflowinfo(Ihl_handle,3,brintt,ierror)
        if(ierror .gt. 0)then
          call hlgetlasterror(errstring)
          write(6,6000) ierror, errstring
          stop
        end if
      end if
      time=RINTERVAL*itime
      write(6,*)'Time is: ',itime,time
      if(ierror .gt. 0)then
        call hlgetlasterror(errstring)
        write(6,6000) ierror, errstring
        stop
      end if
      call hlmomentcomplete(Ihl_Handle,ierror)
      if(ierror .gt. 0)then
        call hlgetlasterror(errstring)
        write(6,6000) ierror, errstring
        stop
      end if
    end do
999   continue
6000  format('Error ',I10, ' : ', A)
    call hlgetcompfact(Ihl_handle,compact,ierror)
    if(ierror .gt. 0)then
```

```
        call hlgetlasterror(errstring)
        write(6,6000) ierror, errstring
        stop
      end if
      compact=compact*100
      write(6,6040)compact
6040  format('Compaction Ratio is: ',F8.4)
      call hlclose(Ihl_handle,ierror)
      if(ierror .gt. 0)then
        call hlgetlasterror(errstring)
        write(6,6000) ierror, errstring
        stop
      end if
      stop
      end
```