National Weather Service Verification Software Users' Manual

National Weather Service/NOAA
1325 East West Highway, Room 8372
Silver Spring, Maryland 20910
301-713-0640

March *2001*

1.0 Preface

This manual contains information for users of the National Weather Service (NWS) Verification System(VS) . This preface describes general information about using this manual.

1.1 Document Design

This manual is designed to be single-sided and inserted into a 3-ring binder. In order to help users quickly locate information about different VS topics, each major section is indicated by page numbers that uniquely identify the chapter (e.g., Preface-1, as opposed to 1). This organization allows sections to be added or moved as development continues without disruption to page numbering**.**

1.2 Intended Audience

The material in this manual is intended to provide support to end users, not model developers or system administrators. Technical references are made throughout this manual for those interested in software and system design or model theory and implementation.

This software manual is written to be independent of any computer operating system. System-specific issues will be addressed in the system manual. Basic understanding of NWS software applications will help the user run the Verification System more easily, but is not required to use the program.

1.3 Scope of this Manual

This Users' Manual provides general information about running the verification system. components that interface directly with other NWS software are described in this document, not in other NWS modeling and time series documentation. The verification software uses WHFS library routines which are documented separately. All database structures and definitions were provided by the NWS. All code was developed around these structures.

In general, the purpose of this Verification Software Users' Manual is to provide summary information about the Verification Software, information about how to use the software, and examples of operations. Where appropriate, other documentation is referenced.

1.4 Conventions

The following typographical conventions are used in this document:

Regular	is used for normal text within sections (Times font)
Italic	is used for window, menu, file, directory, document, table, and plot names
Bold	indicates an action ("**click on the icon**")
Bold Italic	indicates a figure reference ("see ***Figure*** *7*")
`Constant Width`	is used for program output or information shown on-screen (Courier font)
`Constant Bold`	shows commands that the user enters (Courier bold)
[any font]	designates part of a command that is variable

2.0 General Description

The Verification System Software is designed to extract data from the NWS existing IHFS Database (IHFSDB) and populate a secondary database - the Verification Database (VDB) with information regarding hydrologic forecasts and gauge observations. Additional information at a system level are also included in the verification database. Extraction of information is controlled by the user through control files. Parameters in the VDB and control file specify criteria for extraction of information to the VDB.

Users can perform analysis of forecast observation pairs in the VDB by specifying pairing criteria from the control file. Observation and forecast pairs are grouped together by user specified criteria. Statistics are then calculated, and the results, and pairs exported to an output file if specified.

Several administrative functions are provided with the Verification Software to update location information and do simple administration tasks on VDB tables.

The Verification Software currently runs as an executable using a batch control file to input parameters. The software can readily be run from cron. The code was developed in a modular fashion using C++. Calls to existing NWS WHFS libraries are made to access the IHFSDB. Verification code was designed to readily be adapted for a graphical user interface (GUI) in the future. It is envisioned that the information provided in the control files could be readily accessed through pull-down menus and text screens in a GUI.

3.0 Running Verification

3.1 App_defaults Tokens

The following tokens should be set for the Verification Software:

Name	Example Implementation	Purpose
vsys_dir	$(rfs_dir)/verify	base verification directory
vsys_output	$(vsys_dir)/output	base directory for output files
vsys_input	$(vsys_dir)/input	base directory for input files
vsys_scipts	$(vsys_dir)/scripts	base directory for script files
vsys_files	$(vsys_dir)/files	base directory for files
vsys_ihfsdb	hd1_2tua	IHFS database name
vsys_vdb	vdb1_1tua	Verification database name
vsys_output_log	verify.log	Output log file name - this file is stored in *vsys_output* directory

3.3 Execution

The verification program is run by executing *verify* with the appropriate command line options.

```
%> verify -commands [commandfile] -[command file options]
```

Command Line Options
The following command line arguments are recognized:

`-commands[commandfile]`
Runs the verification using commandfile as input control file. If this option is not used, verification will look for the explicit filename *commandfile* as a default. First $(vsys_input) directory is examined for *commandfile*. If the file is not found in that directory, the current working directory (cwd) is examined. If no file is found, the software will exit. The command

```
%> verify
```

will execute the verification software using the file *commandfile* as the default input file.

`-d#,#`
Sets the debug level to # (screen, file). By default, debug messages are set to level 10. Level 1 messages indicatee run-time control diagnostics resulting from critical failure.. Level 10 include all lower level messages in addition to general run-time control diagnostics. Level 20 messages include all lower level messages in addition to detailed run-time control diagnostics. Higher level messages include all lower level messages in addition to very detailed, low-level library run-time control diagnostics. Debug messages are identified by the string “Debug” on both screen and logfile preceding the debug message.

`-h`
Prints brief help message.

-nolog
Eliminates the generation of a log file.

`-s#,#`
Sets the status message level to # (screen, file). Status messages provide general information regarding location of execution within the program and other general user information. Message level descriptions are the same as those described in the debug option. Status messages are identified by the string “Status” on both screen and logfile preceding the status message.

`-w#,#`
Sets the warning message level to # (screen, file). Warning messages provide information regarding both critical and non-critical execution errors. Message level descriptions are the same as those described in the debug option. Warning messages are identified by the string “Warning” on both screen and logfile preceding the status message.

4.0 Command File Syntax

Key words are used in interpretation of the control file. A keyword is defined by the reserved token immediately followed by an equals sign ("="), followed by the token definition. Token definitions are listed here by typical use, not alphabetically, for ease in understanding program usage. When multiple definitions are possible, each definition must be separated by a comma (",").

TOKEN Syntax is as follows:

```
TOKEN=Value
TOKEN=Value1,Value2
```

Token names defining groupings of actions are required. These tokens are defined by a START_* and END_*. Tokens defined within each grouping are stored for use in completing the appropriate actions. Four such groupings are currently available.

Extraction:
START_OF_EXTRACTION=true
.
.
extraction tokens
.
.
END_OF_EXTRACTION=true

Pairing:
START_OF_PAIRING=true
.
.
Pairing tokens
.
.
END_OF_PAIRING=true

Update_Location:
START_OF_UPDATE_LOCATION=true
.
.
location update tokens
.
.
END_OF_UPDATE_LOCATION=true

Dba:
START_OF_DBA=true

.
.
DB admin tokens
.
.
END_OF_DBA=true

An example command file follows:
```
# lines beginning with # are comment lines
#
START_OF_EXTRACTION=true
IHFSDB=hd1_2tua
VDB=vdb1_1tua
#LOCATION="QUAO2"
#LOCATION="WOOR1,WSLO3,WSRI1,WTTO2,YRB,LNRO2,BGCO2,SPRA4,HSTC2,VASO2,"
LOCATION=ALL
PE_CODES=HG
QUALITY_CODE=2
SENSOR_PREF=FF
OBS_START_TIME="2000-01-01 00:00:00"
OBS_END_TIME="2000-01-15 00:00:00"
TSINTERVAL="1 hr"
END_OF_EXTRACTION=true
#
#
START_OF_BUILDPAIR=true
IHFSDB=hd1_2tua
VDB=vdb1_1tua
LOCATION=ALL
PE_CODES=HG
START_TIME="2000-02-17 00:00:00"
##START_TIME="2000-02-27 00:00:00"
#END_TIME="2000-02-29 18:00:00"
END_TIME="2000-03-01 12:00:00"
PAIRING_WINDOW="3 hr"
RFC=ABRFC
#RIVERRESPONSE=FAST
LEADTIME_START="6 hr"
LEADTIME_END="24 hr"
# this specifies a range for observations between 10% below the floodstage
#    to 100% above the floodstage
MIN_OBS=10
MAX_OBS=100
# this specifies a range for forecasts between 10% below the floodstage
#    to 100% above the floodstage
#MIN_FCST=10
```

```
#MAX_FCST=100
BUILD_PAIRS=(export_pair,export_stat,APPEND)
END_OF_BUILDPAIR=true
#
#
START_OF_UPDATE_LOCATION=true
IHFSDB=hd1_2tua
VDB=vdb1_1tua
#UPDATE_LOCATION="VBUA4"
UPDATE_LOCATION=ALL
END_OF_UPDATE_LOCATION=true
#
#
START_OF_DBA=true
IHFSDB=hd1_2tua
VDB=vdb1_1tua
DATE_RANGE=200707200708
END_OF_DBA=true
```

Specific tokens require values to be enclosed in double quotes. These are documented below.

4.1 Extraction

The extraction operation uses information defined in both the Verification System static tables and the control file token list to query the IHFSDB for forecasts and observations. Because a single location may report data of multiple types, and not all that data is relevant to verification, a subset of that data, sorted by PE, quality and time will be extracted and written to the Verification database. However, all forecasts will be extracted. Forecasts will be extracted from the IHFS-DB table FcstHeight. Observations will be extracted from the IHFS-DB Height table.   Note that the user must either specify location id's(lids) in the commandfile or enter the location id's into the vrivergaugeloc table before any forecast or observations can be extracted from the IHFSDB. The Verification System first checks for lids from the Token list, if the lid list is set to `ALL` then, it checks vrivergaugeloc for all lids. Running the script *insert_lid.script* distributed by HRL populates location id's into the verification database vrivergaugeloc table.

For any two observations that fall within the same user specified time window, the observations will be selected according to the following criteria:, the observation with the best quality code, and then the observation from the preferred sensor (TS) and then the time. This function will work as follows. The time interval will specify the intervals into which each day/hour will be broken. For a 15 minute interval you would have the intervals :00, :15, :30, :45. For any observation, the Verification database will be checked to see if there is already an observation for the desired interval; if there is not the new observation will be written into the Verification database. If there is already an observation, the Quality codes, then the Sensors and the distance from the **start** of the interval will be checked and the preferred observation will be written into the database.

The following tokens can be used in extraction**:**

TOKEN	EXAMPLE	DEFINITION
START_OF_EXTRACTION	true	Extraction group start flag.
IHFSDB	hd1_2tua	IHFS database name. Required if $(vsys_ihfsdb) is not set.
VDB	vdb1_1tua	Verification database name. Required if $(vsys_vdb) not set.
LOCATION	"QUAO2" "QUAO2,WTTO1,RTTA4" ALL	Location identifier must be enclosed in quotes. Multiple locations separated by comma. ALL specifies all lids in vrivergaugeloc table used.
PE_CODES	HG	2 character physical element codes. Multiple codes separated by comma.
OBS_START_TIME	"2000-02-27 00:00:00"	Time at which to start extracting observations. Must be enclosed with quotes. Format is: "YYYY-MM-DD HH:MM:SS"
OBS_END_TIME	"2000-02-28 00:00:00"	Time at which to stop extracting observations. Must be enclosed with quotes. Format is: "YYYY-MM-DD HH:MM:SS"
SENSOR_PREF	"RG"	List of sensor preferences to extract from the IHFSDB. Looks at order of list. Multiple codes separated by comma. Defaults to any if no token specified.
TS_INTERVAL	"30 min"	Time interval used to query, check, and populate new obs value to verification database. Format is "Value Unit" where unit can be "min" or "hr"
END_OF_EXTRACTION	true	Extraction group end flag.

4.2 Pairing

The Pairing/Statistics software will create pairs of observed and forecast data, will write those pairs to an output file with the format specified below, will compute statistics from those pairs, and will write out tables of those statistics.

A pair will be defined by the Location ID, the SHEF PE, and the time. Sorting through the SHEF quality code, and the SHEF Type Source (TS) will have been done in the extraction function.   The Observed Location ID, ,the Observed SHEF PE and the Forecast Location ID, the Forecast SHEF PE must match for a forecast and an observed to be considered for a pair. The system will key off the forecast and then search for an observed value to match with it. It will select an observed value that has an Obstime closest to the forecast Validtime within a user specified window. This window may be different from the window used in the Extraction function.

For those locations with additive adjustments specified, the adjustment will be made after the observation is selected.

Through runtime controls the user will be able to request that the software create pairs with the criteria below. In addition for selected criteria the user can specify that the software loop through a group of the characteristics.

Criteria:

1) **Within a starttime and endtime**. The start and end time are designated by `START_TIME` and `END_TIME` tokens.

2) **Within a user defined window.** The user may specify the window width within which pairs are to be found, or the maximum allowed difference between the forecast validtime and the observed obstime. The token `PAIRING_WINDOW` is used to define the window width.

3) **For a single RFC**. The RFC for which pairs are to be extracted can be specified with the `RFC` token.   Location ids found in the vrivergaugeloc table will be checked to see if they are located in the RFC designated by the `RFC` token. Those lids which fall within that criteria will be used in pairing.

The user may loop through all the RFCs, one at a time, by specifying `ALL`. Output will be written to files with the name of the RFC appended to the end.

4) **For a single river response time**. The river response is defined as “FAST,” “MEDIUM,” and “SLOW.” These designations are updated into the VDB from the IHFSDB with the UPDATE_LOCATION function.   The *IHFSDB response_time* is converted to these classifications s follows: response times < 24 hour are classified as “FAST,” response times < 60 hours and >24 hours are considered “MEDIUM”, and response times > 60 hours are considered “SLOW”.

The user may loop through all of the response times by setting the `RIVERRESPONSE` token to `ALL`. Output will be written to files with the river response category appended to the end.

The response time is the time to peak of the forecst point hydrograph. We have used the response time to sort the forecast points into small, medium, and large river forecast points. 0 to 24 hour response times are small rivers, 24 to 60 hour response times are medium rivers and greater than 60 hours are large rivers.

5) **For a single location**. Pairs for a single location are specified by the `LOCATION` token and the location id.

6) **For multiple locations**. Pairs for multiple locations are specified by the `LOCATION` token and either (1) the location ids, (2) the LOOP_FOR_ALL_LIDS token value, or (3) the ALL token value. The first two will result in statistics being produced independently for each lid, the last will result in statistics being produced for all the LIDs lumped together.

7) **Forecasts above or below some % of the flood stage**. Two tokens are required to specify the lower bound and upper bound of a stage range within which forecast observation pairs will be extracted. Because the ranges are independent and may be either overlapping, or consecutive, the endpoints are included in the range. The system will key off either forecasts or observations depending on the user input.

To define a range of forecasts within which the system should select pairs, the tokens `MIN_FCST` and `MAX_FCST` are used. All forecasts whose values are within the range determined by a percentage of the flood stage will be paired. `MIN_FCST` and `MAX_FCST` are used to bound the range. Setting the first token to 10 and the second token to 100 would return all forecasts/observation pairs that have forecasts between 10% and 100% of the flood stage including the endpoints. Flood stage is determined from the *vlocation* table.

8) **Observations above or below some % of the flood stage**. Two tokens are required to specify the lower bound and upper bound of flood stage range within which forecast/observation pairs will be extracted. Because the ranges are independent and may be either overlapping, or consecutive, the endpoints are included in the range.. The system will key off either forecasts or observations depending on the user input.

To define a range of observations within which the system should select pairs, the tokens `MIN_OBS` and `MAX_OBS` are used. All forecasts whose value are within the range determined by the min percentage and max percentage of flood stage will be paired. `MIN_OBS` and `MAX_OBS` are used to bound the range. Setting the first token to 50 and the second token to 100 would return all forecasts/observation pairs with the observation between 50% and 100% of flood stage, including the endpoints. Flood stage is determined from the vlocation table.

9) **Lead time ranges** - using the `LEADTIME_START` and `LEADTIME_END` tokens, queries will be conditioned to return pairs where the calculation of forecast valid time - forecast basistime falls within the range. As an example `LEADTIME_START="6 hr"` and `LEADTIME_END="12 hr"` returns pairs where (6 <= validtime - basistime <= 12)

Setting the `LEADTIME_END` token to `ALL` will cause the software to loop through a number of lead times equal to 72/<lead time step>, where the default value of lead time step is 6. The user may override this default by specifying a value for the token LEADTIME_STEP.

Using the command `START_LEADTIME` as before and setting the `END_LEADTIME and LEADTIME_STEP` as

```
END_LEADTIME=ALL
LEADTIME_STEP=24
```

The program will run for the leadtime ranges set as:

Leadtime start = start_leadtime , Endtime = start_leadtime + 24 hr
Leadtime start = start_leadtime + 24hr , Endtime = start_leadtime + 48 hr
Leadtime start = start_leadtime + 48hr , Endtime = start_leadtime + 72 hr

Output will then be written to files with the leadtimes appended to the end.

Q2.pairs_24hr, Q2.pairs_48hr, Q2.pairs_72hr.
Q2.stat_24hr, Q2.stat_48hr, Q2.stat_72hr.

The following table lists the tokens that can be used in pairing:

TOKEN	EXAMPLE	DEFINITION
START_OF_PAIRING	true	Pairing group start flag
IHFSDB	hd1_2tua	IHFS database name. Required if $(vsys_ihfsdb) is not set.
VDB	vdb1_1tua	Verification database name. Required if $(vsys_vdb) not set.
LOCATION	“QUAO2" “QUAO2,WTTO1,RTTA4" ALL	Location identifier must be enclosed in quotes. Multiple locations separated by comma. ALL specifies all lids in vrivergaugeloc table used. LOOP_FOR_ALL_LIDS specifies all lids in vrivergaugtloc table used, but compute statistics independently for each lid.
PE_CODES	HG	2 character physical element codes. Multiple codes separated by commas.
TS_CODES	FF	2 character type source codes. Multiple codes separated by commas.
START_TIME	"2000-02-27 00:00:00"	Start time to extract pairs from. Must be enclosed with quotes. Format is YYYY-MM-DD HH:MM:SS
END_TIME	"2000-02-28 00:00:00"	End time to extract pairs to. Must be enclosed with quotes. Format is YYYY-MM-DD HH:MM:SS
PAIRING_WINDOW	“3 hr”	Interval at which to look for forecast and observation pairs. The difference between the forecast basis time and observed time cannot differ by more than this amount in order for the forecast and observed to be paired.
RFC	NWRFC,OHRFC ALL	RFC token to query all lids in verification system within RFC . ALL specifies that the program loop through 13 RFC and append the RFC name to the output file name. ex: test.pairs_NWRFC, test.pairs_OHRFC.

RIVERRESPONSE	FAST ALL	River response used to determine which locations will be used in the pairing query. Valid tokens are "FAST", "MEDIUM" and "SLOW". “ALL” speciifes that the software loop through all three options. It will append the river response to the end of the output file. ex: test.pairs_FAST, test.pairs_MEDIUM, test.pairs_SLOW
LEADTIME_START	“6 hr”	Starting lead-time for selecting forecasts. Lead-times between LEADTIME_START and LEADTIME_END will be paired. Format is "Value Unit" where unit can be "hr", "hour", or "min" Lead-time is calculated from the forecast as validtime-basistime.
LEADTIME_STEP	6	Difference to use between successive lead times if LEADTIME_END is ALL. It is an integer between 0 and 72 that divides 72 evenly. If not specified, the default value is 6.
LEADTIME_END	“12 hr” ALL	Ending lead-time for selecting forecasts. Lead-times between LEADTIME_START and LEADTIME_END will be paired. Format is "Value Unit" where unit can be "hr", "hour", or "min". Lead-time is calculated as validtime-basistime. When ALL is set and LEADTIME_STEP is n, the leadtimes used are [0, n], [n+1, 2n], [2n+1, 3n], and so on until you reach 72.

MIN_OBS	100	Only observations equal to and above `MIN_OBS` percent of the flood stage will be selected. This token combined with the `MAX_OBS` token allows users to select only those pairs that have an observation within some range. Pairs will be selected which contain observation data which is greater than or equal to MIN_OBS percent of the flood stage and less than or equal to `MAX_OBS` percent of the flood stage: (obs_max/100)*fs >= ***OBS*** >= (obs_min/100)*fs Both tokens are required to set this condition.
MAX_OBS	1000	Only observations equal to or below `MAX_OBS` percent of the flood stage will be selected to make pairs. This token combined with the `MIN_OBS` token allows users to select only those pairs that have an observation within some range. Both tokens are required to set this condition.
MIN_FCST	100	Only forecasts equal to or above `MIN_FCST` percent of the flood stage will be selected. This token combined with the `MAX_FCST` token allows users to select only those pairs that have a forecast within some range. Both tokens are required to set this condition.

MAX_FCST	1000	Only forecasts equal to or below MAX_FCST percent above the flood stage will be selected. This token combined with the MIN_FCST token allows users to select only those pairs that have a forecast within some range. Both tokens are required to set this condition.
BUILD_PAIRS	(file1,file2,APPEND)	Pairing call. File1 specifies ASCII output pairing file, file2 specifies ASCII statistics output file, third character flag is APPEND or CREATE. APPEND and CREATE indicate if a new file should be created or if output should be appended to an existing file.
END_OF_PAIRING	true	Pairing group end flag

4.2.1 Additive Adjustments

Prior to statistics being calculated on pairs of forecasts and observations, a static table is read in the Verification database - AddAdjust. For those locations found in this table, the value in the adjustment field is added to the observation. Statistics are then calculate on the forecast and adjusted pair.

$$AAE = \frac{1}{n}\sum_{n} \left| fcst - obs \right| \qquad AE = \frac{1}{n}\sum_{n} (fcst - obs)$$

4.2.2 Statistics

The BUILD_PAIRS token, described above, allows the user to calculate statistics on the forecast/observation pairs returned from the user defined criteria. Any forecast/observation pairs that has a value that is less than -99 will not be used in calculation of the statistic data. Currently the Root Mean Square Error (RMSE), the Average Error (AE), and the Average Absolute Error (AAE) are calculated. These statistics are defined as follows:

4.2.3 Output files

ASCII format pair files and statistic files can be generated in the pairing routines. The format pairs is pipe ("|") delimited for ease in importing into spreadsheets and other databases. The filename is user defined with the BUILD_PAIRS token or can default to paired_YYYYMMDD_HHMMSS where the date is the execution time. The file is written into $vsys_output.

The format has header lines defined by the "#" which are taken from user specified criteria for pairing. The paired information is taken from the verification forecast and observed tables.

```
# Start_time=YYYYMMDD
#End_time = YYYYMMDD
#RFC = rfc
#River_Response = River Response(s)
#Leadtime = leadtime(s)
#Forecast Range = [min forecasts, max forecasts, max
observations, min observations]
lid | fcst_dur | fcst_ts | fcst_extremum | fcst_probability |
basistime | validtime | fcst_value | obs_pe | obs_dur | obs_ts
| obs_extremum | obstime | obs_value | shef_qual_code |
quality_code
```

```
# Start_time=YYYYMMDD
#End_time = YYYYMMDD
#RFC = rfc
#River_Response = River response(s)
```

$$RMSE = \sqrt{\frac{1}{n}\sum_{n}\left(fcst - obs\right)^{2}}$$

```
#Leadtime =
leadtime(s)
#Forecast Range =
[min forecasts,
max forecasts, max
observations, min
observations]
lid | fcst_dur | fcst_ts | fcst_extremum | fcst_probability |
basistime | validtime | fcst_value | obs_pe | obs_dur | obs_ts
| obs_extremum | obstime | obs_value | shef_qual_code |
quality_code
```

The format for statistic files includes similar header information as the pairs files, followed by the RMSE, AB, AAB, and number of samples.

```
#Start time=YYYYMMDD
#End time = YYYYMMDD
#RFC = rfc
#River Size = river size
#Leadtime = leadtime
#Flood_stage = [forecasts above, forecasts below, observations
above, observations below]
RMSE = rmse
AvgBias = ab
AvgAbsBias = aab
Number of Samples = number
```

The format for the statistic files generated for ingesting into a spreadsheet (i.e. the file with the same filename except "_tab" concatenated to it) contains the same information as the normal statistic files, except that everything is delimited with a |. The first line specifies the column headers for each column of the spreadsheet. All additional lines provide the data.

```
Start Time|End Time|RFC|River Response|Pairing
Interval|Leadtime Start|Leadtime End|Location|Max Fcst|Min
Fcst|Max Obs|Min Obs|RMSE|MeanErr|MeanAbsError|Count
```

4.3 Update Location

Within the Verification database, the location information will be stored in two tables, the first a general location table and the second a table for River Gauge location specific information.

The software has an option to update the vlocation table and the vrivergaugeloc table both of which are in the Verification database, with data acquired from the IHFS database. No updates to the IHFS-DB will be made. Based on the location ID's in the Verification database vrivergaugeloc table the update location function will update the location information in the Verification database from the IHFS-DB. To update data in vlocation table run the following command.

```
verify -commands update_loc.in
```

Command file update_loc.in:

```
START_OF_UPDATE_LOCATION=true
IHFSDB=hd1_2tua
VDB=vdb1_1tua
UPDATE_LOCATION=ALL
END_OF_UPDATE_LOCATION=true
```

Location information that is not currently stored in the IHFS_DB will have to be entered by the user using SQL statements until a user interface can be developed.

The following tokens can be used in updating location information

TOKEN	EXAMPLE	DEFINITION
START_OF_UPDATE_LOCATION	true	Update location group start flag
IHFSDB	hd1_2tua	IHFS database name. Required if $(vsys_ihfsdb) is not set.
VDB	vdb1_1tua	Verification database name. Required if $(vsys_vdb) not set.
LOCATION	"QUAO2" "QUAO2,WTTO1,RTTA4" ALL	Location identifier must be enclosed in quotes. Multiple locations separated by comma. ALL specifies all lids in vrivergaugeloc table used.
END_OF_UPDATE_LOCATION	true	Update location group end flag

4.4 Database Administration

Both the extraction and pairing routines require that observation and forecast tables exist. Currently, data is stored in monthly tables where table names are identified by data type, year, month, and size of table (in months). For example, observation data might be stored in obsht200012_1 which would correspond to December of 2000. Similarly, forecast information would be stored in fcstht2000012_1.

The software will check to make sure that tables corresponding to windows of interest exist. If the tables do not exist, the software will exit.

For security purposes, database administration permissions are required to add new tables. A function was added to the verification software to simplify this task. Users running this token group must have appropriate database permissions. If the user does not have correct permissions, the software will exit.

The user specifies a simple date range formatted on a single token by YYYYMMYYYYMM. The DBA routine will then check each month beginning with the first year/month in the range to see if the observed and forecast tables exist. If they do not, they are then created.

The following tokens can be used in database administration

TOKEN	**EXAMPLE**	**DEFINITION**
START_OF_DBA	true	DBA group start flag
IHFSD	hd1_2tua	IHFS database name. Required if $(vsys_ihfsdb) is not set.
VDB	vdb1_1tua	Verification database name. Required if $(vsys_vdb) not set.
DATE_RANGE	"200001200512"	Date range in format YYYYMMYYYYMM where the first year/month corresponds to the start date, the second year/month to the end date
END_OF_DBA		DBA group end flag

5 DB Schema

The following is a description of the Verification Database (VDB)

5.1 Naming Convention

The naming convention for the Verification database is to use the following pattern.

vdb*version**rfc***

For example, if the version number is 1.1 and the datbase is at portland then the database name will be **vdb1_1tua**.

The databases will be numbered starting at 1.1 in order to avoid using that very confusing 0.

5.2 Schema

The following text describes the tables to used in the Verification database.

```
table VLocation
(
lid       char(8) not null,    -- location id
county    char(20) not null,  -- county name
elev      float,               -- elevation of station
hsa       char(3) not null,    -- hydrologic service area
lrevise   date,                -- date of last change to locid
name      char(25),            -- location name
rb        char(30),            -- river basin
rfc       char(6) not null,    -- river forecast center
state     char(2) not null,    -- state
wfo       char(3) not null,    -- weather forecast office
region    char(20) not null.  - - NWS Region name
);

primary key (lid)

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

table VRiverGaugeLoc
(
lid             char(8) not null,           -- location id
bankfull_stg    float,                      -- Bankfull Stage
warn_stg        float,                      -- Warning Stage
action_stg        float,                    -- Action Stage
fld_stg           float,                    -- Flood Stage
mod_fld_stg       float,                    -- Moderate Flood Stage
maj_fld_stg       float,                    -- Major Flood Stage
rec_fld_stg       float,                    -- Record Flood Stage
bankfull_flow     float,                    -- Bankfull Flow
warn_flow         float,                    -- Warning Flow
action_flow       float,                    -- Action Flow
fld_flow          float,                    -- Flood Flow
mod_fld_flow      float,                    -- Moderate Flood Flow
maj_fld_flow      float,                    -- Major Flood Flow
rec_fld_flow      float,                    -- Record Flood Flow
flow_size         char(6),                  -- River Size (small, med, large)
sensor_1          char(2),                  -- First preferred sensor (TS)
sensor_2          char(2),                  -- Second preferred sensor (TS)
sensor_3          char(2),                  -- Third preferred sensor (TS)
pe_1              char(2)  not null,        -- First PE
pe_2              char(2)  not null,        -- Second PE
pe_3              char(2)  not null,        -- Third PE
pe_4              char(2)  not null,        -- Fourth PE
);

primary key (lid)


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++

table AddAdjust
(
lid               char(8) not null,   -- LOCATION id
pe                char(2)  not null,
dur               smallint  not null,
ts                char(2)  not null,
extremum          char(2)  not null,
adjustment        float
);

primary key (lid, pe, dur, ts, extremum)


+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

The obsht and fcstht tables are named with the following convention. The first string is the fcstht or obsht then the 4 digit year, 2 digit month and 2 digit day of the start of the data in that table. Then an underscore and the number of months stored in the that table. The table named obsht20001201_1 will be one month of observations beginning on December 1, 2000.

```
table obsht***YYYYMMDD_M***
(
lid                 char(8)   not null,
pe                  char(2)  not null,
dur                 smallint   not null,
ts                  char(2)  not null,
extremum            char(1)  not null,
obstime             datetime year to second  not null,
value               float,
shef_qual_code      char(SHEF_QC_LEN),
quality_code        integer
);

primary key   (lid, pe, dur, extremum, obstime);
```

+++

```
table fcstht***YYYYMMDD_M;***
(
lid                 char(8)   not null,
pe                  char(2)  not null,
dur                 smallint          not null,
ts                  char(2)  not null,
extremum            char(1)  not null,
probability         smallfloat        not null,
basistime           datetime year to second  not null,
validtime           datetime year to second  not null,
value               float
);

primary key (lid, pe, dur, ts, extremum, basistime, validtime);
```

+++

```
table StaticParms       -- reference-only
(
extract_interval            datetime hour to second  not null,
Pairing_interval            datetime hour to second  not null,
obs_greater_flood_stage     float   – same as MAX_OBS
obs_less_flood_stage        float   – same as MIN_OBS
fcst_greater_flood_stage    float   – same as MAX_FCST
fcst_less_flood_stage       float);   – same as MIN_FCST
```