I M P L ©

"M a k i n g O p t i m i z a t i o n a n d E s t i m i z a t i o n F a s t e r , B e t t e r , S m a r t e r !"

Output Modeling Language (OML)

OML© Output Deployment Manual

i n d u s t r i @ l g o r i t h m s

"IMPLementing Industrial Optimization & Estimization Applications Faster, Better, Smarter!"
(Better Data + Better Decisions = Better Business)

Release 1.10, January 2026, *IAL-IMPL-OML-RM-1-10*

Introduction

OML (Output Modeling Language) is a straightforward and simple way to output, export, print, write, retrieve, review, report, return and most importantly integrate IMPL (Industrial Modeling & Programming Language) solution data for calc-scalars, data-vectors, text-strings and variable values (validations), results or responses sent to one or more output files of the user's, modeler's or analyst's choice with any path name, file name and file type in CSV (comma separated value) format. If an OML output file name does not contain a path name prefix string i.e., there are no colons ":", forward or back slashes "/" or "\" character symbols / delimiters, then the OML keyword **"PATHNAME = "** may be used to specify it, else the default or starting path name for IMPL output is the same path name as the IML fact flag / subject signal which can be reset back to its default or original by configuring PATHNAME to be blank e.g., PATHNAME = or PATHNAME = "". **Please note that although the maximum character length of an OML output filename is limited by the computer operating system, if the OML output filename is to be used as a Microsoft Excel sheet or tab name for example, then its length is limited to thirty-two (32) characters only following the character length limit of Microsoft Excel.**

One of the primary purposes of OML is support the integration of calculation, constant and solution data which may be transferred and used from one problem or sub-problem (i.e., logistics or quality) to another (i.e., quality or logistics) and in IMPL terms this is what IMPL refers to as conjunction or congruence data. The OML file is processed by the IMPL Interfacer (cf. its face argument set to the enumeration IMPLoutput found in the IMPL.hdr file) and is called at the end of the IMPL Console program execution as well as during each logistics- / integer-feasible callback of the MIP logistics solver when incidental and/or intentional intermediate logistics-feasible solutions are found during the MIP solve. These OML files may also output the foreign-variables and foreign-constraints (e.g., Xnnn / Xname and Fmmm / Fname) found in the ILPet / ILP / INPet / INP foreign-files.

OML may be considered as a basic and simplistic type of business intelligence reporting tool in that it accesses IMPL's global, common, universal or shared internal memory-resident sparse data structures (i.e., its resource-entities for sets, parameters, variables, constraints, etc.) and outputs, writes, exports, prints, views, reports, returns or messages them into an adhoc, bespoke, custom or user defined arrangement and in either row- or column-oriented format (cf**. "COLUMNS = -1, 0 or 1"**). OML may also be considered for other types of systems integration such as ETL (extract, transform and load) and

MQTT (message queue telemetry transport) as part of the Industrial Internet of Things (IIoT), Industry 4.0, Smart Manufacturing and what we call Industrial AI (Algorithms and Integration).

The OML file may be used to assist in performing basic “post-processing” or “post-programming” of the calculation, constant and variable solution data i.e., after a computation, optimization, estimization or simulation solution has been found when recursive, successive or iterative computations are required to be performed by dynamic, nonlinear and/or implicit data calculations. Since any variable response or result value may be output with a tagname or identifier (see below), this can be included into another post-programming IML (Industrial Modeling Language) file containing concatenation or companion data (include files) and calculation / constant / computation data (calc-scalars and data-vectors) and then run or executed using IMPL with no solver selected (i.e., fork= IMPLsolverless) post or after any optimization, estimization / estimation or simulation has been performed. Then, a post-programming OML file can be configured to output the final results as required. Typically, the IML file performs “pre-programming” data calculations and computations before the modeler and solver processing begins though in this case we are using IMPL’s calculation and computation data handling capability to post-program using a second IML and OML file.

Furthermore as described below, calc-scalars and data-vectors may also be calculated / computed during the post-processing of the OML file directly (cf. CALCDATA = 1). However, for more advanced post-programming calculations / computations especially involving do- / for-loops and sophisticated multi-line if-then-else statements, it is recommended to retrieve the optimization, estimization / estimation and simulation solution data directly from the internal IMPL memory using IPL (Industrial Programming Library / Language) which is callable from any computer programming or scripting language that can invoke or call dynamic link or shared libraries (DLL’s / SO’s). With IPL, both the IML and OML files are not necessarily required unless you are using “mixed-language programming” i.e., configuring a problem or sub-problem using both IML / OML and IPL in any combination which is made possible by the fact that both IML / OML and IPL receive and retrieve the same single-sourced, shared and common in-memory IMPL data structures. Also see OML’s WRITEFCN() functions which may be programmed in C, C++ or Fortran and can be easily programmed to output, write, print or export calc-scalar, data-vector and/or text-string data in any user, modeler or analyst defined file and format.

OML

The semantic types or codes for the QLQP variables that may be output by OML are as follows using two-letter OML mnemonics where the “D” and “DD” suffixes stand for 1-norm (absolute, Manhattan) and 2-norm (squared, Euclidean) deviations respectively and the “E” suffix stands for excursions (artificial, elastic, error, infeasibility breaker, safety, etc. variables):

```
Quantity:  FL = flow,
           HU = holdup,
           YD = yield,
           FC = factor,

           FLD  = flow deviation 1-norm,
           FLDD = flow deviation 2-norm,
           HUDD = holdup deviation 2-norm,

           FLE = flow excursion,
           HUE = holdup excursion

Logic:     SP   = setup,
           SU   = startup,
           SV   = switchover,
           SD   = shutdown,

           FS   = fillswitch or fillstatus,
           DS   = drawswitch or drawstatus
           DFS  = drawfillswitch or drawfillstatus
           DFS2 = drawfillswitch2 or drawfillstatus2
           DFS3 = drawfillswitch3 or drawfillstatus3

Quality:   DN = density,
           CM = component,
           PP = property,
           CN = condition,
           CO = coefficient,

           DNDD = density deviation 2-norm,
           CMDD = component deviation 2-norm,
           PPDD = property deviation 2-norm,
           CND  = condition deviation 1-norm,
           CNDD = condition deviation 2-norm,

           DNE = property excursion,
           CME = property excursion,
           PPE = property excursion,
           CNE = condition excursion
```

The format of the OML file and its statements are provided below where comments are output or echoed “as-is” when they appear directly after the first OML output file name and are useful to provide leader and trailer features for data integration such as post-processing using concatenation (include files) and calculation / constant data as mentioned above. However, when **“COLUMNS = 1”** comments should be avoided or not included in the OML file as only 2D tabular or column-oriented data will be output where unfortunately comment or blank lines will be appended to the end of the data matrix, table, grid or block distorting the output. If it is required to ignore, passover or skip these as-is comment lines, rows or features, then the OML keyword **”COMMENTECHO = 0”** maybe configured, set or specified anywhere in the OML / OMLet file. The default is COMMENTECHO = 1 (or any other number other than zero), which will result in these as-is comments being output to the configured OML output file.

OML output file names must explicitly include its file extension / type (i.e., file name = “OutputFile.dta”, “OutputFile.adt”, “OutputFile.dat”, “OutputFile.csv”, etc.) where any number of output file names (OutputFile2.dta, OutputFile3.dta, etc.) may be placed in the same OML file i.e., there is no limit to the number of OML output files that can be configured / specified. The output file names must not contain any commas (“,”) in their name for obvious reasons. If a line, row or feature is recognized as a calc-scalar or data-set / -list / -1D-array / -vector name configured and known from the corresponding IML file, IPL functions or the IMPL Console’s formulas and formulasfile flag, then it is output “as-is” with its (integer index number and) real number value, where as mentioned, this is useful for post-processing as a tagname, time-/trial-index (timestamp), value (TTV) tuple where the value may also be a tuple i.e., multiple values also separated by commas (see **“DATAFORMAT = 0”**). **It is important to highlight that if no OML output files are configured / specified, then OML output will simply be outputted, exported, printed or written to the IMPL Console (i.e., standard output) if it exists or the IMPL log file.**

To retrieve the objective function term variable values individually, specify the keywords **`PROFITOBJ`**, **`PERFORMANCE1OBJ`**, **`PERFORMANCE2OBJ`**, **`PENALTYOBJ`** and **`TOTALOBJ`** anywhere in the OML file where the keywords **`INCDIV`** and **`INCDIVTERM`** retrieve the Industrial / Incumbent Diversification Search’s (IDS’s) incumbent diversification variable and its objective function contribution term. Selected IMPL Server signal and statistic keywords found in the IMPL.hdr header file are also available such as **`OBJVALUE`** (same as **`TOTALOBJ`**), **`MIPCHKSUM`**, **`MIPCHKSUM0`**, **`ECLOSURE1`**, **`ECLOSURE1MAX`**, **`ECLOSURE1MAXI`**, **`ECLOSURE2`**, **`ECLOSURE2MAX`**,

ECLOSURE2MAXI, **ECLOSUREOO**, **ECLOSUREOOMAX**, **ECLOSUREOOMAXI**, **ICLOSURE1**, **ICLOSURE1MAX**, **ICLOSUREOOMAXI**, **ICLOSURE2**, **ICLOSURE2MAX**, **ICLOSURE2MAXI**, **ICLOSUREOO**, **ICLOSUREOOMAX**, **ICLOSUREOOMAXI**, **STATUS**, **STATUSSIGNAL**, **SCROLL** (a.k.a. the folio flag), **SELECTIVE** (a.k.a. the furcate flag) and **SEEK** (a.k.a. the fish flag).

In addition, specific computed attributes of the problem's solution may also be retrieved such as **MAXVETISTIC** which outputs the maximum, largest or worst absolute vetistic for data reconciliation and regression (estimization) problems known as the maximum-power measurement test (MPMT) statistic for gross error detection and identification (GEDI). The auxiliary keywords **MAXVETISTICX** conveniently output only the foreign-variable's index number nnn in Xnnn (or v1_X(nnn,1)) unless a corresponding text-string value for the nnn is known or exists which can provide a mapped, assigned, associated or attached tagname for example. The **MAXVETISTIC / MAXVETISTICX** keywords may also be parameterized with an optional "=" equals sign to specify a single validation item (cf. "value" = 0, "viability / validation" = 1, "variance" = 2, "vetistic" = 3, etc.) and an optional tagname which labels or tags the item accordingly. A tagname is also available for **TOTALOBJ** as well as **INCDIV** and **INCDIVTERM** and if a tagname exists and **CALCDATA = 1**, then calc-scalars with calc-scalar expressions or formuals are supported for both these keywords i.e., a calc-scalar is created, generated, inserted or received using the supplied tagname as its calc-scalar name whereby calc-scalar expressions or formulas may include these total objective function, incumbent diversification variable / term and maximum vetistic values.

To expose the "worst" variable or constraint that are identified when the problem or sub-problem is detected to be infeasible or inconsistent by IMPL Server's shrinkability() routines or IMPL Presolver's SLPQPE() routine, use the keywords **WORSTVARIABLE** and **WORSTCONSTRAINT**. These keywords allow the worst variable or constraint names to be extracted from the IMPL Console log and exposed in any OML output file i.e., the variable or constraint name associated with the largest or maximum result, response or residual value.

To record the current system date and time with an optional user, modeler or analyst supplied tagname in any of the OML output files, the keywords **DATETIMENOW**, **DATENOW** and **TIMENOW** may be configured similar to the IML data functions DATETIMENOW(), DATENOW() and TIMENOW(). The

formats for the date and time respectively are ***yyyy,mm,dd*** and ***hh,mm,ss,mss*** where mss are the milli-seconds with three (3) digits and an optional tagname for the dates / times are supported.

For convenience, to output all of the calc-scalars and/or all of the data-vectors, include the OML keywords **`ALLCALC / ALLCALCS`** and **`ALLDATA / ALLDATAS`** where the data-vectors are written out in the implied **"COLUMNS = 0"** and **"DATAFORMAT = 0"**.

The lines with the UOPSS, QLQP, time tuple and tagname keywords below may be configured in any order and any line may contain IMPL's standard in-line comment character ("!") where all characters after and including the comment character will simply be stripped or removed from the OML statement line or row. Comment characters found in exactly column one (1) in the OML file will simply be output, echoed, shadowed or written "as-is" to the particular output file. Calc-groups, data-groups and text-groups statements, including calc-lists and text-lists, will output all calc-scalar/data-vector/text-string member assignments for each group when encountered where re-assigning or re-mapping a calc-name / data-name / text-name to a tagname is not available. For calc-groups and text-groups, **"COLUMNS = 0 or 1"** are supported and for data-groups, **"COLUMNS = -1, 0 or 1"** including the **"DATAFORMAT = 0, 1, etc."**. Specifically for calc-groups / -lists and text-groups / -lists only and when **COLUMNS = 1**, if the keyword **SKIPNAMES = 1**, then no calc-scalar nor text-string names will be written, output, exported or printed i.e., only their values will be outputted.

```
Output filename
…
Comment, Calc,tagname or Data[1..n],tagname or Text,tagname or Calc-/Data-/Text-Group
…
uname,oname,QLQP,time/trial,tagname
…
Comment, Calc,tagname or Data[1..n],tagname or Text,tagname or Calc-/Data-/Text-Group
…
uname,oname,qname,QLQP,time/trial,tagname
…
Comment, Calc,tagname or Data[1..n],tagname or Text,tagname or Calc-/Data-/Text-Group
…
uname,oname,pname,sname,QLQP,time/trial,tagname
…
Comment, Calc,tagname or Data[1..n],tagname or Text,tagname or Calc-/Data-/Text-Group
…
uname,oname,pname,sname,qname,QLQP,time/trial,tagname
…
Comment, Calc,tagname or Data[1..n],tagname or Text,tagname or Calc-/Data-/Text-Group
…
uname,oname,pname,sname,uname,oname,pname,sname,QLQP,time/trial,tag
…
Comment, Calc,tagname or Data[1..n],tagname or Text,tagname or Calc-/Data-/Text-Group
```

The unit-operation-port-state combinations and the quality names must of course be known and consistent with the UOPSS-QLQP© / UQF© names that have been configured to construct the problem or sub-problem before the modeling. The "time" (or integer "timestamp") field is either a single time-period / trial-period index (integer number) or a time-period / trial-period index slice, interval, range or plank indicated by IMPL's special dot dot ellipsis or double-period delimiter symbol ".." (two periods in-series) i.e., "time1..time2" (or "trial1..trial2") where time2 >= time1 and the tagname string field can be used as a simple alias, label or nick-name to address, reference, index, point or map to the output data. If the tagname is absent, then a unique name is automatically created / derived / generated using the UOPSS-QLQP© / UQF© names or keys. If the UOPSS-QLQP© / UQF© names and/or the time-period indexes are not recognized or known, then no result is output but the line is echoed in the user's, modeler's or analyst's chosen output file to indicate which UOPSS-QLQP© / UQF© names or keys are not recognized by OML. **It should be said that the second time- / trial-period index specified may exceed the total number of periods available as it is appropriately truncated. However, the first period index must not exceed the number of time- / trial-periods in the past and present time-horizon or -profile though it may be less. If it does exceed the past and present horizon, then the OML line, row or feature is echoed into the output file.**

In addition, OML has the capability for the time-plank, time-interval or time-range for the starting, initial or beginning time-/trial-period index and the finishing, final or ending time-/trial-period index separated by the well-known dot dot ellipsis or the double-period characters ".." to be calc-scalar / data-vector-element expressions i.e., "time1-expression .. time2-expression" and "trial1-expression .. trial2-expression". However, when configuring the time-plank calc-scalar / data-vector-element expressions, the ending, appended or suffixed comma "," is required after the time2- or trial2-expression. The ".." is also available for the data-set / -list / -vector where the beginning or starting and the ending or finishing indices / iterators within the square brackets and dot dot ellipsis (i.e., "[index1-expression .. index2-expression]") may also be calc-scalar / data-vector-element formulas.

Particular to the data-sets / -lists / -vectors, OML has the "**DATAFORMAT = 0"** (default) which will output the data-vector using the tagname, time-/trial-index (timestamp), value (TTV) format identical to the output of UOPSS-QLQP© / UQF© variables. If "**DATAFORMAT = 1"**, then OML will write out the data-set, -list or -vector in the format that can be easily imported, inputted, loaded or read back into

IML via its data frame (`&sData,@sValue`) which has no time-/trial-index and only the first instance of the tagname or data name is output followed by the values separated of course using commas (",") on separate lines or rows (cf. IML's constant data frame). However, it is salient to note that the **"DATAFORMAT = 1"** is not respected when **"COLUMNS = -1 or 1"** as this will output the data-vector's data-elements / -points / -rows on the same line, row or feature or in a 2D-array, grid, block, table or matrix. Furthermore, if **"DATAFORMAT = 2, 3, 4, etc."**, then OML will output the data-vector in what is known as pipe-separated value (PSV) format, as opposed to comma-separated value (CSV) format, where the pipe is the "|" symbol character which is compatible with IML's constant data. The right-hand-side (R.H.S.) integer number value of the **"DATAFORMAT"** determines the number of pipe-separated values to be outputted, exported, printed or written per feature, line or row.

Specific to UOPSS-QLQP© / UQF© related output, it is useful to note that since qualities are only assigned, associated or attached to physical pool units and not necessarily to pool unit-operations in IMPL, and therefore to output qualities for the projectional pools, please specify the pool unit-operation names as there is no unit name only keyword line in OML. It is also important to note that to output "static" coefficients (CO's), the time field must be configured as the negative of the total number of time-periods in the past / present time-horizon or greater. For example, if the number of time-periods in the past / present time-horizon is four (4) including the zeroth (0^{th}) time-period i.e., -3, -2, -1, 0, then the time field for the static coefficient should be entered as negative four (-4) or smaller (i.e., -5, -6, -7, etc.). For convenience, simply specify a large negative number with an absolute value larger than the number of time-periods in the past / present time-horizon, for example, "-100", "-1000", etc.

In order to manage multiple situations, suspensions, substitutions, samples, surveys, snapshots, sub-solutions or cases, IMPL allows for a **"#if nnn | nnn2"** followed by a **"#endif"** to be included anywhere in the OML file but must be found starting exactly in column one (1) only and located in any row. The "nnn" and "nnn2" are a plus or minus integer literal number and/or calc-scalar / data-vector-element expression which, if it matches the value specified in IMPL's furcate flag, then the rows contained within the "#if nnn | nnn2" and "#endif" will be included else they will simply be ignored, skipped or passed-over. The "nnn" and "nnn2" 32-bit integer numbers can be likened to index numbers for the specific situation, suspension, substitution, sample, survey, snapshot, sub-solution or case where the optional "|" represents the OR relational logic identical to that found in IML and ILP / INP.

Typically, IMPL will output an objective function frame or rows at the very top of the configured output file summarizing the terms of the objective function, however this can be changed as follows using **"OBJROWS"**. If in the OML file a **"OBJROWS = 1"** line (with arbitrary spacing) is found, then OML will output the objective function rows in the output file. The default is **"OBJROWS = 0"** which will output the objective function rows but with a prefixed "!" comment character. If **"OBJROWS"** is any other number such as **"OBJROWS = -1"**, then no objective function rows will be output. Outputting with no objective function rows is especially useful when the output file name has the file type or extension of "*.csv" as Microsoft Excel will automatically open these files and convert the comma separated values (CSV) into cell values. Please note that the **"OBJROWS"** must be found before the output file is declared or configured, else the default or the previously set **"OBJROWS"** configuration value will be used.

If in the OML file a **"LOGFILEECHO = 1"**, or any number other than zero (0) is found, then OML will simply echo output these lines to the IMPL log file or the IMPL Console window / screen (cf. the IMPL USELOGFILE setting); the default is **"LOGFILEECHO = 0"**.

When the IMPL Console folio flag, which is equivalent to the IMPL Server's scroll signal are non-zero, OML will suffix double underscores "__nnn" to the file name and the folio / scroll$ number nnn to the OML output file name for persistence and retention when **"APPENDFILE = 0"** (default). However, if the keyword assignment **"APPENDFILE = 1"**, then OML will append the output content into one single output file without the suffixed "__nnn". The **"APPENDFILE = 1"** keyword setting is useful to run or execute IMPL for many instances (e.g., from a Microsoft DOS batch file or the like) or to retrieve many logistics-/integer-feasible solutions incidentally or intentionally and to retain or record the contents of the selected OML output data into the same (appended) file. **Please take note that all of OML output files that are appended (APPENDFILE = 1) should be configured before all other files that are not appended (APPENDFILE = 0) i.e., are to be located or placed at the top or start of the OML file.**

Foreign-variables and foreign-constraints configured in the *.ilp/*.ilpet and *.inp/*.inpet foreign-files (scalar-based algebraic or mathematical modeling) may also be output by configuring **`v1_X,index,tagname`** and **`c1_F,index,tagname`** or **`v1_X,index1..index2,tagname`** and **`c1_F,index1..index2,tagname`** anywhere in the OML file where **`index`**, **`index1`** and **`index2`** may be either literal integer numbers or calc-scalar / data-vector-element expressions or formulas. On output, the integer index number is suffixed or appended directly to the tagname and a

comma delimiter separates the tagname from the real value (i.e., consistent with DATAFORMAT = 0). If **DATAFORMAT /= 0** or **<> 0**, that is the DATAFORMAT does not equal zero (0), then OML will output the foreign-variable or -constraint with its index number suffixed or appended to **v1_X / c1_F** or to whatever **tagname** is specified. This will allow these outputted foreign-variables / -constraints to be imported, read or inputted back into IML using the calc frame (`&sCalc,@sValue`). In addition, **v1_X,substring,tagname** and **c1_F,substring,tagname** are also supported with the sub-string name or sub-name representing a root, base or source name fragment whereby all matches found are systematically output, written, exported or printed to the appropriate OML output file where the retrieved / returned results or responses are sorted in ascending or increasing order / sequence based on the foreign-variable / -constraint name. For more specific outputting, printing, exporting or writing, **v1_X,index1..index2,tagname** and **c1_F,index1..index2,tagname** are also supported when an index or indice range can be configured where **index1** and **index2** may also be either literal integer numbers or calc-scalar / data-vector-element expressions or formulas and all matches of **substringindex1..substringindex2** are found using IMPL's leading-zero indexing.

As well, the OML keywords **ALLX / ALLXS** and **ALLF / ALLFS** conveniently output all of the foreign-variables and -constraints to the OML output files in the same format as the EXL file. Both **"COLUMNS = 0"** and **"COLUMNS = 1"** are supported with these keywords especially when the data reconciliation and regression diagnostics (cf. "viability / validation", "variance", "vetistic", etc.) are computed for estimization problems. These keywords may also be parameterized using the "=" equal sign to specify a single validation item to be output i.e., **ALLX / ALLXS = 1, 2, 3, 4, 5, 6 or 7** and **ALLF / ALLFS = 1, 2, 3, 4, 5 or 6** where item number seven (7) is the "detectability" metric known in IMPL as the "validation2" attribute computed for each metered, measured or reconciled variable (i) *detectability_i = SQRT(1.0 – reconciled_variance_i / raw_variance_i)* taken from the book Narasimhan and Jordache, "Data Reconciliation & Gross Error Detection", *Gulf Publishing Company*, 2000. Detectability is related to the notion of practically or statistically non-redundant where values closer to zero (0.0) are more statistically non-redundant than values closer to one (1.0) i.e., if the reconciled_variance_i = raw_variance_i the detectability_i = 0.0 and is statistically or practically non-redundant. Similarly, there is also the notion of statistically unobservable unmeasured or regressed variables when its 95% confidence-interval spans or includes zero (0.0) as opposed to being structurally unobservable as indicated by its validation / viability diagnostic.

Finally, the OML keyword **NUMBERFORMAT** = E20.10E3 (default) is available to change or modify OML's default Fortran format specification sub-string, at any location or position in the OML / OMLet files, for all real numbers written, printed, exported or outputted by OML to any valid Fortran format specification sub-string (cf. F, E, EN and G) using Intel Fortran's variable or run-time formatting capability. For example, NUMBERFORMAT = F18.10 would write IMPL's or IMPL-DATA's real number values with 18 possible digits padded by leading spaces and 10 numbers after the decimal point as opposed to the default output in scientific notation with an exponent.

OMLet Sub-Files (Like IMLet Include Files But w/ No Include Frame)

At any point or location in the OML file, multiple OMLet sub-file names may be configured by specifying the OMLet sub-file name appended with the ***.omlet, *.OMLet** or ***.OMLET** file type or extension where the OMLet sub-file name may or may not match the fact. IML / subject.IML. If the OMLet sub-file cannot be opened for any reason, IMPL will simply proceed to the next OML statement with no error, exception nor warning i.e., OMLet sub-files are optional. And, a default OMLet with the sub-file name **fact.OMLet / subject.OMLet** is expected to be found with the same path name as the fact.OML / subject.OML file. If any OMLet sub-file is not found, then OML will try to open the fact.OMLet / subject.OMLet sub-file as a last resort or final file open attempt. These OMLet sub-files have exactly the same syntax as the OML files and are similar to the IMLet include files found in IML except that no explicit include frame leader / trailer features are required for OMLet sub-files to be recognized. OML processes these OMLet sub-files by straightforwardly diverting or re-directing the OML file reads, imports or inputs to the OMLet sub-file contents until the end-of-file indicator is detected in the OMLet sub-file and then the processing returns back to the OML file contents. **However, unlike the include file frames in IML which supports one level of nested IML include files i.e., an included IMLet file may contain only one other included IMLet file, OML with its OMLet sub-files, does not allow any nesting, cascading or layering of embedded OMLet sub-files i.e., the OMLet sub-file must not contain any other OMLet file references within its contents.**

Automatic Renaming of OML Output File Types w/ "x" and w/o "x"

To facilitate the integration of IMPL with other off-line and on-line informational and operational technology systems (IT / OT), all output data / csv files that are specified by the user, modeler or analyst in the OML file are first created and opened temporally or transiently with an **"x"** appended or suffixed to its file type or extension i.e., *.datx, *.dtax, *.adtx, *.csvx, etc. Upon completion of outputting, exporting or writing all of the IMPL solution data to the respective files, IMPL automatically renames these output files to *.dat, *.dta, *.adt, *.csv, etc.; the same approach is applied to the IMPL *.exl files. This simple technique allows asynchronous waits on each data / csv output file to then process (lex, parse, map, convert, store, etc.) the data and to integrate to other third-party systems once the proper file type and extension is detected.

Column-Output (Block-, Grid-, Matrix-, Spreadsheet-, Table-Format)

If in the OML file a **"COLUMNS = 1"** line (with arbitrary spacing) is encountered immediately after the output file name, then column-oriented output for the UOPSS-QLQP© variable results will be enabled i.e., block, grid, matrix, spreadsheet, 2D-array or table format with rows as time-periods / trial-periods and columns as tagnames. **"COLUMNS = 1"** output is where the rows, lines or features of the table, grid, block or matrix are the range-exhibits of the variables and the columns are the variable tagnames. If a **"COLUMNS = 0"** (default) is found immediately after another output file name, then the conventional output will be used as above i.e., tagname, time-/trial-index (timestamp), value (TTV) format since the time / trial dimension of the variable is also included. The columns are the UOPSS-QLQP© variable results configured as previously described using the tagnames as column names. The rows of the output are the "ending" (as opposed to "starting") time-points for each time-period including all the time-periods in the past / present and future time-horizons. Temporal data for a variable result not configured will be output using IMPL's real Non-Naturally-Occurring-Number (i.e., RNNON = -99999.0) which is to ensure that all variables in the block, grid, matrix, spreadsheet or table have the same number of rows. Column-based output is useful to load, import or integrate into a spreadsheet or database software for further analysis and analytics. And, if a **"COLUMNS = -1",** then the data are transposed with columns as the time- or trial-period dimension (i.e., range-exhibit rows, elements or points) and the rows as the variable tagnames. Data-sets / -lists / -vectors will also be output as a transposed matrix, block, grid or table where the data-elements / -points / -rows are written out horizontally as columns and can be easily transposed or pivoted in spreadsheet software such as Microsoft Excel.

When **"COLUMNS = -1", "COLUMNS = 0"** or **"COLUMNS = 1"** (with arbitrary spacing)**,** the setting **"SKIPZEROS = 0D+0"** is respected where **0D+0** is the default tolerance and may be replaced by any positive real number. This setting will simply not output, export or write out the variable value / result / response when the value is near-zero i.e., below the tolerance specified and is identical to ignoring, disregarding or skipping zeros in a sparse matrix. The keyword **"SKIPZEROS"** is helpful to reduce or minimize the output of UOPSS-QLQP© solution data transferred to the output files especially when the problem is large where fast and efficient integration is required to adjacent systems. Further, if in the OML file, a **"TCOLUMN = 1"** line (with arbitrary spacing) is found, then OML will output a time-period or trial-period in the first column for the temporal or training / testing data index when **"COLUMNS = 1"** only. The default is **"TCOLUMN = 0"** which will not output this first "t" column.

Data Calculation Keyword (CALCDATA) for Direct OML Post-Processing

If **"COLUMNS = 1"** and **"CALCDATA = 1"**, then OML will use the tagname as the IMPL data-set, -list-, -vector or -1D-array name and copy the UOPSS-QLQP© or UQF© variable value, result or response into a data-set, -list or -vector with the size of the data-vector typically of length 1..NTPF where NTPF is the total number of time-periods in the past/present and future time-horizons / -profiles (cf. Industrial Programming Library / Language (IPL) for details on NTP, NTF and NTPF). These data-vectors, converted and identified from the tagname of a UOPSS-QLQP© / UQF© variable solution data, may then be employed in any data calculation function found or contained directly inside the OML file between the frames `&sCalc,@sValue, &sDataCalc,@sValue` and `&sDataData,@sValue` enabling OML's post-processing capability. In addition, any known data-vector may also be included in OML's post-processing as-is when **"CALCDATA = 1"** where **"COLUMNS = 1"** is not necessary as **"COLUMNS = 1"** is reserved only for converting, populating or transforming UOPSS-QLQP© or UQF© variable values to user, modeler or analyst tagname labeled data-sets, -lists or -vectors.

In order for quicker debugging and troubleshooting of these post-processing calculations, if an output *.dta, *.adt, *.dat, *.csv, etc. file is not configured or specified i.e., absent, missing or not present, then OML will write, print, export or output these calculations to the IMPL Console screen, window or terminal for convenience also referred to as the standard output (stdout, cf. stdin and stderr). **Please note that the `&sCalc,@sValue` and `&sDataCalc,@sValue` frames are fully capable but the**

`&sDataData,@sValue` frame is only partially capable where currently only a subset of the datadata frame data functions are available or supported i.e., SLICE, SPLICE, SCATTER, SEQUENT, SWAP, SWAP2, SUBSTITUTE, SKIP, SYNTHESIZE, SHED, SHIFT, STACK, STRETCH and SLIDE keywords and the algebraic data expression calculations although more functionality will be added incrementally as required.

For foreign-models with foreign-variables and -constraints (i.e., v1_X and c1_F), these variables and constraints may also be populated or persisted into data-sets, -lists or -vectors or 1D-arrays when **CALCDATA = 1** and with or without **COLUMNS = 1** provided that there is either *ellipsis-indexing* using ".." or *data-vector-indexing* supplying or providing a known data-set, -list or -vector with user, adhoc or custom index / indice numbers (i.e., the nnn in Xnnn and mmm in Fmmm) where a tagname, if specified, becomes the data-vector's name or identifier symbol. Data-vector-indexing allows for adhoc, custom or user groups, partitions or collections to be specified for any number of v1_X and c1_F result or response values populated, added or inserted into any number of data-sets, -lists or -vectors.

In addition, if in the OML file a data calculation function name (with arbitrary spacing) is encountered immediately after the tagname field and irrespective of the **"COLUMNS"** keyword, then this data aggregation, cumulative or accumulation calculation will be performed respecting the time-periods / trial-periods configured using IMPL's ellipsis delimter **".."**. Currently, only **"NORM1", "NORM2"** and **"NORMOO"** data calculation function names are supported for the 1-norm, 2-norm and oo-norm condition deviation variables on unit-operations. For example, these norm aggregations over the time-/trial-periods can provide the individual contribution to the performance objective function term for each deviation variable sans / without their performance weighting. Please note that more data calculation functions / operations will be incrementally added in this fashion upon request to Industrial Algorithms Limited (IAL).

And as mentioned previously, calc-scalars are received for the **`TOTALOBJ, INCDIV, INCDIVTERM`** and **`MAXVETISTIC / MAXVETISTICX`** when a tagname is supplied with or for these keywords.

WRITEFCN() Write Functions Similar to IML's DATAFCN() Callbacks

Similar to XFCN and DATAFCN user-, modeler- or analyst-coded functions, also referred to as callbacks, WRITEFCN's may also be configured and linked to WRITEFC1, …, WRITEFC9, WRITEFCA, …, WRITEFCZ

whereby the user, modeler or analyst can compile their own C, C++ or Fortran DLL / SO dynamic / shared library functions and call them directly from IMPL in the OML files i.e., adhoc, bespoke, custom or user write functions. These WRITEFCN functions may also be called from the computer programming or scripting language that calls IPL (IMPL-API) as well since they are regular C / C++ / Fortran DLL's or SO's. Essentially, the WRITEFCN may be used to write, print, export or output single or multiple (grouped) calc-scalars, data-vectors and text-strings (optional) in any format via a machine-coded computer programming language. The call statement or signature for these user-, modeler- or analyst-coded / -programmed WRITEFCN's are as follows where all of the 1D-arrays are linearized or vectorized in order to pass or transfer its multiple data-vector-elements, -points, -rows or -values across diverse computer programming and scripting language.

```
integer function WRITEFCN(lun: integer,
                          ncse: integer,
                          cse: real*ncs,
                          ndv: integer,
                          ndve: integer,
                          dve: real*(ndv*ndve),
                          ntse: integer,
                          csn: character(basestringlen)*ncs,
                          dvn: character(basestringlen)*ndv,
                          tsn: character(basestringlen)*ntse,
                          tse: character(basestringlen)*ntse)
```

```
       function writefunc(lun,ncse,cse,ndv,ndve,dve,ntse,csn,dvn,tsn,tse)
#if stdcalling == 1
cDEC$ ATTRIBUTES DLLEXPORT, STDCALL, REFERENCE, DECORATE, ALIAS : "writefunc" :: WRITEFUNC
#else
cDEC$ ATTRIBUTES DLLEXPORT, ALIAS : "writefunc" :: WRITEFUNC
#endif

c * IMPORTANT NOTE * - IMPLserver.mod and IMPLserver.lib are optional.
c
c        use IMPLserver

        implicit none

        integer(4) :: writefunc

c * IMPORTANT NOTE * - IMPLmodeler.fi is optional.
c
c        include "IMPLmodeler.fi"

c Supplied logical unit number for OML-related output, print, export or write statements.

        integer(4) :: lun
cDEC$ ATTRIBUTES VALUE :: lun

c Number of calc-scalar elements (values), their names and the real number elements.

        integer(4), intent(in) :: ncse
cDEC$ ATTRIBUTES VALUE :: ncse
        real(8), intent(in) :: cse(1:ncse)
```

```
cDEC$ ATTRIBUTES REFERENCE :: cse

c Number of data-vectors, their names and the equal number of real number data-vector elements
c (values) for each data-vector in linearized or vectorized form.

        integer(4), intent(in) :: ndv
cDEC$ ATTRIBUTES VALUE :: ndv
        integer(4), intent(in) :: ndve
cDEC$ ATTRIBUTES VALUE :: ndve
        real(8), intent(in) :: dve(1:ndv*ndve)
cDEC$ ATTRIBUTES REFERENCE :: dve

c Number of text-string elements (values), their names and the character(BASESTRINGLEN=64) number
c elements.

         integer(4), intent(in) :: ntse
cDEC$ ATTRIBUTES VALUE :: ntse

        character(64), intent(in) :: csn(1:ncse)
cDEC$ ATTRIBUTES REFERENCE :: csn
        character(64), intent(in) :: dvn(1:ndv)
cDEC$ ATTRIBUTES REFERENCE :: dvn
        character(64), intent(in) :: tsn(1:ntse)
cDEC$ ATTRIBUTES REFERENCE :: tsn
        character(64), intent(in) :: tse(1:ntse)
cDEC$ ATTRIBUTES REFERENCE :: tse

c ... Insert developer user, modeler or analyst code here ...
c ...

c ...
c ... Insert developer user, modeler or analyst code here ...

        writefunc = 0

c ... Insert developer user, modeler or analyst code here ...
c ...

c ...
c ... Insert developer user, modeler or analyst code here ...

      end function writefunc
```

The `lun` argument is the logical unit number provided by OML for the user, modeler or analyst to assign to the OML file that is currently being written to by OML. Else, the user, modeler or analyst may internally in the WRITEFCN() write function, open and attach to a different logical unit number of their choice depending on their requirements. Please note that all of the character string 1D-arrays are purposefully relegated to the end of the call statement's argument list given that when dispersed amongst the other integer and real number arguments, access violations / exceptions arose.

A very important and useful capability of the WRITEFCN() callback's is that if programmed or coded with Intel Fortran (IFX), the IMPLserver.mod, IMPLserver.lib and IMPLmodeler.fi, provided by Industrial Algorithms Limited (IAL) upon request, may *optionally* be used and included when compiling and linking the WRITEFCN() DLL / SO dynamic link library – see also the IMPC Manual. This means that the user, modeler or analyst has complete / full access to all of the IMPL data structures

and routines / methods when OML is called or invoked by IMPL through the IMPLinterfacere() routine.

The OML write function declaration statement below requires the following feature, line or row to be configured first in the OML file and specifies the dynamic or shared link library name (DLL / SO), the write function name contained within the library where the "@" asperand special character indicates the alphanumeric single character number of the write function i.e., **0** (WRITEFCN), **1** (WRITEFC1), …, **9** (WRITEFC9), **10** (WRITEFCA), …, **23** (WRITEFCN), …, **35** (WRITEFCZ).

```
DRIVE:\DIRECTORY\LIBNAME.DLL,FUNCNAME,@
DRIVE:/DIRECTORY/LIBNAME.DLL,FUNCNAME,@
```

Following the above OML statement, the WRITEFCN() may then be called or invoked anywhere in the OML file as many times or instances as necessary where the text-string and text-group argument is optional and can be omitted.

```
WRITEFCN(calc-scalar;data-vector)
WRITEFCN(calc-scalar;data-vector;text-string)

WRITEFCN(calc-group;data-group)
WRITEFCN(calc-group;data-group;text-group)
```

The above call statements highlight that either a single calc-scalar or a calc-scalar-group, either a single data-vector or a data-vector-group or a either single text-string or a text-group in any combination may be specified or supplied where only the text-string / -group is optional.

Data Reconciliation and Regression Diagnostics (IMPL's SORVE)

If the IMPL sub-solver SORVE is called which estimates the observability for unmeasured or regressed variables and coefficients (without targets), redundancy for measured or reconciled variables (with targets) and variances for all variables then these diagnostics will be output for **"COLUMNS = 0"** only and when one (1) and only one (1) time-period/trial-period index is configured using a single "time" (i.e., an integer "timestamp" field) or "time1..time2" where time1 = time2. In addition, these validation diagnostics are also available for the foreign-variables and foreign-constraints when SORVE has been invoked.

The diagnostics are: (1) *"viability" / "validation"* (observability i.e., if near-zero, then observable else if non-zero, then non-observable and redundancy i.e., if non-zero, then redundant, else if near-zero, then non-redundant), (2) *"variance"* (square-root equals standard-deviation), (3) *"vetistic" / "veracity"* (Student-t statistic also known as a t-score or z-score and more specifically the "maximum-power" measurement test), (4) and (5) *"valuations"/"variability"* (nominal 95% confidence-interval), (6) *"violation"* (lower and upper hard bounds excursion or violation check) and (7) *"viability2" / "validation2"* (known as the detectability computed as SQRT(1.0 – reconciled_variance / raw_variance)) respectively (cf. the IMPL.hdr file). If the viability / validation is near-zero (typically less than or equal to 1D-9 for linear problems and numerically larger for nonlinear problems) for unmeasured variables then it is considered as "observable", solvable, bounded or calculatable else if non-zero then it is "unobservable", non-solvable, unbounded or incalculatable. If the viability / validation is non-zero (typically greater than or equal to 1D-9 for linear problems and numerically larger for nonlinear problems) for measured variables represented by their 2-norm deviation variables from target i.e., raw measurement, then it is considered as "redundant" else if near-zero it is "non-redundant". As mentioned, the vetistics are the maximum-power measurement test (MPMT) gross error detection statistics which should be less than or equal to their 95% threshold or critical value which is approximately between 3.5 to 4.0 depending on the number of degrees-of-freedom (DOF) and its Bonferroni / Sidak adjusted significance-level where if the number of measurements is large, then the level of significance (alpha) is adjusted by dividing by the number of measurements. These vetistics are simply computed as the reconciled adjustment, revision or residual divided by its reconciled standard-deviation and in statistics is known as a t- / z-score and are of course subject to the well-known *Type I* and *Type II* errors i.e., **false positive / false alarm** and **false negative / missing alarm** respectively. For interest, possible short names, aliases, labels, nick-names, tags or identifiers for the reconciliation data and diagnostics are as follows: **RAW** = raw measured value, **REV / RES** = revision / residual or adjustment value, **REC** = reconciled or estimated value, **RED** = redundancy viability or validation, **MPZ / VET** = 95% maximum-power z-score vetistic, **REG** = regressed or unmeasured estimated value, **RIG** = fixed or rigid value, **OBS** = observability viability or validation, **CI1 / CI2** = 95% confidence-interval or 5% significance-level valuations or variabilities and **LUV / LUX** = lower and upper bounds violation or excursion check.

It is worth mentioning that there is also a vetistic for the total objective function (i.e., the 2-norm performance term) which is known as the overall, collective, global or total gross error detection statistic for the weighted quadratic data reconciliation objective function with a 95% crititcal or threshold value automatically computed by IMPL as a Chi-Square (X2) statistic (found in the vetistic field). The degrees-of-freedom (DOF) for X2 critical or threshold value are calculated as the *number of equality constraints or equations (ng) minus the number of independent unmeasured / regressed variables (ny12)* conveniently provided in the variance field and in the valuation1 / variability1 field is the *number of independent unmeasured / regressed variables* where DOF = ng - ny12 uses the nomenclature from Kelly, "A regularization approach to the reconciliation of constrained process data sets", *Computers & Chemical Engineering*, 1998. For example, if more raw measurements are removed, deleted or excluded from the data reconciliation problem due to the gross error detection, identification and elimination methodology, then the X2 threshold value will decrease since the number of regressed or unmeasured variables increases reducing the number of DOF. Furthermore, the maximum absolute or worst vetistic over all of the raw / measured variables and its original variable index are appropriately found in the valuation2 / variability2 and violation fields. Please consider that the total objective function vetistic, if greater than its 95% critical value, is the preferred and reliable indication that one or more gross errors are present and should always be considered when assessing the health and validity of the problem's or sub-problem's solution. Individual measured variables with a vetisitic greater than say from 3.0 to 4.0 may be suspected of being in gross error where the measured variable with the largest, worst or maximum absolute value may be considered as a prime suspect but others close to this vetistic value should also be considered as alternatives due to gross error confounding, confusion and correlation. The plus/minus (+/-) direction of the vetistic is also a good indication of whether the gross error's bias / offset either inflates (-ve vetistic value) or deflates (+ve vetistic value) the measurement.

Another term worth mentioning that is useful for data reconciliation and regression applications, is the well-known concept of "cross-validation" or cross-valuation via sample splitting. This is the technique of fitting, regressing, calibrating or training a model or sub-model using training (interpolation) data, then to assess, scrutinize, validate, vet or test the model by predicting the testing (extrapolation) data also known as out-of-sample data. Although IMPL does not perform cross-validation automatically, the user, modeler or analyst should consider as best practice, running cross-validation either manually or programmatically whenever a digital-twin or cyber-physical model is identified (structure, order, degree) and/or estimated (parameters, coefficients, biases).

Please be informed that for unit-operation-port-state (UOPS) flows, no validation (observability), variance and valuation information is available for the flow value. However, if the UOPS flow is measured and reconciled i.e., it has a soft bound target configured, then the redundancy validation, variance, vetistic, etc. are available for the 2-norm flow deviation variable only. The reason is due to the fact that internally IMPL does not explicitly create or generate UOPS flow variables as these are simply calculated as the sum of the all tee- or extreanal-stream transfer flows into the in-port-state or sum of all flows out of the out-port-state though if a target is specified, then explicit UOPS flow deviation variables are created and hence the reason these have the validations, vetistics, etc. information.

Summary List of OML Keywords with their Enumerations

In OML there are configuration- and content-related keywords which are parameterized and/or non-parameterized documented respectively below. Parameterized keywords have a left-hand-side and a right-hand-side demarcated by the equal sign ("=") special character similar to the IMPL settings or options found in the IMPL.set, IMPL.mem and the solver-specific settings files i.e., IMPL.solver. Configuration-related keywords may be likened to input / independent parameters which are usually parameterized and convey to OML more format and content specific details for the output files. Content-related keywords may be likened to output / dependent parameters which are usually non-parameterized and specify certain attributes, responses, results, values, etc. of the problem and its solution.

OBJROWS = -1, 0, 1

COLUMNS = -1, 0, 1

TCOLUMN = 0, 1

LOGFILEECHO = 0, 1

SKIPZEROS = 0D+0..INFIN

DATAFORMAT = 0, 1, 2, 3, 4, etc.

NUMBERFORMAT = E20.10E3 (for more options see Fortran's format specification syntax)

APPENDFILE = 0, 1

SKIPNAMES = 0, 1

CALCDATA = 0, 1 (SLICE, SPLICE, SCATTER, SEQUENT, SWAP, SWAP2, SUBSTITUTE, SKIP, SYNTHESIZE, SHED, SHIFT, STACK, STRETCH, SLIDE)

COMMENTECHO = 0, 1

PATHNAME = “” (quotes are optional)

TAGDELIMITER = “_” (default) or “-“, “;”, “:”, etc. (quotes are optional)

PROFITOBJ

PERFORMANCEOBJ1

PERFORMANCEOBJ2

PENALTYOBJ

TOTALOBJ

MAXVETISTIC, MAXVETISTICX

MAXVETISTIC = 1..6, MAXVETISTICX = 1..6

INCDIV

INCDIVTERM

SEEK

WORSTVARIABLE

WORSTCONSTRAINT

DATETIMENOW

DATENOW

TIMENOW

OBJVALUE

MIPCHKSUM

MIPCHKSUM0

LPITERS

MILPITERS

NLPITERS

ECLOSURE1, ECLOSURE1MAX, ECLOSURE1MAXI

ECLOSURE2, ECLOSURE2MAX, ECLOSURE2MAXI

ECLOSUREOO, ECLOSUREOOMAX, ECLOSUREOOMAXI

ICLOSURE1, ICLOSURE1MAX, ICLOSURE1MAXI

ICLOSURE2, ICLOSURE2MAX, ICLOSURE2MAXI

ICLOSUREOO, ICLOSUREOOMAX, ICLOSUREOOMAXI

STATUS, STATUSSIGNAL

SCROLL

SCROLLPREFIX

SELECTIVE

ALLCALC, ALLCALCS

ALLDATA, ALLDATAS

ALLX, ALLXS

ALLX / ALLXS = 1..7

ALLF, ALLFS

ALLF / ALLFS = 1..6