InfoWorks ICM

Binary Results Export

Version 3.0

# 

### August 2016



## Contents

[Introduction 3](#_Toc17977247)

[User Interface 3](#_Toc17977248)

[ICM Exchange 4](#_Toc17977249)

[File Format 5](#_Toc17977250)

[Data Types 5](#_Toc17977251)

[Dates 5](#_Toc17977252)

[Strings 6](#_Toc17977253)

[File Contents – Full time varying results export 6](#_Toc17977254)

[File Contents – Summary results export. 8](#_Toc17977255)

[Differences between the summary and full time varying results files 10](#_Toc17977256)

*Important: This document documents the initial format of the ICM binary results introduced in InfoWorks ICM version 2.5 and enhanced in version 6.5. Whilst Innovyze has no immediate plans to change this file format, it does not guarantee that the file format will not change in future releases of InfoWorks ICM or that future versions of the software will provide options to export this initial version. The file format is identified by a long integer at the beginning of the file.*

# Introduction

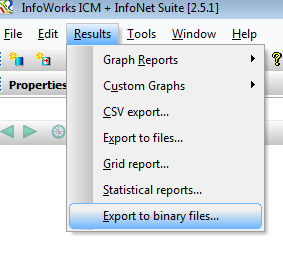
InfoWorks ICM, from the user interface, and InfoWorks ICM exchange, via its Ruby API, both provide a mechanism for exporting the results of simulations to binary files. The binary files are intended to be read by programs specially written by user organisations or by third parties to read them rather than off the shelf software such as spreadsheets, databases etc. The purpose of having a binary export file format is to decouple the internal files used by InfoWorks ICM to store its results, which are complex and hard to read, from files designed to be used by users of the software.

From version 6.5 ICM Exchange also provides a mechanism to export the summary results simulations. Despite this enhancement, the format of the ‘full’ binary results file is unchanged in version 6.5 from that introduced in version 2.5.

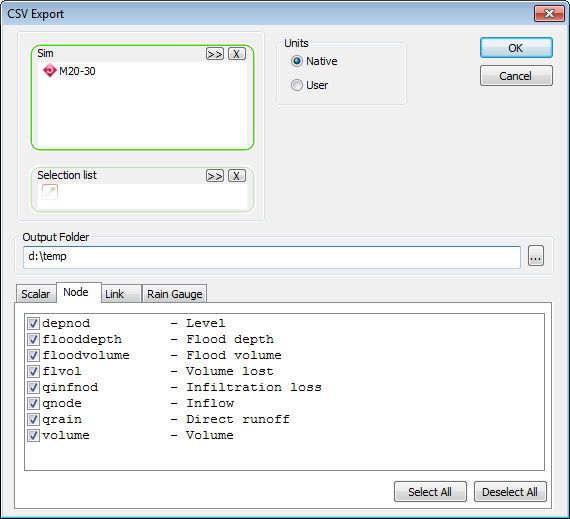
From version 7.5 ICM Exchange also provides a mechanism to export results for Risk Analysis Results objects and Risk Analysis Sim objects to binary files.

# User Interface

The binary results export option in InfoWorks ICM is invoked by a menu item ‘Export To Binary File’ on the results menu.



This brings up a dialog which is essentially a subset of the CSV Export. There are no options to export results based on results analysis objects. As there are no headers there are no options relating to headers.



it is possible to export the results for more than one simulation at once. One file will be exported for each simulation.

All results files are written to the chosen output folder.

## ICM Exchange

From ICM Exchange the full binary files are written using the *results\_binary\_export*  method of the WSSimObject class:

**results\_binary\_export(selection,attributes,file)**

The WSSimObject object can be a simulation or a results analysis results object. Although it is possible to have a WSSimObject which is a Results Analysis Sim object it is not possible to call this method for an object of this type as the Risk Analysis Sim object does not have multiple results for each attribute for each object, its results corresponding to the maximum results for a normal simulation.

For a Risk Analysis Results object the results are produced for each return period, with the return periods corresponding to the timesteps for ‘normal’ simulations.

Consult the ICM Exchange documentation for more details. Note that this is different from the user interface in that this method exports the results for *one* simulation, and that the parameter is the name of the individual file rather than the folder.

The summary (max/min) results files are written using the using the max\_*results\_binary\_export*  method of the WSSimObject class:

**max\_results\_binary\_export(selection,attributes,file)**

In this case the WSSimObject can be of the 3 object types:

* Sim (normal simulation)
* Risk analysis results
* Risk analysis sim

Consult the ICM Exchange documentation for more details.

# File Format

As described above, the files are binary files designed to be read by software written by 3rd parties and user organisations specifically for this task.

### Data Types

Values written out fall into the following five types:

1. Long integer (4 bytes)
2. Floating point value in IEEE format (4 bytes) – usually referred to as a ‘Float’ or a ‘Single’ depending on the language in which you are writing your program to read the file in.
3. Floating point value in IEEE format (8 bytes) – usually referred to as a ‘Double’ – these are only used in the summary results files.
4. Timestep - 8 bytes – see below..
5. String – variable length – see below.

### Dates

Dates are written as 8 byte doubles.

Recall that times in InfoWorks ICM can be ‘absolute’ or ‘relative’ – ‘absolute’ representing ‘real’ times, typically from data from real rainfall events, and ‘relative’ representing times typically from synthetic rainfall events.

If the value of the double is less than or equal to zero then it represents a relative time in seconds – so the number should be negated and then treated as a number of seconds.

If the value of the double is positive then it represents an absolute time in Microsoft’s ‘OLEDateTime’ format. In this format dates are represented as a number of days since midnight December 30, 1899.

e.g. the value -120 means a relative time of 120 seconds, whereas the value 40909.625 means 15:00 on the 1st March 2012.

### Strings

Strings are written in UTF8 encoding without Byte Order Marks prefixed with the number of bytes in the string as an unsigned character (1 byte). The strings are then padded so that they take up a multiple of 4 bytes **including the size prefix**. They are therefore followed by between 0 and 3 padding characters. The unsigned character 0 is used as the padding character. Note that the number of characters contained in the first byte for the string does not include the number of padding characters – this needs to be calculated.

The purpose of the padding is to help in cases where it is advantageous for a program reading the data to have the data values on 4 byte boundaries within the file i.e. to have every floating point number start at an offset of a multiple of 4 bytes from the beginning of the file.

e.g. the string OUTFALL would be represented as follows:

|  |  |
| --- | --- |
| Meaning | Value (byte) |
| Length | 7 |
| O | 79 |
| U | 85 |
| T | 84 |
| F | 70 |
| A | 65 |
| L | 76 |
| L | 76 |

Whereas the string OUTFALLS would be represented as follows:

|  |  |
| --- | --- |
| Meaning | Value (byte) |
| Length | 7 |
| O | 79 |
| U | 85 |
| T | 84 |
| F | 70 |
| A | 65 |
| L | 76 |
| L | 76 |
| S | 83 |
| Padding | 0 |
| Padding | 0 |
| Padding | 0 |

### File Contents – Full time varying results export

Throughout this description below, for risk analysis results objects the return periods stand in for the timesteps in ‘normal’ simulations e.g. rather than a value representing the number of timesteps as it would for a normal simulation, for risk analysis results objects is represents the number of return periods.

Each file contains the following:

1. A format indicator, currently a long containing the value 20110922 (4 bytes).
2. A long containing the number of (4 bytes)  
   Then for each timestep
   1. For each timestep, EITHER (for ‘normal’ simulations) the timestep as a double in the OLE date format (8 bytes) OR (for risk analysis results objects) the return period as a double (8 bytes)
3. A long containing the number of tables exported (4 bytes).
4. A long containing the number of words in block E [see below] (4 bytes) (*not blocks E and F as erroneously stated in previous versions of this document – i.e. the value has always been the number of words in block E)*
5. Then for each table with results to export:
   1. A long containing the number of objects exported for that table (4 bytes)
   2. A long containing the number of non-blob attributes exported for that table (4 bytes) – these are attributes where there is one value for each object.
   3. A long containing the number of blob attributes exported for that table (4 bytes) – these are attributes where there can be more than one value for each object.
   4. A string containing the table name
   5. A string containing the table description
   6. There then follows a block containing strings containing the information for the attributes and objects of that table as follows:
      1. For each non blob attribute:
         1. A string containing its internal name
         2. A string containing its description
         3. A string containing the units for that attributes (if any)
         4. A long integer containing the precision for the field (4 bytes).
      2. For each blob attribute the same as for the non-blob attributes.
      3. For each object in the table
         1. A string containing its ID
         2. For each blob attribute for this table a long value (4 bytes) indicating the number of values there are for this object for that attribute. This is necessary because in some cases the number of values varies by blob attribute as well as by object.
6. There then follows the results data. This is best explained by means of some pseudo-code:

For all timesteps in order

For all tables with results to export (in the same order as the header described above)

For all objects in that table (in the same order as the header described above)

For all non-blob attributes (in the same order as the header)

export value (float)

End

For all blob attributes (in the same order as the header)

For I = 1 To the number of values for this attribute for this object

Export value (float)

Next I

End

End

End

End

#### Notes

The value described in D above, the number of 4 byte words in blocks E and F may be used to skip past those blocks if, for example, you believe you know what is in them already. It is a number of ‘words’ i.e. the number of bytes in blocks E and F divided by 4.

The results begin at *offset 4 + (2 \* number of timesteps) + ‘number of words in block E’* 4 byte words from the start of the file.

The results for each timestep take up the same amount of space in the file, and each results attribute for each object (or attributes in the case of blobs) can be found at the same position in the results for each timestep relative to the start of that timestep. That amount of space and those offsets can be calculated when reading the header of the file.

Note that the number of results for a blob may be zero for a given attribute e.g. for river reaches without bank flows.

### File Contents – Summary results export.

Each file contains the following:

1. A format indicator, currently a long containing the value 20151009 (4 bytes).
2. A long containing the number of tables exported (4 bytes).
3. A long containing the number of words in block J [see below] (4 bytes).
4. Then for each table with results to export:
   1. A long containing the number of objects exported for that table (4 bytes)
   2. A long containing the number of non-blob attributes exported for that table (4 bytes) – these are attributes where there is one value for each object.
   3. A long containing the number of 4 byte floating point number (sometimes termed ‘single’) blob attributes exported for that table (4 bytes) – these are attributes where there can be more than one value for each object.
   4. A long containing the number of 8 byte floating point number (sometimes termed ‘double’) blob attributes exported for that table (8 bytes) – these are attributes where there can be more than one value for each object.
   5. A string containing the table name
   6. A string containing the table description
   7. There then follows a block containing strings containing the information for the attributes and objects of that table as follows:
      1. For each non blob attribute:
         1. A string containing its internal name
         2. A string containing its description
         3. A string containing the units for that attributes (if any)
         4. A long integer containing the precision for the field (4 bytes).
      2. For each 4 byte blob attribute the same as for the non-blob attributes.
      3. For each 8 byte blob attribute the same as for the non-blob attributes.
      4. For each object in the table
         1. A string containing its ID
         2. For each blob attribute for this table a long value (4 bytes) indicating the number of values there are for this object for that attribute. This is necessary because in some cases the number of values varies by blob attribute as well as by object.
5. There then follows the results data. This is best explained by means of some pseudocode:

For all tables with results to export (in the same order as the header described above)

For all objects in that table (in the same order as the header described above)

For all non-blob attributes (in the same order as the header)

export value (float)

End

For all 4 byte blob attributes (in the same order as the header)

For I = 1 To the number of values for this attribute for this object

Export value (float)

Next I

End

For all 8byte blob attributes (in the same order as the header)

For I = 1 To the number of values for this attribute for this object

Export value (double)

Next I

End

End

End

#### Notes

For the summary results an extra table is exported with the name ‘scalars’. This is always has exactly one object called ‘Scalars’. Its attributes are the single values reported for the whole simulation e.g. total lost, total outflow from outfalls, total rainfall.  
Because this ‘Scalars’ object is not a real network object, and therefore does not appear in any selection list, the table will not be exported if the export is done for a selection list.

The format indicators are the same for all 3 types of ICM object, the normal sim, the risk analysis results and the risk analysis sim. As described above the only significant difference is that the timesteps for normal sims are replaced by the return periods for the risk analysis results objects.

### Differences between the summary and full time varying results files

The differences between the two formats is as follows:

1. A different header value to distinguish the 2 files
2. The summary file does not have the number of timesteps or the values specifying the timesteps (section D above), instead there is just one set of summary values (primarily maxima and minima)
3. Each table has an additional single value for it in the header (J.d. above) the number of double precision floating point values i.e. the section for each table begins with 4 long values rather than 3.
4. The full time varying results do not contain any double values – another way of thinking of this is that the full time varying results can be considered as always having zero 8 byte blob attributes for each object type.