Export Block Descriptor

REFERENCE GUIDE

RG-0009-00 ENGLISH

Connecting Devices™

1 Export Block Descriptor

Exports are used to export block descriptor data from the eWON.
Export block can be used in the following situations:

- **Attach eWON data to an email**
- **Include eWON data into an eMail content**
- **Make an FTP PUT of eWON data from the eWON to a FTP server**
- **Make an FTP GET from a FTP client out of the eWON FTP server**
- **Include data in an eWON HTML custom page.**
- **Access data in Basic with OPEN "exp:....."**

In all these cases, an Export Block Descriptor will be used to describe the data to export.

1.1 Export block descriptor

An Export Block Descriptor is a string of characters describing the eWON data to export a predefined syntax.
Typically, the Export Block Descriptor will include information about:

- **Which data to export (Event log, Historical logging, etc.)?**
- **How to format the exported data (Binary, Text, Html table, Graphic)?**
- **Start time?**
- **End time?**
- **Which Tag is concerned?**
- **…?**

Example of Export Blocks <descriptor: `$dtHL $ftT $st_m10 $et_0 $tnMyTag $fnData.csv`

The export syntax is composed of a sequence of fields followed by its value. A field is a 3 characters identifier starting with $ and followed by 2 lower cap letters (case sensitive).

- **The first letter of the parameter value follows immediately the second letter of the field.**
- **The parameter is considered up to the first space found or until a $ or a [is detected.**
- **The parameter can also be placed between quotes ("). In that case the parameter value is the value between the quotes.**

The following fields are defined:

Fields	Description
$dt	Data type
$ft	Export format
$st	Start time
$et	End time
$tn	Tag name
$ut	Update last time
$ct	Compression type
$se	Script expression
$fl	Group filter
$fn	File name

Table 1: Export Block Descriptor fields description

2 Export fields syntax definition

The syntax for the different fields is defined in the following chapters.

2.1 $dt [Data Type]

The $dt field defines what data to export from the eWON.
The $dt parameter is made up of 2 upper case letters (case sensitive) that can take one of the following values:

$dt Parameter	Description	Binary	Graph	Text	Html
AH	Alarm history			T*	H
AR	Alarm Real time			T*	H
CF	Config	B*		T	
ES	estat file			T	H
EV	Event file			T*	H
FW	Firmware	B*			
HL	Historical Logging	B*	G	T	H
HT	Historical Table			T*	H
IV	Instant values	B*		T	
PG	Program			T*	
PP	PPP dump file	B			
RL	Real time logging	B*	G	T	H
SC	Communications configuration file			T	H
SE	Script Expression	B*		T	H
SS	Scheduled status			T*	H
SV	System Variable			T	
TL	Tag list			T*	H
UF	User file	B*		T	H
RE	Real Time diagnostic			T*	
TR	TAR File	B*			

Table 2: $dt parameters description

(*) The asterisk in the previous table denotes the default value of the $ft (export format). For example, for the DataType HL (Historical Logging), the default export format will be B (Binary) if you do not specify a $ft in your Export Bloc Descriptor (using [$dtHL] is equivalent to [$dtHL $ftB]).

2.2 $ft [Format]

The $ft field defines how to format the data exported. The following formats are available:

$ft Parameter	Format description
B	Binary
G	Graph
T	Text
H	HTML Table

Table 3: $ft parameters description

- **Binary: the data is sent in a raw binary format, not modified by the export module.**
- **Graph: the data is used to produce a PNG image representing a graph of the values (historical trend or real time graph).**
- **Text: The data is formatted as a CSV file, this means that each record is represented with each field on a line separated by a semicolon (;). The string fields are written between quotes, each line is ending by a CRLF (0x0D, 0x0A) sequence.**
- **Html: Instead of the text format, the data is placed in a simple HTML table. This format is useful for inserting data in the user custom HTML pages.**

2.3 $st [Start Time] and $et [End Time]

These 2 fields are used to limit the time range of an export operation. $st and $et provide the start and end time of the export. The parameter format is the same for both fields. There are 3 different formats for the $st, $et parameter:

- **Relative time**
- **Absolute time**
- **From last $ut (see also "$ut [Update Time]").**

2.3.1 $st, $et with relative time

Syntax: `$st_([s]|[m]|[h]|[d])100 _ = back`

(h,m,s,d = Hour, min, sec, day. 100 is the amount)
This represents a time regarding to the current time expressed in days, hour, minutes or seconds. If no letter is specified minutes are considered.

Examples:

$st_m10	10 minutes in the past
$et_0	0 minutes in the past (= now)
$st_d2	2 days in the past

Table 4: $st with relative time examples

2.3.2 $st, $et with absolute time

Syntax: `$stDDMMYYYY[[_HHMMSS][[_mmm][[_I][[_T]]]]]`

Where:

DDMMYYYY	Means Day, Month, Year, 8 characters. This parameter is required.
HHMMSS	Means Hour, Minute, Second, 6 characters. This parameter is optional (0 used by default)
mmm	Means milliseconds (000 to 999) 3 characters. This parameter is optional but if present, HHMMSS must also be specified.
I	Means intra sec counter. This value is present when receiving a historical logging from the eWON. It can be specified in export request to allow precise repositioning in the historical file. This parameter is optional, but if present, HHMMSS and mmm must also be specified.
T	Means Tag id. As for I, this parameter is used for precise positioning in historical file. This parameter is optional, but if specified, HHMMSS, mmm and I must be present also.

Table 5: $st parameters

When ALL the Tags are specified, the Tag values are exported in chronological order. For the same time there can be 2 Tag values. In order to reposition correctly in the file, it is necessary to provide the last Tag output during a previous export.

Examples:

$st01012000_120000	1 jan 2000 at 12 AM
$st01012000_120000_010	1 jan 2000 at 12 AM + 10 msec

Table 6: $st with absolute time examples

2.3.1 $st , $et with Last time

By adding the $ut command in an Export Block Descriptor, you can ask the eWON to remember the time of the last point exported, this time can be used for the next export.
The last time is reset when the eWON boots.

Syntax: `$stL`

Where "L" is the time parameter meaning last time.

2.4 $ut [Update Time]

This field has no parameter, it means that at the end of this export, the time of the last point exported must be saved in the eWON so that it can be used as a reference time for a later call.

Example: `$stL$et_0$ut`

This sequence will specify a time range from last time to current time AND will ask to update the last time at the end of the export.
The last time is stored on a per Tag basis if one Tag is specified for the export. A global last time can also be saved if "ALL Tag" is specified in an export.

2.5 $tn [Tag Name]

This field is used to specify a Tag name. It is required for graph commands. The parameter specified is the name of the Tag. When a $tn field can be specified for an export and no $tn is given, then the command is executed for ALL the Tags.

Example: `$tnMyTag` (MyTag is the name of the Tag)

2.6 $ct [compression format]

This field is only applicable when sending a file from the eWON to an FTP server, or as an attachment to an email.
The compression format is gzip (http://www.gzip.org). So that the unique argument to add after the field "$ct" is "G".

Example:

```
Putftp "test2.txt.gz","[$dtUF $ctG $uf/test.txt]"
```

Or:

```
SENDMAIL "destinator@provider.net", "", "Subject", "Mail body &[$dtUF $ctG
$uf/usr/test.txt $fntest2.txt.gz]"
```

Note:

If you give to the destination file the ".gz" extension only (and not ".txt.gz" for example), the destination file will be correctly exported, but in this case you will have to indicate the extension when uncompressing (".txt" in the above case).

You can then use a tool such as Winrar to extract the file; it will be extracted in a folder named "test2.txt".*

2.7 $se [Script Expression]

This field is only required for $dtSE export data. The $se parameter specifies the "script expression" to compute. Usually, the $se parameter will be inserted between quotes because if a $ is found in the expression it will be considered as the end of parameters.

Example: `$dtSE $se"A$"` (Exports the content of A$)

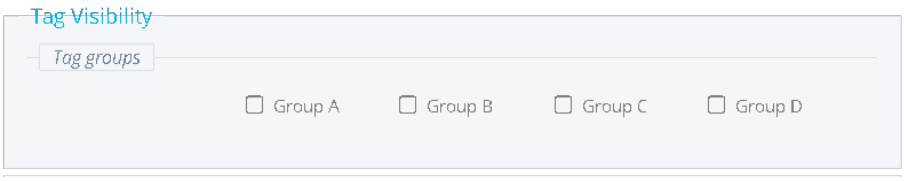
2.8 $fl [Group Filter]

The filter can be used like for the instant values ($dtIV), with an additional option 'X' The $fl can be any string containing A,B,C,D,X.

Example: `ACX or BDAX or X`

If no filter is specified then all the Tags with an enabled Historical logging are output. If filter include “X”, then tags without Historical logging enabled are also included, this is provided in case recording has been disabled but tags have been previously recorded in the file.
If filter include any of the A,B,C,D, then only the tags that belong to those groups are included in the output

Group filter can be added during the TAG creation:



2.9 $fn [File Name]

This field is used for specifying a file name to the export data (destination name). Usually this file name is used to specify the output of the data, for example when sending an attachment to an email. In this case, the $fn file name gives the name of the attachment: When doing a PUTFTP, then $fn does not need to be specified, because the PUTFTP command manages the name of the destination file.

Example: `PUTFTP "MyFileWithANewName.txt", "[$dtUF $uf/myfile.txt]"`

3 Data Types description and syntax

A Data type defines what is exported from the eWON. The data type is defined by the $dt field followed by 2 uppercase letters. The $dt field is mandatory for each "Export Block Descriptor" and usually the $ft (Format) field will also be present to define the output format of your data (although a default format is defined for each data type).
For each Data type, a set of other fields must be provided (some are mandatory and others are optional).

Note:

If you specify an unused field (neither mandatory nor optional), it will then be ignored.

This section will describe the syntax for each data type with the specific features for each of them.

3.1 $dtHL [Historical Logging]

3.1.1 Export content

The Historical logging outputs the data from the File system for ONE or ALL the fields. The output format can be TEXT, HTML Table or BINARY. The GRAPH format is also available IF only ONE Tag is specified.
A time range can also be specified for this export.

3.1.2 Detailed Example

```
$dtHL $ftT $st_h4 $et_m0 $tnA1
```

$dtHL	data type historical logging
$ftT	output format requested is CSV
$st_h4	start time is current time – 4 hours
$et_0	end time is current time – 0 minutes < > NOW
$tnA1	Tagname “A1” history to output

Table 7: $dtHL detailed example

3.1.3 Used Fields

Fields	Value if not specified
Mandatory	
$dt	
Optional	
$ft	Binary
$st	01/01/1970
$et	31/12/2030
$tn	All tags
$fl	All tags groups
$ut	No time update
$fn	Export block descriptor
$ct	Compression type

Table 8: $dt - used fields

3.1.4 Special parameters and fields

$st $et

If Last time is specified: ($stL or $etL): there is a last time logged for each Tag and a last time logged for all Tags.
If you specify a given Tag, its own last time will be used.
If a specific Tag is not requested, the export is performed for Tags concerned by historical logging and another last time memory is used.
If the output format is graph: $et_0 should be used instead of default value, otherwise the graph would span up to 31/12/2030.
For binary or text output, the default value can be kept.

$ft

Acceptable values			
Binary	**Text**	**HTML**	**Graph**

Table 9: [$dtHL] $ft acceptable values

The Graph format is only allowed if a Tag has been specified.
The Text format will output a comma-separated file. The separator is ';' to avoid any confusion with decimal point.
If all the Tags are output, they will be output in a chronological order in the file.

$ut

If only one Tag is specified, the time of the last point for that Tag will be memorized.
All the Tags can be output individually and last time is saved for each point.
Another memory is available if $ut is requested for ALL the Tags.

$tn

If this Tag is not specified, ALL the Tags will be selected for export. Otherwise, the Tag with the given name will be selected.

$fl

The group selection is only available with binary, text and HTML formats (not allowed for Graphic format $ftG).

3.1.4.1 examples

$dtHL	export all the Tags records in binary format
$dtHL $ftT	export all the Tags records in *($ftT)* Text format (like CSV file)
$dtHL $ftT $tnTemp	export all the values of the *($tnTemp)* Tag named "Temp" in *($ftT)* Text format
$dtHL $ftB $flAB	export all the values of *($flAB)* tags belonging to group A and B in *($ftB)* Binary format
$dtHL $ftT $tnTemp $st_h1 $et_s0	export the values *($st_h1)* from 1 hour to *($et_s0)* now of *($tnTemp)* Tag named "Temp" in *($ftT)* Text format
$dtHL $ftT $flCD $st_h1 $et_s0	export the values *($st_h1)* from 1 hour to *($et_s0)* now of *($flCB)* Tags belonging to group C and D in *($ftT)* Text format

Table 10: [$dtHL] examples

3.2 $dtHT [Historical Table]

3.2.1 Export content

The historical table is a representation of the IRCALL.BIN (incremental recording).
This representation provides a recordings representation as a table where columns are Tag names and rows are recording times.

3.2.2 Detailed Example

```
$dtHT $ftT $st_h4 $et_m0 $flAB $in10
```

$dtHT	data type historical table
$ftT	output format requested is text (CSV)
$st_h4	start time is current time – 4 hours
$et_0	end time is current time – 0 minutes < > NOW
$flAB	Filter to Instant value groups A and B
$in10	Interval fixed to 10 seconds

Table 11: $dtHT detailed example

3.2.3 Fields used

Fields	Value if not specified
Mandatory	
$dt	
Optional	
$ft	Text
$st	01/01/1970
$et	31/12/2030
$fl	All tags are displayed
$in	interval from the ircall.bin file

Table 12: $dt - used fields

3.2.4 Special parameters and fields

`$ft`

Acceptable values	
Text	**HTML**

Table 13: [$dtHT] $ft acceptable values

Text format will output a comma-separated file. The separator is ';' to avoid confusion with decimal point.

`$fl (filter)`

The filter can be used like for the instant values ($dtIV), with an additional option 'X'
The $fl can be any string containing A, B, C, D, X.

Example: ACX or BDAX or X

If no filter is specified, all the Tags with an enabled Historical logging are output.
If filter include "X", then tags without Historical logging enabled are also included, this is provided in case recording has been disabled but tags have been previously recorded in the file.
If filter include any of the A, B, C, D, then only the tags that belong to those groups are included in the output.

$in (interval)

The Historical Table time interval definition can follow 2 different standards:

- **Historical file defined interval**
- **Fixed interval**

For fixed interval, $in parameter must be used. The interval is defined in seconds.

Example: $in10 to output one value every 10 seconds

If $in is not specified, then the output time is defined by the time in the recording file.

Example:
Let's assume that we have 2 Tags logged with the following time and values (for clarity, the date has been omitted):

Time	Tag	Value
10:01:00	Tag1	1
10:10:00	Tag1	1.5
10:10:00	Tag2	1
10:11:00	Tag1	2
10:12:00	Tag1	3
10:21:00	Tag2	2
10:30:00	Tag1	4

A) If the fixed interval is not requested, then the following output will be produced

	Time	Tag1	Tag2
1	10:01:00	1	Undef
2	10:10:00	1.5	1
3	10:11:00	2	1
4	10:12:00	3	1
5	10:21:00	3	2
6	10:30:00	4	2

Notes:

At line 1, Tag2 is **Undef**, because no value are available in the log file.
At line 2, Tag1 and Tag2 are updated at on the same line, although there are 2 records in the incremental recording file, only 1 line is produced. So, except for the case when multiple Tags changed at the same time, when no interval is specified, the output contains one line for every record that has been logged.

B) If an interval of 10 minutes has been requested ($in600), then the following output would be produced.

	Time	Tag1	Tag2
1	10:01:00	1	Undef
2	10:11:00	2	1
3	10:21:00	3	2

Notes:

The output starts with the first time found in the file then it increases by 10 minutes.
There is no record with time equal (or higher) to 10:31, so the last line is 10:21

C) If an interval of 10 minutes is requested and the start time is 10:00, then the following output would be produced.

	Time	Tag1	Tag2
1	10:00:00	Undef	Undef
2	10:10:00	1.5	1
3	10:20:00	3	1
4	10:30:00	4	2

Notes:

On the first line, no values are available for Tag1 or Tag2 before 10:01:00 (for tag Tag1) in the recording file, so the values are **Undef**.

3.3 $dtRL [Real time Logging]

3.3.1 Export content

The Real-time logging outputs the data from the File system for ONE Tag.
The output format can be TEXT, HTML Table, BINARY or GRAPH.
A time range can also be specified for this export.

3.3.2 Detailed Example

```
$dtRL $ftG $st_m10 $et_m0 $tnA1
```

$dtRL	data type Real time logging
$ftG	output format requested is GRAPH
$st_m10	start time is current time – 10 minutes
$et_0	end time is current time – 0 minutes < > NOW
$tnA1	Tag log to output, A1
A1	Name of the Tag

Table 14: $dtRL - detailed example

3.3.3 Used Fields

Fields	Value if not specified
Mandatory	
$dt	
$tn	
Optional	
$ft	Binary
$st	01/01/1970
$et	31/12/2030
$ut	No time update
$fn	Export block descriptor
$ct	Compression type

Table 15: $dtRL - fields used

3.3.4 Special parameters and fields

`$st $et`

If the output format is "graph", $et_0 should be used instead of default value, otherwise the graph would span up to 31/12/2030.
For binary or text output, the default value can be kept.

`$ft`

Acceptable values			
Binary	**Text**	**HTML**	**Graph**

Table 16: [$dtRL] $ft - acceptable values

Text format will output a comma-separated file. The separator is ';' to avoid any confusion with the decimal point.

`$tn`

Note: The Tag MUST be specified for this export (Real Time logging enabled).

3.4 $dtAH [Alarm History]

3.4.1 Export content

The Alarm History outputs data from the File system for ONE or ALL the Tags. The output format can be TEXT or HTML Table.
A time range can also be specified for this export.

3.4.2 Detailed Example

```
$dtAH $ftH $st01012001
```

$dtAH	data type Alarm history logging
$ftH	output format requested is HTML table
$st01012001	1st of January 2001
$et	If not specified > until the end of file
$tn	If not specified > all the Tags

Table 17: $dtAH - detailed example

3.4.3 Fields used

Fields	Value if not specified
Mandatory	
$dt	
Optional	
$ft	Text
$st	01/01/1970
$et	31/12/2030
$tn	All
$ut	No time update
$fn	Export block descriptor
$ct	Compression type

Table 18: $dtAH - fields used

3.4.4 Special parameters and fields

$ft

Acceptable values	
Text	**HTML**

Table 19: [$dtAH] $ft - acceptable values

Text format will output a comma-separated file. The separator is ';' to avoid any confusion with the decimal point.
If all Tags are output they will be output in a chronological order in the file.

Line content of output file:

```
"EventDate";"TagName";"Status";"UserAck";"Description"
```

$tn

If this Tag is not specified, ALL the Tags will be selected for export. Otherwise, the Tag with the given name will be selected.

3.5 $dtAR [Alarm Real-time]

3.5.1 Export content

The Alarm Real-time outputs the Arlam Real-time data for ONE or ALL the Tags. The output format can be TEXT or HTML Table. If only ONE Tag is specified, 1 or 0 lines will be appended to the output header line (Time range is not applicable here).

3.5.2 Detailed Example

```
$dtAR $ftT
```

$dtAR	data type Alarm Real time
$ftT	output format requested is CSV
$tn	If not specified > all Tags

Table 20: $dtAR $ft - detailed example

3.5.3 Fields used

Fields	Value if not specified
Mandatory	
$dt	
Optional	
$ft	Text
$tn	All
$fn	Export block descriptor

Table 21: $dtAR - fields used

3.5.4 Special parameters and fields

$ft

Acceptable values	
Text	**HTML**

Table 22: [$dtAR] $ft -acceptable values

Text format will output a comma-separated file. The separator is ';' to avoid confusion with the decimal point.

If all the Tags are output, they will be output in a chronological order in the file.

Line content of output file:

```
"TagId";"AlarmTime";"TagName";"AlStatus";"AlType";"StatusTime";"UserAck";"Description"
```

$tn

If this field is not specified, ALL the Tags will be selected for export. Otherwise, the Tag with the given name will be selected.

3.6 $dtEV [EVent file]

3.6.1 Export content

The Event file outputs data from the File system. The output format can be TEXT or HTML Table.
A time range can also be specified for this export.

3.6.2 Detailed Example

```
$dtEV $ftT $st_m30
```

$dtEV	data type events logging
$ftT	output format requested is CSV
$st_m30	last 30 minutes
$et	If not specified > until now

Table 23: $dtEV - detailed example

It will output a CSV file containing the events during the last 30 minutes.

3.6.3 Fields used

Fields	Value if not specified
Mandatory	
$dt	
Optional	
$ft	Text
$st	01/01/1970
$et	31/12/2030 < > NOW
$fn	Export block descriptor
$ct	Compression type

Table 24: $dtEV - fields used

3.6.4 Special parameters and fields

`$ft`

Acceptable values	
Text	**HTML**

Table 25: [$dtEV] $ft - acceptable values

Text format will output a comma-separated file. The separator is ';' to avoid any confusion with the decimal point.
Line content of output file:

```
"EventTimeInt";"EventTimeStr";"Event"
```

EventTimeInt	Time provided as an integer (number of seconds since 1/1/1970)
EventTimeStr	Date and time as text

Table 26: EventTime types

3.7 $dtSS [Scheduled Status]

3.7.1 Export content

The scheduled actions are actions that are executed in a scheduled manner, for example: PutFTP, Send Mail, Send SMS.
When one of these actions is requested, it does not occur immediately, but it is queued for a sequential execution. This export allows checking the content of this queue and giving the status of all the actions in queue: "in progress", "executed (success)" and "executed with error".

3.7.2 Detailed Example

```
$dtSS
```

3.7.3 Fields used

Fields	Value if not specified
Mandatory	
$dt	
Optional	
$ft	Text
$fn	Export block descriptor

Table 27: $dtSS - fields used

3.7.4 Special parameters and fields

$ft

Acceptable values	
Text	**HTML**

Table 28: [$dtSS] $ft - acceptable values

Text format will output a comma-separated file. The separator is ';' to avoid any confusion with the decimal point.

Line content of output file:

```
"ActionId","ActionType","StatusCode","StatusText","Start","End"
```

3.8 $dtSE [Script Expression]

3.8.1 Export content

This export provides a means to get the content of a script expression. The script expression is a standard eWON Basic-like expression returning a STRING, and INTEGER or a FLOAT. The evaluation of the expression will always occur between 2 scripts execution, for example between 2 ONTIMER executions, or between 2 cycles of the cyclic sections.

3.8.2 Detailed Example

```
$dtSE $se"A$"
```

3.8.3 Fields used

Fields	Value if not specified
Mandatory	
$dt	
$se	
Optional	
$ft	Binary
$fn	Export block descriptor

Table 29: $dtSE - fields used

3.8.4 Special parameters and fields

$ft

Acceptable values		
Text	**HTML**	**Binary**

Table 30: [$dtSE] $ft - acceptable values

Binary and Text format means that the output is the content of the Script Expression itself.
HTML output supposes that the content of the script expression is a comma-separated data (string between quotes, items separated by ';' and end of lines marked with CRLF (0x0d, 0x0a)).

Then the exported output is an HTML table containing these data.

$se

Defines the script expression to output, usually this expression is typed between quotes because $ characters are considered as separator otherwise.

3.9 $dtUF [User File]

3.9.1 Export content

The User File export returns the content of a file in the User File area (/usr/ directory – or subdirectory).

When the file is exported, the <%#ParamSSI> and <%#TagSSI> Tags are replaced by the actual values.

3.9.2 Detailed Example

```
$dtUF $uf/ufdir/uf1.txt
```

$dtUF	User file
$uf/ufdir/uf1.txt	Will export the *uf1.txt* file located in the */usr/ufdir* directory

3.9.3 Used Fields

Fields	Value if not specified
Mandatory	
$dt	Must always precede "UF": dtUF (means Data Type= User File)
$uf	
Optional	
$fn	
$ft	Binary
$ct	no compression
$fl	

Table 31: $dtUF - used fields

3.9.4 $uf [User File Name]

This field is the name of the user file that you want to export (source name).
The file name can be preceded by the name of the subdirectory inside the /usr directory:

`/myfile.txt`
(myfile.txt is in the /usr directory)

Note: The first "/" is optional.

`/mydir/myfile.txt`
(myfile.txt is in the /usr/mydir directory)

Note: The first "/" is optional.

The complete path can also be specified:

`/usr/myfile.txt`
(myfile.txt is in the /usr directory)

Note: The first "/" is optional.

`/usr/mydir/myfile.txt`
(myfile.txt is in the /usr/mydir subdirectory)

Note: The first "/" is optional.

Example:

```
Putftp "/test.txt","[$dtUF $uf/myfile.txt]"
```

3.9.4.1 $fn [Destination File Name]

This field is used for specifying a file name to the export data (destination name). Usually this file name is used to specify the output of the data, for example when sending an attachment to an email. In this case, the $fn file name gives the name of the attachment:

```
SENDMAIL "MailReceiver@YourMail.com", "", "Mail Subject", "&[$dtUF $uf/myfile.txt $fnNewName.txt]"
```

The above example will attach to an eMail a file named “NewName.txt” that is a copy of the file “/usr/myfile.txt”.

There is also one special use of the $fn: when a user file ($dtUF$fn) is exported and you do not specify the *source name ($uf)*; in that case, the $fn parameter is used as source and as destination file name.

Using only $fn in a send mail string:

```
SENDMAIL "MailReceiver@YourMail.com", "", "Mail Subject", "Mail text &[$dtUF$fnmyfile.txt]"
```

The above syntax will attach a file with its name (and not with the EBD syntax as name).

When doing a PUTFTP, then $fn does not need to be specified, because the PUTFTP command manages the name of the destination file:

```
PUTFTP "MyFileWithANewName.txt", "[$dtUF $uf/myfile.txt]"
```

3.9.5 Special parameters and fields

- **`$ftB` "File Type" binary (default). Other types are unavailable**
- **`$ctG` "Compression Type" GZ.**
- **`$flNOSSI` "Disable SSI parsing in $dtUF".**

  When the $dtUF export bloc descriptor is used to export a user file then eWON will parse the user file during export for any SSI tag (tags starting with <%#). In some cases, this behavior is not wanted (in case the file may contain the <%# sequence but no SSI are used).

  The $flNOSSI can be used to disable SSI parsing in $dtUF
  IMPORTANT: NOSSI must be entered in caps (case sensitive)

  Example: `$dtUF $uf/usr/MyFile.bin $flNOSSI $fnOutFile.bin`

3.10 $dtIV [Instant Values]

3.10.1 Instant value - general information

Instant value means values of Tags at a current time. The file of instant value contains for each Tag the following information:

TagId	Id of the Tag
TagName	Name of the Tag (in text mode)
Value	Current value of the Tag
AlStatus	Current alarm status of the Tag
AlType	Type of the current alarm

Table 32: $dtIV - instant value file's informations

- **The file containing the instant values for every Tag is available in binary or text format; you can download the instant values file directly from the root of the eWON root file list, or you can address it by using an Export Block Descriptor.**
- **The instant values file normally contains all the Tags, but there is an additional feature that allows obtaining only the instant values from specific Tags.**

3.10.1.1 Alarm status code values

The below table lists the different values that the field ***AlStatus*** can have, depending on the Alarm State and of the action the user has performed on it:

Alarm Status	Alarm Status Value	Alarm status explanations
NONE	0	Tag is not in alarm status
PRETRIGGER	1	Tag is in pretrigger alarm status**Warning**: we assume there is no alarm if AlStatus value <= Alarm Pretrigger
ALM	2	Tag's alarm status is active
ACK	3	Tag's alarm has been acknowledged
RTN	4	Tag's alarm returns from an active status

Table 33: inst_val.txt file - alarm status code values

3.10.1.2 Alarm type values

The table below lists the different values that the field ***AlType*** can have, depending on the type of threshold that has been stepped over by the Tag value, depending on the configuration set in the Tag's configuration page:

Alarm Type	Alarm Type Value	Alarm type explanations
NONE	**0**	The Tag value is inside of the limits beyond of which the alarm is triggered
HIGH	**1**	The Tag value exceeds the value entered in the ***Alarm Level High*** field from the Tag configuration page
LOW	**2**	The Tag value is less than the value entered in the ***Alarm Level Low*** field from the Tag configuration page
LEVEL	**3**	The Tag value matches the ***Boolean Alarm Level*** value defined in the Tag configuration page
HIGH_HIGH	**4**	The Tag value exceeds the value entered in the ***Alarm Level HighHigh*** field from the Tag configuration page
LOW_LOW	**5**	The Tag value is less than the value entered in the ***Alarm Level LowLow*** field from the Tag configuration page

Table 34: inst_val.txt file - alarm type values

3.10.2 Writing Instant Values to the eWON

The instant values file can also be written to the eWON. The file must be written by FTP to the ftp root folder and must be written to the file. All the Tags value present in the file will be used to change the corresponding Tag in the eWON. If a Tag is not found, then it will be ignored.

- **Writing in binary format:**
    - **The file format must comply exactly with the definition (see below) and all Tags are identified by their Tag ID.**
- **Writing in text format:**
    - **When writing the instant value in text format, there are different possibilities to address the Tag:**
    - **If a "TagName" column is present, then the Tags will be accessed by their name (even if a "TagId" column is present)**

Example:

```
"TagId";"TagName";"Value";"AlStatus";"AlType"
1;"M1";10.000000;0;0
2;"M2";20.000000;0;0
```

If a "TagName" column is NOT present, the Tags will be accessed by their id:

```
"TagId";"Value";"AlStatus";"AlType"
1;10.000000;0;0
2;20.000000;0;0
```

WARNING:

Remember that the Tag Id is not an index, but a unique number that has been allocated to the Tag when created, and cannot be reused unless the configuration is erased and a new configuration is created.

3.10.3 Binary file format

The file starts with a Header that can be represented by the following C structure:

```
struct InstantValueHeader
	{
		int	Rev;
		int	RecSize;	//Record size
		int	NbTag;  //Number of Tags exported
		int	RecFlag;	//Reserve (must be set to 0) int	Reserved2;
	}
```

Then there is a record number for each Tag (the record number can be obtained from the header (NbTag):

```
struct InstantValueRecord
	{
		int	TagId;
		float Value;
		int	AlStatus;
		int	AlType;
		int	Reserved;
	}
```

3.10.4 Export content

The $dtIV Tag exports either the entire content of the Instant Value file (txt or binary format) or only a part of it, depending on the parameters that might have been defined with the $fl field.

3.10.5 Detailed examples

$dtIV $flAB	Will export all the Tags belonging to group A or B
$dtIV $flA	Will export all the Tags belonging to group A
$dtIV $fl	Will export no Tag (useless)
$dtIV $flABCD	Will export all the Tags belonging to group A or B or C or D (but missing Tags that belong to no group)
$dtIV	Will export all the Tags regardless of group definition

Table 35: $dtIV - detailed examples

3.10.6 Fields used

Fields	Value if not specified
Mandatory	
$dt	
Optional	
$fl	
$ft	Text

Table 36: $dtIV - used fields

3.10.6.1 $fl [Group or Groups]

The $fl (for filter) field must be directly followed by a list of one or more groups A, B, C or D (that have been checked in the Tag's configuration). There must be no other character in the filter and all the groups must be in uppercase.

Example: `$dtIV $flAB`

It will export all the Tags belonging to group A or B.

3.11 $dtSV [System Variable]

3.11.1 Export content

$dtSV returns the value of a defined eWON system variable. A typical use is when the user wants to include the eWON online IP address in an email by using the sendmail Basic syntax. The output format can only be of TEXT type.

3.11.2 Detailed Example

```
sendmail "user@user.be","","Ip","The eWON online IP'address is:
[$dtSV$seOnlineIpAddr]"
```

$dtSV	Data type system variable
$se	Will export a system expression
OnlineIpAddr	The current eWON online IP address (ie. 192.168.10.15)

Table 37: $dtSV - detailed example

Will include the eWON online IP address in the body from a sent eMail.

3.11.3 Used Fields

Fields	Value if not specified
Mandatory	
$se	System expression. At this time, only "OnlineIpAddress" is available

Table 38: $dtSV - fields used

3.12 $dtPP [Dump PPP]

3.12.1 Export content

$dtPP exports the dump.ppp file (binary format): The output format can only be of BINARY type.

3.12.2 Detailed Example

```
sendmail "user@user.be","","eWON PPP dump","&[$dtPP$fndump.ppp]"
```

$dtPP	Data type PPP dump
$fn	Will give the required name to the file

Table 39: $dtPP - detailed example

Will attach the eWON PPP dump file to an eMail.

3.12.3 Used Fields

Fields	Value if not specified
Optional	
$fn	File name

Table 40: $dtPP - used fields

3.13 $dtES [Export Estat]

3.13.1 Export content

$dtES exports the estat.htm file : the file that lists the current status from the main eWON features. The output format can be TEXT or HTML.

3.13.2 Detailed Example

```
sendmail "user@user.be","","eWON estat file","&[$dtES$ftH$fnestat.htm]"
```

$dtES	Data type estat file
$ftH	Will export the file in htm format
$fn	Will give to the file the required name

Table 41: $dtES - detailed example

Will attach the eWON estat.htm file to the eMail.

3.13.3 Fields used

Fields	Value if not specified
Optional	
$ft	File type
$fn	File name

Table 42: $dtES - fields used

3.14 $dtSC [Export COM Config]

3.14.1 Export content

$dtSC exports the communications configuration file (comcfg.txt): The output format can be TEXT or HTML.

3.14.2 Detailed Example

```
sendmail "user@user.be","","eWON COM config file","&[$dtSC$ftH$fncomcfg.htm]"
```

$dtSC	Data type COM config file
$ftH	Will export the file in htm format
$fn	Will give to the file the required name

Table 43: $dtSC - detailed example

It will attach the eWON comcfg.htm file to the eMail.

3.14.3 Used Fields

Fields	Value if not specified
Optional	
$ft	File type
$fn	File name

Table 44: $dtES - used fields

3.15 $dtRE [Real Time Diagnostic]

3.15.1 Export content

$dtRE exports the Real Time Diagnostic data (equivalent to the real-time log)
The output format can be TEXT only.

3.15.2 Detailed Example

```
sendmail "user@user.be","","eWON Real Time Log","&[$dtRE$fndiag.txt]"
```

$dtRE	Data type : Real Time Diagnostic
$fn	Will give to the file the required name

Table 45: $dtRE - detailed example

It will attach to an email the file "diag.txt" holding the real-time diagnostic of the eWON.

3.15.3 Used Fields

Fields	Value if not specified
Optional	
$ft	File type (only **T** available)
$fn	File name

Table 46: $dtRE - used fields

3.16 $dtTR [TAR file]

3.16.1Export content

$dtTR exports the eWON file(s) inside a TAR formated file.
The data to include in the TAR file can be defined using a single file list, a directory and wildcard '*', or/and another export block descriptor's.

3.16.2Detailed example

```
$dtTR $fnmytar.tar $td{/usr/*}
```

$dtTR	Data type : TAR file
$td	Data : {/usr/*} the complete /usr directory
$fn	mytar.tar

Table 47: $dtTR - detailed example

Put the complete /usr directory in the mytar.tar file.

3.16.3Used Fields

Fields	Value if not specified
Mandatory	
$fn	File name
$td	Data
Optional	
$ft	**B**
$ct	

Table 48: $dtTR - used fields

3.16.3.1 $fn [output filename]

$fn is used to define a name for the output file.

Example: $fnMyDataFile.tar will produce a TAR file with the name "MyDataFile.tar".

3.16.3.2 $td [TAR data]

The data consists in a list of items separated by ',' (comma).
The items are specified between "{ }" (curly brackets).

```
$td {item1},{item2},...,{itemX}
```

Each item is one of the following:

- **A /usr file name (complete path to file)**
- **A /usr directory name (complete path to directory) followed by ***
- **An export block descriptor**

If the path represents a directory followed by * then the whole tree is exported.

3.16.3.3 TAR format and eTAR modified format

The TAR file produced by the eWON could be:

- **a standard TAR file, compliant to the USTAR (Uniform Standard Tape Archive) format.**
- **a modified TAR file, called *eTAR*.**

Standard TAR file can be opened by most of Packager Program like Winzip, WinRar.

Due to technical reasons, the eWON produces an eTar format when the package holds file(s) belonging to the eWON root directory.
This eTAR file is viewed as a "corrupted file" by Packager Program. But, you can use our eTar.exe tools to reformat this eTAR as a valid TAR file.

You can find this eTar.exe program on http://cdn.ewon.biz/software/divers/etar.zip.

Examples

```
$dtTR $fnmytar.tar $td{/usr/file1.txt}
```
Will make a TAR file named "mytar.tar" containing the file /usr/file1.txt

```
$dtTR $fnmytar.tar $td{/usr/MyFile1.txt},{/usr/MyFile2.txt}
```
Will make a TAR file named "mytar.tar" containing the files /usr/MyFile1.txt and /usr/MyFile2.txt

```
$dtTR $fnmytar.tar $td{$dtCF $ftT $fnMyConfig.txt}
```
Will make an eTAR file named "mytar.tar" containing the eWON configuration file named "MyConfig.txt"

```
$dtTR $fnmytar.tar $td{/usr/file1.txt},{$dtCF $ftT $fnMyConfig.txt}
```
Will make an eTAR file named "mytar.tar" containing the eWON configuration file named "MyConfig.txt" and the file /usr/file1.txt

```
$dtTR $fnmytar.tar $td{/usr/*}
```
Will make a TAR file named "mytar.tar" containing all the /usr directory

```
$dtTR $fnmytar.tar.gz $ctG $td{/usr/*}
```
Will make a *compressed* TAR file named "mytar.tar.gz" containing all the /usr directory

```
$dtTR $fnmytar.tar $td{/usr/*},{$dtPG $fnprogram.bas},{$dtCF $ftT
$fnconfig.txt},{$dtSC $ftT $fncomcfg.txt}
```

Will make an eTAR file named "mytar.tar" containing all the /usr directory,
the program file named "program.bas", the configuration file named "config.txt"
and the communication configuration file named "comcfg.txt"

```
putftp "Test_TAR.tar","[$dtTR $td{/usr/Page1.shtm},{/usr/Page2.shtm}]"
```

Will put by FTP the file "Test_TAR.tar" containing the files Page1.shtm and Page2.shtm.
Note that for FTP action, the filename is the first parameter of the PutFTP instruction, then the $fn parameter is not required in the TAR command.

Note:

It is forbidden to include an item that described a TAR format itself. THE TAR IS NOT RECURSIVE!

Forbidden example: $dtTR $td{ $TR …….. }

3.17 Additional exports available

$dtTL	Tag List
$dtPG	Program
$dtCF	Configuration File

Table 49: additional exports available

These are all the files from the eWON configuration. They are equivalent to the file available through the eWON FTP server.