Spirit it
A MEMBER OF THE ABB GROUP

Flow-X®

Web Services

Reference Manual

Product	**Flow-X** Web-service References
Reference number	01-0110-1
Revision	A.9
Date	April 2013
Authors	H.F.J. Rutjes

Disclaimer

Spirit IT has taken care in the preparation of this book, but makes no expressed or implied warranty of any kind and assumes no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the issue of the information or programs contained herein.

Special note

Printed in the Netherlands.

Visit Spirit on the Web: http://www.SpiritIT.com

Spirit Innovative Technologies
Prof. dr. Dorgelolaan 20
5613 AM Eindhoven
The Netherlands

Table of contents

List of figures

List of tables

(This page is left blank intentionally.)

Chapter 1 - Introduction

This document describes the web-services supported by the flow computer. This document is intended for internal use and third parties who wish to interface with the flow computer through its web-server.

(This page is left blank intentionally.)

Chapter 2 - Web server

HTTP response headers

The Flow-X web server offers additional response headers which may be of use to automated systems interacting with the flow computer[1].

Field	Explanation
X-FlowX-Instance	This header contains identifying information about a flow computer, in that it can be used to detect when a flow computer has last been restarted (i.e. when it has last been (soft) rebooted). The layout of this field is a JSON object, which contains the last-started time of the web server, as well as a randomly generated instance ID which is different for every boot: `{ "start time" : "2003-01-01T00:00:00Z", "id" : "0123456789ABCDEF0123456789ABCDEF01234567" }`

[1] *This functionality has been introduced in Flow-Xpress 1.7.4.*

Chapter 3 - Web services

The following web-services are supported by the flow computer. This can be considered the flow computer web namespace:

Web-service	Description
/ackalarms	Acknowledges specific or all alarms.
/alarms	Returns the alarms.
/archive	Returns historical archive data. (see also: /snapshots)
/archives	Returns historical archive information.
/dashboard	Returns dashboard information.
/deviceinfo	Returns device info information.
/framework	Returns the framework information for the User Interface.
/languages	Returns the languages and translations.
/logs	Returns the event-log.
/navigation	Returns the navigation structure for the User Interface.
/perfcounters	Returns system performance counter information.
/print	Prints a report.
/report	Returns a report.
/reports	Returns list of reports.
/security	Handles login/logout.
/snapshots	Returns historical archive data.
/tags	Returns the tags.
/units	Returns the units & enumerations.
/writetags	Writes one or more tag values.

Table 3-1: Web-services overview

AckAlarms

This web-service allows for acknowledging all or specific alarms.

Request

Acknowledges alarms can be done by specifying "/ackalarms" in the URL. The following arguments can be specified in the URL:

Argument	Description
Userkey	Required Userkey. The user must have sufficient privileges (high enough security level) for acknowledging alarms).
IDFilter	Optional filter of alarm-ID's to acknowledge. When omitted, all alarms are acknowledged. Multiple ID's can be separated using a comma-character. A range of ID's can be specified using the dash-character (e.g. "1,2,3-5,80-100").

Table 3-2: AckAlarms request arguments

Example #1

Request: /ackalarms?userkey=267AB6

This request acknowledges all alarms.

Example #2

Request: /ackalarms?userkey=267AB6&idfilter=6

This request acknowledges alarm with ID 6.

Example #3

Request: /ackalarms?userkey=267AB6&idfilter=6,7,8

This request acknowledges alarm with ID 6, 7 and 8.

Example #4

Request: /ackalarms?userkey=267AB6&idfilter=100-200

This request acknowledges all alarms in the range 100 to 200.

Alarms

This interface returns current alarm-data from the web-server. It can be used to show an alarm-list on the flow computer display. Additionally it may be used in the future to expose alarm-data to a supervisory computer or an OPC Alarm & Events server.

Request

Reading alarms can be done by specifying "/alarms" in the URL. The following arguments can be specified in the URL:

Argument	Description
Max	Optional number containing the maximum number of items to return. Set this to the maximum number of items you can display on the screen. When omitted all items are returned.
Offset	Optional number containing the offset of items to return. For instance if the display can show only 10 items at a time and the third page is being shown, the offset should be set to 20. When omitted an offset of 0 is used.
Filter	Optional string containing a filter expression (e.g. "ACKED AND NOT(SUPPRESSED)"). When omitted no filtering is performed and all alarms are returned.
SortOrder	Optional string containing the sort-order (e.g. "TIMESTAMP DESC, NAME, ACTIVE"). When omitted the alarms are returned according to the internal ordering.
Fields	Optional integer mask which specifies which fields to return. When omitted, the ID, name, text and state are returned. Instead of an integer mask it is also possible to specify the string "all" which returns all fields. 0x00000001 ID (unique ID as a number). 0x00000002 Name (Name of the tag as configured in the application). 0x00000004 Text (Text in the requested language). 0x00000008 Multi-lingual text key (e.g. "alarm.pt1") 0x00000010 Description (Description in the requested language). 0x00000020 Multi-lingual description key (e.g. "alarmdesc.pt1"). 0x00000040 Type (1=Status, 2=LoLo, 3=Lo, 4=Hi, 5=HiHi, 6=ROC, 7=Deviation). 0x00000080 Priority (number). 0x00000100 Timestamp when the state changed 0x00000200 Active-timestamp (timestamp when alarm became active). 0x00000400 Inactive-timestamp (timestamp when alarm became in-active). 0x00000800 Acked-timestamp (timestamp when alarm was acknowledged). 0x00001000 Block-count (number). 0x00002000 Limit (Alarm limit). 0x00004000 Deadband (Alarm deadband). 0x00008000 Delay (Delay in seconds before changing the alarm state). 0x00010000 Value. 0x00020000 State bitmask (1=Active, 2=Acked, 4=Suppressed, 8=Blocked, 16=Disabled). 0x00020000 Alarm group Id 0x00040000 Id of the tag which is the source of the alarm 0x00080000 Node on which the alarm lives (panel=0, module1 =1, etc..)
Language	Optional ID of the requested language (number). When omitted the default language will be used.
CacheID	Optional Cache ID (string) of last received data-set. When omitted a fresh data-set is returned.
Userkey	This is the user key of the user currently logged on. The server is aware of its user level and therefore will only return allowed data.

Table 3-3: Alarms request arguments

The following keywords can be used in the filter- and sort-order arguments:

Keyword	Type	Description
ACTIVE	Boolean	True when alarm is currently active.
ACKED	Boolean	True when alarm has been acknowledged.
BLOCKED	Boolean	True when alarm has toggled between active/inactive state too much
SUPPRESSED	Boolean	True when alarm is being suppressed.
DISABLED	Boolean	True when alarm has been completely disabled.
DELAYED	Boolean	True when alarm is in delayed state.
ID	Integer	Unique ID of the alarm.
NAME	String	Name of the alarm (e.g. “PT6”).
TEXT	String	Alias in requested language (e.g. “Druk Opnemer 6”).
DESC	String	Description in requested language.
TYPE	Integer	Type (1=Status, 2=LoLo, 3=Lo, 4=Hi, 5=HiHi).
PRIORITY	Integer	Priority.
TIMESTAMP	DateTime	Timestamp at which the alarm state changed (Format: dd-mm-yyyy hh:mm:ss)
ACTIVETIME	DateTime	Timestamp at which alarm became active (Format: dd-mm-yyyy hh:mm:ss).
INACTIVETIME	DateTime	Timestamp at which alarm became in-active (Format: dd-mm-yyyy hh:mm:ss).
ACKEDTIME	DateTime	Timestamp at which alarm was acknowledged (Format: dd-mm-yyyy hh:mm:ss).
BLOCKCOUNT	Integer	Number of times the alarm has been blocked.
LIMIT	Float	Limit.
DEADBAND	Float	Deadband.
DELAY	Integer	Delay in seconds before the alarm state changes.
VALUE	Float	Current value.
STATE	Integer	State bitmask (1=Active, 2=Acked, 4=Suppressed, 8=Blocked, 16=Disabled).
TAGID	Integer	Id of the tag which is the source of the alarm.
NODE	Integer	Node on which the alarm lives (panel=0, module1 =1, etc..).

Table 3-4: Alarms filter- and sort-order keywords

Response

The response from the server is an XML-stream of the following layout:

```
<alarms totalcount="23" cacheid="1" >
  <alarm
    id="1" tagid="2" node="1" name="PT6" text="Druk opnemer 6"
    desc="Some description" type="3" priority="10"
    timestamp="01-01-2007 00:00:00" activetime="01-01-2007 00:00:00"
    inactivetime="01-01-2007 00:00:00" ackedtime="01-01-2007 00:00:00"
    blockcount="0" limit="102.5" deadband="1.6"
```

```
    delay="10" value="201.89" state="9"
  />
  ...
</alarms>
```

Figure 3-1: Alarms XML response

Only those fields are returned that were specified in the request.

Caching

The cacheid can be used to request new alarm data only when needed. When the cacheid is omitted, a fresh result is returned. The device keeps track of all changes to the alarms. When an alarm changes, the cacheid on the device is updated. When a request is made and the cacheid on the device is the same, then the following response is returned:

```
<alarms cacheid="1" />
```

Figure 3-2: Alarms XML response (same cacheid)

When the client receives the same cacheid as it sent to the device, it should use its last full result instead.

Archive

The archive web-service can be used to retrieve historical data. Historical data is stored in so-called archives and for each archive snapshots are created, either periodically or on batch-ends.

- To retrieve information about the contents of snapshots, see the /archives web service.
- When using the SQLite Data Base Storage System, snapshot data may also be retrieved through the /snapshots web service. This web service offers
- he /archive web service has been superseded by the /snapshots service, which offers

Request

Reading historical archives can be done by specifying "/archive" in the URL. The following arguments can be specified in the URL:

Argument	Description
Archive	Name of the archive for which the snapshots should be returned (e.g. "mod1_Hourly_Run").
Name	Same as 'Archive'. This argument was introduced in revision 1.4 for consistency reasons.
Start	Optional start-date/time for which snapshots should be returned. When the DateTimeFormat argument is specified, the specified date/time should be in that same format.
End	Optional start-date/time for which snapshots should be returned. When the DateTimeFormat argument is specified, the specified date/time should be in that same format.
Count	Optional argument (number) that specifies the maximum number of archives to return. It is preferred that this value is set to a reasonable value (e.g. max 32). When the device contains lots and lots of snapshots, the snapshots should be obtained through a series of requests, rather than a single big request. The maximum number of snapshots that can be requested is 100.
Skip	Optional argument (number) which specifies the number of snapshots to skip, given the search-direction. This argument can be used in conjunction with the Count-argument to for instance, read all the snapshots using multiple requests.
Ascending	Optional argument (0/1) that specifies the search-direction. When omitted, the result is returned in Descending order. Descending means, most recent first.
CacheID	Optional argument which can be used to request only the newly created snapshots since the last request.
Fields	Optional integer mask which specifies which fields to return for XML-output. When omitted only RecordCount and CacheID are returned. This field is ignored when the type is CSV or TXT. 0x00000001 Skip (Number of snapshots to skip that were specified in the request, see Skip-argument) 0x00000002 MaxCount (Maximum number of snapshots that were specified in the request, see Count-argument). 0x00000004 Nr (Number of snapshots that have been created for the archive). 0x00000008 Count (Number of snapshots that have been returned by this request). 0x00000010 Ascending (Sort-order that was specified in the request, see Ascending-argument).

	0x00000020 CacheID (Unique number that changes whenever a snapshot is added for the archive). 0x00000040 Changed (Returns '0' or '1' depending on whether a new archive snapshot was added, since the specified CacheID).
Resultmode	Optional string which specifies in what form snapshots are returned. When omitted the "listidonly" mode below is chosen by default, unless "start" or "end" option are provided then the "full" option is chosen. full Returns the full data for a snapshot. This is the same as the old behavior, before this option was introduced. unconverted Return the full data for a snapshot, but does not convert tag names to human readable strings. listidsonly Returns only an ID for each snapshot. countonly Returns only the number of snapshots.
Type	Optional type which specifies in which format the result is returned (e.g. XML, CSV, etc.). When omitted the result-data is returned as XML. xml Returns the result as xml. xmlstream Same as "xml", except that when requested from a browser, "xmlstream" causes the result to be opened inside the browser rather than showing the "Save As" dialog. csv Returns the result as a comma-separated-value file. csvstream Same as "csv", except that when requested from a browser, "csvstream" causes the result to be opened inside the browser rather than showing the "Save As" dialog. txt Returns the result as plain text. txtstream Same as "txt", except that when requested from a browser, "txtstream" causes the result to be opened inside the browser rather than showing the "Save As" dialog. json Returns the result as a JSON encoded object. jsonstream Same as "json", except that when requested from a browser, "jsonstream" causes the result to be opened inside the browser rather than showing the "Save As" dialog.
Language	Optional ID of the requested language (number). When omitted the default language will be used.
DateTimeFormat	Optional date/time format used for formatting date/time fields in the output. When omitted the default format of "YYYY/MM/DD hh:mm:ss" is used.
ShortTexts	Optional argument (0/1) that specifies whether tag-names should be returned as fully qualified names (e.g. "Meter Temperature 1") or with their short-names (e.g. "Meter Temperature").

Table 3-5: Archive request arguments

Example #1

Request: /archive?archive=mod1_Hourly_Run&count=18&type=csv

This request returns the 18 most recent archived snapshots for archive "mod1_hourly_run" in CSV-format.

Example #2

Request: /archive?name=mod1_Hourly_Run&count=18&skip=36

This request returns snapshots 37..55 for archive “mod1_hourly_run” in the default XML-format.

Example #3

Request:
/archive?archive=mod1_Daily_Run&count=1&end=12/03/2011&datetimeformat=dd/mm/yyyy

This request returns the most recent ‘Daily-Run’ snapshot which was created before “12/03/2011”.

Example #4

Request:
/archive?archive=mod1_Daily_Run&start=1/02/2011&end=1/03/2011&datetimeformat=dd/mm/yyyy

This request returns all ‘Daily-Run’ snapshot which was created in the month February.

Response

The response from the server is a stream of the following layout (example).

XML

```
<archive name="mod1_Hourly_Run" nr="2887" cacheid="1010" >
  <snapshot timestamp="26/04/2012 01:59:55" >
    <item name="Forward hourly gross volume prev" value="0.0" unit="m3"  />
    <item name="Forward hourly mass prev"         value="0.0" unit="kg"  />
    <item name="Forward hourly base volume prev"  value="0.0" unit="sm3" />
    <item name="Forward hourly energy prev"       value="0.0" unit="GJ"  />
    ...
  </snapshot>
  ...
</archive>
```

Figure 3-3: Archive XML response

TXT (Plain text)

```
Archive: mod1 Hourly Run
Snapshot: 26/04/2012 01:59:55
Forward hourly gross volume prev: 0.0 m3
Forward hourly mass prev: 0.0 kg
...
```

```
Snapshot: 26/04/2012 00:59:53
Forward hourly gross volume prev: 0.0 m3
Forward hourly mass prev: 0.0 kg
...
```

Figure 3-4: Archive TXT response

CSV (Comma separated value)

```
Archive,mod1_Hourly_Run
Snapshot,26/04/2012 01:59:55
Forward hourly gross volume prev,0.0,m3
Forward hourly mass prev,0.0,kg
...

Snapshot,26/04/2012 00:59:53
Forward hourly gross volume prev,0.0,m3
Forward hourly mass prev,0.0,kg
...
```

Figure 3-5: Archive CSV response

Only those fields are returned that were specified in the request.

Archives

The archives web service offers information regarding the archives that are currently configured on the flow computer.

Request

Reading historical archives can be done by specifying "/archives" in the URL. The archives webservice does not accept any additional arguments.

Response

The response from the server is an XML-stream of the following layout (see Figure 3-6). This lists the archives currently configured on the flow computer. For each archive, the tags that will be recorded in a snapshot are given.

```
<?xml version="1.0" encoding="utf-8" ?>
<archives>
  <archive name="mod1 BatchRun">
    <item name="mod1_LM_Run!BATCH_NR_PRV" />
    <item name="mod1_LM_Run!BATCH_ID_PRV" />
    ...
  </archive>
  ...
</archives>
```

Figure 3-6: List of archives configured on a flow computer

Dashboard

The dashboard web-service is used to obtain basic Web UI information such as alarm indicator status, seal status, and the cacheid of the navigation tree. These components were integrated into a single web-service so that the Web-UI periodically only needs to call the service once.

Request

Obtaining the dashboard info can be done by specifying "/dashboard" in the URL. The following arguments can be specified in the URL:

Argument	Description
Userkey	Optional Userkey. When specified, the Userkey is checked for validity. When not valid an optional entry is returned in the response indicating that the Userkey is not valid: <security invaliduserkey="1">.
TimeFormat	Optional time-format. When omitted, the time is returned using the "h:mm" format.

Table 3-6: Dashboard request arguments

Response

The response from the server is an XML-stream of the following layout (example):

```
<dashboard signature="01/03/2010 11:19:55.860">
  <time val="10:24" />
  <device name="FQI-400" />
  <alarmindicator color="green" blink="0"
     activecount="5" unackedcount="2" activeunackedcount="6" />
  <seal locked="0" />
  <navigation cacheid="707" />
</dashboard>
```

Figure 3-7: Dashboard XML response

DeviceInfo

The deviceinfo web-service can be used to obtain basic status information of the flow computer. This web-service is typically used to check whether the flow computer is alive and what application/version it is running.

Request

Obtaining the deviceinfo can be done by specifying "/deviceinfo" in the URL. No additional arguments exist for this web-service.

Response

The response from the server is an XML-stream of the following layout (example):

```
<deviceinfo signature="17/09/2015 11:12:06.872">
  <version val="1.9.0.6254" brand="Spirit"/>
  <type val="FlowX-P2"/>
  <brand val="Spirit"/>
  <platform val="spirit1"/>
  <application_name val="Gas_Metric.xls"/>
  <application_version val="1.2.3.0"/>
  <application datetime val="17/09/2015 10:57:16"/>
  <application type val="FlowX-P1"/>
  <application signature val="-264023790"/>
  <device name val="FQI400"/>
  <locked val="0"/>
  <enclosure val="3"/>
  <paramsignature val="197883635"/>
  <spreadsheetsignature val="-421235412"/>
  <memusage val="56685100"/>

  <memavail val="39018496"/>
  <diskusage val="9.49"/>
  <cpu       totaltime="26.72"       kerneltime="0.00"       usertime="26.72"
curprocid="2933426862"/>
  <datetime val="30/09/2015 16:02:24"/>
  <uptime days="13" hours="4" minutes="51" seconds="55"/>
  <modules current="0">
    <module                          id="0"                          status="0"
ident="FWVer=1.9.0.6254&BusSetup=3&AppType=FlowX-
P2&AppVer=1.2.3.0&AppName=Gas Metric.xls&AppSignature=-
264023790&FWChecksum=0&FirstIP=10.0.101.240" platform="spirit1"/>
    <module                          id="1"                          status="0"
ident="FWVer=1.9.0.6254&BusSetup=3&AppType=FlowX-
P2&AppVer=1.2.3.0&AppName=Gas Metric.xls&AppSignature=-
264023790&FWChecksum=0&FirstIP=10.0.101.240" platform="spirit1"/>
  </modules>
</deviceinfo>
```

Figure 3-8: DeviceInfo XML response

Framework

The framework returns an XML structure which contains information for displaying the Web-UI. Includes is, supported languages, common language texts, alarm-states.

Request

The framework can be obtained by specifying the "/framework in the URL. The framework web-service is actually not a real web-service but merely a reference to files on the flow computer. For each languages that is configured a framework xml-file exists. To obtain such a file append '/<languageid>.xml' to the URL. For instance, to obtain the framework data for language 1 use: "/framework/1.xml".

Response

The response from the server is an XML-stream of the following layout:

```
<framework
  labeluser="User"
  labelpassword="Password"
  labellogout="Logout"
  labellogin="Login"
  labelcurrent="Current"
  labelnew="New"
  labelrestartreq="The device needs to be restarted in order for the change
to take effect"
  datetimeformat="dd/mm/yy hh:mm:ss"
  clocktimeformat="hh:mm"
  autologofftimeout="3600"
  homelocation="/"
  alarmlocation="/207/"
  alarmacklevel="1000"
  seallockevel="1000"
  alarmledvisible="1"
  reportprintlevel="500"
  reportsavelevel="500"
  loginrequired="0"
  direction="ltr"
  title="Flow-X/P (Panel) - Gas Metric.xls"
  displaydevicename="1"
  displayfulldatetimeonweb="0" >
  <languages count="3" >
    <language id="1" image="images/flags/english.png" name="English" />
    <language id="2" image="images/flags/dutch.png" name="Nederlands" />
    <language id="3" image="images/flags/chinese.png" name="ㄎㄉㄑ" />
    ...
  </languages>
  <alarmstates count="3" >
    <state   mask="16"   value="16"   forecolor="#000000"   backcolor="#808080"
blink="0" />
    <state   mask="9"   value="8"   forecolor="#FFFFFF"   backcolor="#00C000"
blink="0" />
    <state   mask="3"   value="1"   forecolor="#000000"   backcolor="#FF0000"
blink="1" />
     ...
  </alarmstates>
```

```
  <texts>
    <text name="user" value="Username" />
    <text name="password" value="password" />
    <text name="ackall" value="Ack All" />
    ...
  </texts>
</framework>
```

Figure 3-9: Framework XML response

Languages

The languages web-service can be used to obtain the supported languages and/or the language texts for each language.

Request

Obtaining the language data can be done by specifying "/languages" in the URL. The following arguments can be specified in the URL:

Argument	Description
Type	Type of data to request. When omitted, all language data is returned, including the languages and the texts. 1 Returns both the languages and the texts. 2 Returns the languages only. 3 Returns the texts only.
Language	ID of the requested language (number). When omitted, all language data is returned.

Table 3-7: Languages request arguments

Response

The response from the server is an XML-stream of the following layout (example):

Type = 1

```
<multilingual>
  <languages>
    <language id="1" name="English" />
    <language id="2" name="Nederlands" />
    …
  </languages>
  <texts>
    <text key="tag.gv_inc" L1="Gross vol increment" />
    <text key="tag.comp2_c2cur" L1="Ethane in use" L2="Ethaan in gebruik" />
    …
  </texts>
</multilingual>
```

Figure 3-10: Language XML response (Type 1)

Type = 2

```
<languages>
  <language id="1" name="English" />
  <language id="2" name="Русский" />
  …
</languages>
```

Figure 3-11: Language XML response (Type 2)

Type = 3

```
<texts>
  <text key="tag.gv_inc" L1="Gross vol increment" />
  <text key="tag.comp2 c2cur" L1="Ethane in use" L2="Ethaan in gebruik" />
  …
</texts>
```

Figure 3-12: Language XML response (Type 3)

Note that only those texts are returned that exist. For instance, if a language is defined (2 = Dutch) and a translation exists for a certain text, then an attribute "L2" is returned containing the translation.

Logs

The logs web-service returns the event-log.

Request

Reading log-entries can be done by specifying "/logs" in the URL. The following arguments can be specified in the URL:

Argument	Description
Max	Optional number containing the maximum number of items to return. When omitted, returns all items matching the query.
ChronDir	Indicates direction of search; if set, search is performed in chronological direction (from old towards more recent items). If set to "0" or omitted, search is performed in anti-chronological direction (from recent towards older items).
Time	Optional string containing timespan specification to match log-entries against. Only log-entries within the timespan will be returned, separate from content-filtering (described hereafter). The string may be of the form MOMENT_FROM|MOMENT_TO (e.g. "01-01-2006 00:00:00 | 31-12-2007 23:59:59"), in which case only log-entries within [MOMENT_FROM..MOMENT_TO) (that is, including MOMENT_FROM, but excluding MOMENT_TO) will be considered. The string may also be of the form MOMENT, in which case MOMENT|NOW is implicitly used (NOW being the current time). When omitted, timespan START|END is used, i.e. all log-entries will be considered.
Filter	Optional string containing a filter expression (e.g. "(LOC LIKE "drivers.modbus.*") AND (SEV > INFO) AND (TEXT LIKE "*bereik*")"). When omitted, all log-entries are considered.
Module	Optional string containing a filter expression for filtering on specific modules (e.g. "(MODULE = 1) AND (MODULE > 3)"). When omitted, all modules are queried.
Fields	Optional Integer mask which specifies the fields to return. When omitted only the timestamp and text are returned: 0x0001 Timestamp. 0x0002 Text (in the requested language). 0x0004 Location (string). 0x0008 Module ID (number). 0x0010 Log tags (snapshot of tag values when entry was created).
Language	Optional ID of the requested language (number). When omitted the default language is used.
Iterator	String indicating the result-window (startitem..enditem) of the last search-operation; to the user, this is an 'anonymous' string, resulting from a previous search operation. A search operation continues from the position indicated by the specified iterator, if present; if missing, search is started at either the most recent or oldest item in the log, according to the 'ChronDir' field.
Userkey	This is the user key of the user currently logged on. The server is aware of its user level and therefore will only return allowed data.

Table 3-8: Logs request arguments

The following keywords can be used in the filter-argument:

Keyword	Type	Description
LOC	String	Location-string from the actual log-entry.
SEV	Integer	Severity-indicator; see following table for more information.
TEXT	String	Translated text, according to the Language-field in the request.

Table 3-9: Logs filter keywords

The following keywords can be used in the module-argument:

Keyword	Type	Description
MODULE	Integer	ID of the module.

Table 3-10: Logs module keywords

The following severity constants can be used in combination with the “SEV” keyword:

Constant	Description
DEBUG	Severity: debug-messages (lowest priority).
INFO	Severity: informational messages.
WARNING	Severity: warnings.
ERROR	Severity: errors.
FATAL	Severity: fatal errors (highest priority).

Table 3-4: Logs severity constants

The severity-constants are listed in ascending order, and have operators ‘<’ and ‘>’, ‘<=’ and ‘>=’ defined.

Example #1

/logs?Time=01-07-2010 16:00:00|01-08-2010 00:00:00

Returns all events of the specified period, e.g.

```
<log time="01/07/10 16:11:02" text="Flow rate hi alm changed from Normal to Alarm" />
<log time="01/07/10 16:11:02" text="Parameter Flow rate hi limit was changed from 1000000 to -1 by a (5000)" />
<log time="01/07/10 16:10:50" text="Parameter Station totals and rates was changed from Disabled to Enabled by a (5000)" />
<log time="01/07/10 16:10:45" text="User a (Touchscreen) has logged in" />
```

Example #2

/logs?Filter=text like “parameter*

Returns all events which start with the word ‘parameter’.

```
<log time="01/07/10 16:11:02" text="Parameter Flow rate hi limit was changed from 1000000 to -1 by a (5000)" />
<log time="01/07/10 16:10:50" text="Parameter Station totals and rates was changed from Disabled to Enabled by a (5000)" />
```

Example #3

/logs?Time=01-01-2013 01:00:00|01-02-2013 00:00:00&Filter=loc like "parchanged"

Returns all parameter changes of January 2013

Response

The response from the server is an XML-stream of the following layout:

```
<logs

iterator="vTIC5gAwAAAAABAAAAAAAAADgBAAAAAAA/////////8HAAAAAAAAAAEAAAB2AAAA
AAAAAHYAAAABQA"
  endreached="1" >
  <log
    time="01-01-2007 00:00:00"
    loc="err.drivers.modbus.startup"
    text="Timeout bij initialiseren modbus instantie #4."
    mod="1"
  />
  ...
</logs>
```

Figure 3-13: Logs XML response

The ‘EndReached’ flag (either ‘0’ or ‘1’) indicates the last search hit the end of the log (corresponding to the specified search direction). Only those fields are returned that were specified in the request.

Navigation

The navigation web-service returns the whole navigation tree of the application depending on the user-level and language.

Request

Obtaining the navigation structure can be done by specifying "/navigation" in the URL. The following arguments can be specified in the URL:

Argument	Description
Fields	Optional integer mask which specifies the special fields to return. When omitted, no special fields are returned. 0x0000 No special fields returned 0x0001 Condition (also nodes which condition evaluates false are returned)
Userkey	This is the user key of the user currently logged on. The server is aware of its user level and therefore will only return the pages applicable for the current user. When omitted, the navigation tree is returned that is visible to not logged in users.
Language	Optional ID of the requested language (number). When omitted the default language is used. The specific language-texts are returned in the "text" attributes as seen below in the XML stream.

Table 3-11: Navigation request arguments

Response

The response from the server is an XML-stream of the following layout (example):

```
<navigation cacheid="56" >

  <folderpage
    id="1" name="home"
    text="Thuis" location="Thuis" icon="page" />

  <alarmpage
    id="2" name="alms"
    text="Alarmem" location="Alarmem" icon="alarms"
    filter="( ACTIVE OR NOT(ACKED) ) AND NOT(DISABLED) AND NOT(SUPPRESSED)"
    sortorder="STATE, ACTIVETIME DESC, TEXT" />

  <tagpage
    id="4" name="str1"
    text="Stroom 1" location="Location" icon="tags"
    shorttexts="1" tags="1,2,3,4,5" count="5" />

  <reportpage
    id="5" name="rep"
    text="Report 1" location="Report 1" icon="reports"
```

```
    shorttexts="1" mod="-1" type="-1:" />

  <logpage
    id="6" name="log"
    text="Logs" location="Logs" icon="logs"
    filter="1" />

</navigation>
```

Figure 3-14: Navigation XML response

Note: icon will always be at "<web-server>/data/icons/<icon>.png".

Folder-pages are hierarchical which means that any page-type can be placed under a folderpage. The following example shows a hierarchy of pages. Attributes have been omitted for clarity of the example:

```
<navigation cacheid="56" >
  <folderpage id="1" name="page1" />
  <folderpage id="2" name="page2" />
  <folderpage id="3" name="nested page" >
    <tagpage id="4" name="tagpage" />
    <folderpage id="5" name="nested page 2" >
      <logpage id="6" name="logpage" />
    </folderpage>
  </folderpage>
</navigation>
```

Figure 3-15: Navigation-hierarchy XML response

Caching

The cacheid indicates the number of times the navigation tree has been changed. The navigation tree is dependent on parameters and therefore when a parameter is changed, the navigation tree may change as well.

The "/dashboard" web-service can be used to check whether the navigation-tree has changed. It returns the cacheid of the navigation tree, and when that returned cacheid differs from the cached, then the navigation tree must be reloaded. See the Dashboard web-service for further reference.

Perfcounters

This web service provides system specific diagnostic data, such as detailed cycle-time statistics.

Request

Getting the performance counters can be done by specifying "/perfcounters" in the URL. The performance counters are in fact a tree of subsystems, and each sub-system has its own performance counters. When plain "/perfcounters" is specified on the URL, the counters of all sub-systems are returned. By specifying the name of the sub-system (e.g. "/perfcounters/comm") it is possible to drill down to only that sub-system.

The following arguments can be specified in the URL:

Argument	Description
Counter	Optional name of the counter(s) to filter on. The filter can contain wildcards '*' and '?'.

Table 3-12: Perfcounters request arguments

Example #1

/perfcounters

Returns all performance counters.

Example #2

/perfcounters/spreadsheet

Returns all performance counters for the spreadsheet sub-system.

Example #3

/perfcounters?counter=*lastcycletime

Returns all performance counters which end with the name "lastcycletime" for all sub-systems.

Response

The response from the server is an XML-stream of the following layout:

```
<counters n="">
  <counters n="system">
    <c n="MaxCycleTime"  v="1.3959"   u="ms"/>
    <c n="DiskUsage"     v="6.98"     u="%" />
    <c n="LastCycleTime" v="133.5266" u="ms"/>
    ...
  </counters>
  ...
</counters>
```

Figure 3-16: Perfcounters XML response

Print

The print web-service gives access to printing reports on the flow computer.

Request

Reading reports can be done by specifying "/print" in the URL. The following arguments can be specified in the URL:

Argument	Description
Report	Instance-name of the archived-report (e.g. "1:Run_Daily-1.20100623.fxr"). The instance-name of a report can be obtained through the "/reports" web-service. The instance-name consists of the module-id and the file-name o f the report (<mod>:<file>).
Printer	Optional name of the printer to be used for printing. When omitted, the default printer that was specified for the report is used.
Userkey	Required Userkey of the logged on user.

Table 3-13: Print request arguments

Response

No XML-stream is returned for this request.

Report

The report web-service returns a report from the device.

Request

Reading a manual report can be done by specifying "/report" in the URL. The following arguments can be specified in the URL:

Argument	Description
Type	Format-type to return the report in (default "fxr"). fxr — Returns the report in the secured fxr-format. txt — Returns the report as plain text. txtstream — Same as "txt", except that when requested from a browser, "txtstream" causes the report to be opened inside the browser rather than showing the "Save As" dialog. xml — Returns the report as xml. xmlstream — Same as "xml", except that when requested from a browser, "xmlstream" causes the report to be opened inside the browser rather than showing the "Save As" dialog.
Report	Instance-name of the archived-report (e.g. "1:Run_Daily-1.20100623.fxr"). The instance-name of a report can be obtained through the "/reports" web-service. The instance-name consists of the module-id and the file-name of the report (<mod>:<file>).

Table 3-14: Report request arguments

Example #1

Request: /report?type=txt&report=0:Snapshot-0.20100701160000.fxr

This request returns the corresponding report of module 0 as plain text

Response

The response is either a .FXR file, a plain text stream (UTF-8) or an xml stream.

Reports

This reports web-service allows for getting a list of the existing reports on the device.

Request

Reading a manual report can be done by specifying "/reports" in the URL. The following arguments can be specified in the URL:

Argument	Description
Filter	Optional filter which specifies which reports to return. *Syntax: [[<mod>:]<filter>]* mod — ID of the module for which reports are requested (when omitted, reports from all modules are returned). filter — Filter expression which can contain wildcards '*' and '?'. The filter is compared against the report file-name (e.g. "*Daily*", "Run_Daily*").
Count	Optional maximum number of reports to return (default 20). The number of reports returned by a single request is limited to 100 reports.
Skip	Optional parameter to specify the position of the returned list of reports in the list of existing reports. This must be a string representing the date and time of the first or last report in the returned list. By default, the current time is used. OBSOLETE: when this parameter is not a date/time string, but the integer number, the given number of reports will be skipped. This feature is extremely slow on large lists of reports, and will be removed in the future.
Before	Optional parameter to specify the meaning of the time specified by "skip", default is 0 0: reports created AFTER that time will be returned 1: reports created BEFORE that time will be returned
DateTimeFormat	Optional date/time format to use for the reports (default "YYYY/MM/DD hh:mm:ss").
Language	Optional ID of the requested language (number). When omitted the default language is used.

Table 3-15: Reports request arguments

Example #1

Request: /reports?filter=*Daily*

This request retrieves all reports with the word "Daily" it its name, e.g.

```
<report time="2010/07/01 16:00:00" file="Daily-0.20100701160000.fxr" mod="0" />
<report time="2010/07/01 16:00:00" file="Daily-0.20100701160000.fxr" mod="0" />
<report time="2010/07/01 15:00:00" file="Daily-0.20100701150000.fxr" mod="0" />
<report time="2010/07/01 15:00:00" file="Daily-0.20100701150000.fxr" mod="0" />
<report time="2010/07/01 14:00:00" file="Daily-0.20100701140000.fxr" mod="0" />
```

Example #2

Request: /reports?count=20&skip=now

This request retrieves 20 last reports. It is equivalent to simply /reports.

Example #3

Request: /reports?skip=2011/08/12 14:46:59&before=0&count=5

This request retrieves 5 oldest reports generated after 2011/08/12 14:46:59.

Response

The response from the server is am XML- stream of the following layout (example):

```
<reports oldest="2011/08/11 16:54:08" latest="2011/08/17 12:32:42" first="2011/08/12
14:46:49" last="2011/08/12 14:46:59" skip="2011/08/12 14:46:59" before="1" count="5">
<report time="2011/08/12 14:46:59" file="Snapshot-1.20110812144659.fxr" mod="1"
text="Snapshot"/>
<report time="2011/08/12 14:46:56" file="Snapshot-1.20110812144656.fxr" mod="1"
text="Snapshot"/>
<report time="2011/08/12 14:46:54" file="Snapshot-1.20110812144654.fxr" mod="1"
text="Snapshot"/>
<report time="2011/08/12 14:46:52" file="Snapshot-1.20110812144652.fxr" mod="1"
text="Snapshot"/>
<report time="2011/08/12 14:46:49" file="Snapshot-1.20110812144649.fxr" mod="1"
text="Snapshot"/>
</reports>
```

Figure 3-17: Reports XML response

The following values are returned:

Value	Description
Oldest	Oldest existing report matching the filter.
Latest	Latest existing report matching the filter.
First	First (by time; last by position) report in the returned list. Not available when the list is empty.
Last	Last (by time; first by position) report in the returned list. Not available when the list is empty.
Skip	Repeats the request parameter “skip”
Before	Repeats the request parameter “before”
Count	Number of returned reports
Time	Time when the report was generated
File	File id to be used to retrieve or print the report
Mod	Id of the module where the report is stored
Text	Display text (depends on the selected language)

Table 3-16: Reports response values

Security

The Security web-service handles login and logout operations. After a user logs in a Userkey is returned which can be used in requests to other web-services.

Request

Authentication can be done by specifying "/security" in the URL. The following arguments can be specified in the URL:

Argument	Description	
Action	Action to perform:	
	"login"	Authenticates a login request by verifying a user-name and password and returning a user- key and –level.
	"logout"	Informs the server to invalidate the given user-key and returns information about the logged-out status.
	"verify"	Verifies that the given user-key is valid. If it is no longer valid, the reason is returned.
	"autologin"	Attempts to automatically login (no username or password required). This is only possible when this feature is supported.
	"downloadconfig"	Requests for the security configuration file. User-key might be required.
	"uploadconfig"	Upload a new security configuration file to the server. The new configuration file is directly applied. A valid user-key is required.
Username	This is the username of the user (required when action="login").	
Password	This is the password of the user (required when action="login").	
Userkey	Existing user-key currently in use by the web-application. This argument is required when action="logout" or "verify".	

Table 3-17: Security request arguments

Response

The response from the server is an XML-stream of the following layout:

Action = Login/Autologin

```
<user
  authenticated="1"
  userlevel="1000"
  userkey="uazeicOAczUA"
  username="Operator"
  message="User has successfully logged in" />
```

Figure 3-18: XML response when login is successful

Or

```
<user
  authenticated="0"
  userlevel="0"
  username="Guest"
  message="Access denied (invalid username/password)" />
```

Figure 3-19: XML response when login has failed

Action = Logout

```
<user
  authenticated="0"
  userlevel="0"
  username="Guest" />
```

Figure 3-20: XML response when logging off

Action = Verify

```
<user
  authenticated="1"
  userlevel="30"
  username="Admin" />
```

Figure 3-21: XML response when userkey was successfully verified

Or

```
<user
  authenticated="0"
  userlevel="0"
  username="Guest"
  message="Invalid userkey specified (it may have expired)" />
```

Figure 3-22: XML response when verification of userkey failed

Snapshots

The snapshots web-service can be used to retrieve historical data and is a replacement for the obsolete “Archives” web-service. The main difference is another way of addressing snapshots: snapshots are addressed by their unique identifiers (strings) rather than by their timestamps. This is more convenient for the automatic downloading snapshots from the flow computer, albeit less human-readable. The advantage is that it is known exactly which snapshots have been retrieved, and which are not. This allows for automatic retrieval of snapshots without the risk of either duplicate or missing snapshots.

Note: This web-service does not support the on-the-fly translation of tag names. I.e. this web service returns snapshot data using tag names, rather than the tag descriptions. Should the client need such a translation, it must translate tag names after downloading snapshots.

Request

Reading historical archives can be done by specifying “/snapshots” in the URL. The following optional arguments can be specified in the query string:

Argument	Description
archive	String specifying the name of the archive for which the snapshots should be returned (e.g. “mod1_Hourly_Run”). • When omitted, snapshots of all archives will be returned. • When the archive cannot be found, the web server will return a “HTTP 404 not found” response.
ascending	Number specifying the search-direction. Must be either ‘1’ or ‘0’. • When ‘1’, the result is returned in ascending order (oldest first). • When ‘0’, the result is returned in descending order (most recent first). • When the given value cannot be converted to either a ‘1’ or a ‘0’, the web server will return a “HTTP 400 Bad Request” response. The default value for this parameter is ‘1’.
count	Number specifying the maximum number of snapshots to return per request. • This value must be between 0 and 100, inclusive. • If the value is less than 0, or greater than 100, the web server will return a “HTTP 400 Bad Request” response. The default value for this parameter is ‘100’.
iterator	String specifying the unique identifier of the newest snapshot already retrieved by the client. Only snapshots created after (or before, depending on the value of the *ascending* parameter) that snapshot will be returned. The snapshot with the given iterator is not returned. • When omitted, snapshots are returned starting from the newest or oldest snapshot depending on the value of the *Ascending* parameter. • If no snapshot can be found with the given unique identifier, the web server will return a “HTTP 404 not found” response. • If the iterator does not follow the correct format, the web server will return a “HTTP 400 Bad Request” response.
type	Type specifying the format that the result is returned in. This must be one of the following formats:

json	Returns the result as a JSON encoded object.
jsonstream	Same as 'json', except that when requested from a browser, 'jsonstream' causes the result to be opened inside the browser rather than showing the "Save As" dialog.

- If the requested type is not one of the supported formats, the web server will return a "HTTP 400 Bad Request" response.

The default value for this parameter is 'jsonstream'.

Table 3-18: Snapshots request arguments

Response

The response from the server is a stream of the following layout (example).

```
[
  {
    "uuid"      : "0123456789ABCDEF0123456789ABCDEF01234567",
    "timestamp" : "2003-01-01 00:00:00",
    "name"      : "tag name",
    "id"        : 1,
    "archive"   : "mod1_Daily_Run",
    "snapshot"  : {
      "PN"   : "6557-123-4567",
      "SN"   : "11-22-3-44",
      "rnd"  : "38A40A0D1632C55",
      "tags" : {
        "mod1_LU_Run!API_OBS_DYAVG_FWD_PRV" : { "u" : "api", "v" : 136.88 },
        "mod1_LU_Run!API_STD_DYAVG_FWD_PRV" : { "u" : "api", "v" : 133.46 },
        "mod1 LU Run!CPL DYAVG FWD PRV"     : {              "v" : 1.0238 },
        ...
      },
      "ts"   : "2003-01-01 00:00:00.000"
  },
  ...
]
```

Figure 3-23: Snapshots JSON response

The following is a brief overview of the fields that can be returned for each snapshot.

Section	Field	Meaning
Meta data		
	uuid	Universally unique identifier for this snapshot. This can be used to uniquely identify any snapshot generated by any Flow-X flow computer.
	timestamp	The date and time that the snapshot was generated. *Note: This field is provided for convenience, and mirrors the "ts" field in the snapshot. Always prefer to use the "ts" field, as that field is covered by the snapshot's UUID and therefor traceable.*
	name	The "name" of the snapshot. This is the value of the tag that is configured in Flow-Xpress as the archive's id tag.
	id	The id field is the internal database id of the snapshot. This field gives an indication of how many snapshots (in total over all archives) the

		flow computer has created since the last time the “Clear archives” command was issued.
	archive	The name of the archive to which this snapshot belongs.
Snapshot		
	PN	The Product Number of the flow computer that generated this snapshot.
	SN	The Serial Number of the flow computer that generated this snapshot.
	rnd	A random number, which is used to guarantee that every archive can be uniquely identified (see below).
	ts	The date and time that the snapshot was generated. The values in the snapshots represent the state of the tags at this moment in time.
Tags		
	-	A tag object consists of a field, itself an object, which represents the name of the tag. E.g. “`mod1_LU_Run!API_OBS_DYAVG_FWD_PRV`”.
	value	The value of the tag. This may be a natural number, a string, or some other datatype, depending on the data type that has been configured for the tag.
	unit	(optional) A string representing the unit of the tag as configured in the flow computer. E.g. “`sm3`” or “`mbar_d`”.

Examples

Example #1

Request: /snapshots

This request returns at most 100 snapshots (the default value for “count”), sorted by oldest first. Snapshots are returned from all archives.

Example #2

Request: /snapshots?archive=mod1_Batch

This request returns at most 100 snapshots (the default value for “count”) from archive “mod1_Batch”, sorted by oldest first.

Example #3

Request: /snapshots?count=10

This request returns at most 10 snapshots, sorted by oldest first. Snapshots are returned from all archives.

Example #4

Request: /snapshots?ascending=0

This request returns at most 100 snapshots (the default value for “count”), sorted by newest first. Snapshots are returned from all archives.

Example #5

Request: /snapshots?iterator=0123456789ABCDEF0123456789ABCDEF01234567

This request returns at most 100 snapshots (the default value for “count”, sorted by oldest first, starting *after* the snapshot with the given iterator. Snapshots are returned from all archives.

Example #6

Request:
/snapshots?archive=mod1_Batch&count=10&ascending=0&iterator=0123456789ABCDEF0123456789ABCDEF01234567

This request returns at most 10 snapshots from archive “mod1_Batch”, sorted by newest first, starting *after* the snapshot with the given iterator.

Working with the /snapshots service

Uniquely identifying snapshots

When using the /snapshots web service, each snapshot can be uniquely identified, even among different applications and flow computers using the snapshot’s UUID. This works as follows. The UUID is calculated by taking the SHA1 hash over the contents of the object in the snapshot field in the JSON output. Each snapshot contains a number of fields to ensure that each snapshot will generate a unique hash value; even if all the tag values in the snapshot are identical. The most important field to mention is the “rnd” field, which contains a sequence of random data. This helps to ensure that even two snapshots taken on the same flow computer (PN/SN combination) at the same time with the same tag values can be distinguished from one another.

Retrieving all snapshots from a flow computer

In order to retrieve the full set of snapshots stored on a flow computer, start by simply requesting the /snapshots web service with no parameters. Optionally, you may specify the name of an archive to retrieve only snapshots belonging to a specific archive. See Example #1 or Example #2. This will return the first set of snapshots. After this initial request you can use the UUID of the last snapshot that was returned as the “iterator” parameter for the next request. See Example #5. This process can be repeated until a request returns an empty list (“[]”), indicating no snapshots newer than the snapshot with the given UUID exist.

Monitoring for new snapshots

You can monitor for new snapshots by requesting only snapshots newer than a given snapshot UUID, as per the previous section. Since snapshot UUIDs are stable, the web service will return the empty list as long as no new snapshots have been created.

Retrieving snapshots in a multi-module set up

The /snapshots web service operates on a per-module basis. It is therefore not possible to retrieve snapshots from a different module. In order to retrieve snapshots from a multi-module set up you need to retrieve the snapshots from each module individually.

Tags

Tags encompass all values that are externally readable and writable. These include communication tags, parameters and system settings.

Request

Reading tags can be done by specifying "/tags" in the URL. The following arguments can be specified in the URL:

Argument	Description
IDFilter	Optional string argument containing the ID's of the requested tags. Multiple ID's can be separated using a comma-character. A range of ID's can be specified using the dash-character (e.g. "1,2,3-5,80-100"). When omitted all tags are returned.
Filter	Optional string argument for filtering out tags. It can be the name of a predefined set of tags or a custom filter expression (e.g. "(ID < 100) AND (LEVEL = 1000)"). This argument is ignored when IDFilter is specified. Supported predefined sets are: "params" Returns all parameters. "writable" Returns all tags that are writable. "local" Returns all tags that are located on the module being queried.
Units	Optional string argument for obtaining tags in a specific unit (only applicable when IDFilter is specified). Multiple units can be separated using a comma-character (e.g. "16892156,0,,13849028"). The position of the unit in the comma-separated string corresponds to the tag at the same position in the IDFilter string.
Formats	Optional string argument for obtaining tags formatted with a specific format (only applicable when IDFilter is specified). Multiple formats can be separated using a comma-character (e.g. "`MC4wMA==,IyMj`"). The position of the format in the comma-separated string corresponds to the tag at the same position in the IDFilter string. The format (e.g. "###.##") is BASE64 encoded. This is necessary so that it can be encoded safely in a comma-separated string which is to be placed in an URL.
Fields	Optional integer mask which specifies which fields to return. When omitted the ID, value, name, unit and text are returned. Instead of an integer mask it is also possible to specify the string "all" which returns all fields. 0x00000001 ID (unique ID as a number). 0x00000002 Name (Name as configured in the application). 0x00000004 Text (Text in the requested language). 0x00000008 Multi-lingual text key (e.g. "%tag.pt1% 1") 0x00000010 Units (e.g. "kg/s"). 0x00000020 Unit-ID (e.g. "16802891", see Units web-service). If the Unit-ID = 16777218, the tag is considered a text-value and should be edited through a regular keyboard. In all other cases the value should be edited through a

		numerical keypad.
	0x00000040	Description (description in the requested language).
	0x00000080	Multi-lingual description-key (e.g. “tagdesc.pt1”).
	0x00000100	Address (e.g. “Sheet1!B7”)
	0x00000200	Value.
	0x00000400	Format (e.g. “#.##”)
	0x00000800	Writable level (-1<=not externally writable, 0>=required level for writing to the tag/parameter).
	0x00001000	Writable flag (0=not writable, 1=writable)
	0x00002000	Default tag-value (used for auto-reset tags).
	0x00004000	Node-Id on which the tag lives (panel=0, module1=1, etc..).
	0x00008000	Auto-reset flag (0=regular tag, 1=auto-reset tag).
	0x00010000	Retentive flag (0=not retentive, 1=retentive)
	0x00020000	Minimum allowed value
	0x00040000	Maximum allowed value
	0x00080000	Short text (without any prefix/suffix)
	0x00100000	Multi-lingual short text key (e.g. “tag.pt1”)
Options	Optional integer mask which specifies additional options.	
	0x00000001	Prevents the value from being formatted in any way (see ‘RawValues’ argument below).
	0x00000002	Returns the XML-tags in compact form (e.g. <t> instead of <tag>).
	0x00000004	Includes an index attribute for all requested tags.
	0x00000008	Returns the configuration signature of the tags (the value of the tag is not included in this signature). This value is useful for detecting whether the tags configuration has changed.
	0x00000010	Returns the configuration signature of all the parameter-tags (the value of the parameter is not included in this signature).
	0x00000020	Returns the start date/time of the tags-subsystem. This value is useful for detecting whether the device has been restarted which causes the ‘cacheids’ to be reset.
	0x00000040	Prevents conversion of enumeration values to their textual form. E.g., when specified, returns ‘1’ instead of ‘Enabled’.
RawValues	Optional boolean argument. When this option is set, the value is returned in its raw form, and no unit-conversion, enumeration-translation, number-formating or date/time formatting is performed. Using ‘RawValues=1’ is the equivalent to enabling bit 0x00000001 in the ‘options’ field.	
Language	Optional ID of the requested language (number). When omitted the default language is used.	
CacheID	Optional cache ID (string) of last received data-set. When omitted a fresh data-set is returned.	

Table 3-19: Tags request arguments

Response

The response from the server is an XML-stream of the following layout (example):

```
<tags cacheid="32" >
  <tag
    index="0"
    id="10"
    name="mod3_mysheet!PT"
    text="Druk opnemer 3"
    shorttext="Druk opnemer"
    desc="Put your description here"
    unit="kg/s"
    unitid="16890281"
    format="#.##"
    address="mod3 mysheet!B17"
    value="6.7889"
    default="100"
    level="1000"
    writable="1"
    autoreset="1"
    retentive="0"
    node="3"
    min="-1000"
    max="1000"
  />
  ...
</tags>
```

Figure 3-24: Tags XML response

Only those fields are returned that were specified in the request.

Caching

The cacheid can be used to request new tag data only when needed. When the cacheid is omitted, a fresh result is returned. The cacheid is in fact a very large counter. When a request is made and a cacheid is specified, only the changed tags since the previous cacheid are returned. This makes it possible to efficiently poll the tags on the device.

Example #1

Request: /tags?idfilter=1,2,3&cacheid=600
Response:

```
<tags cacheid="605" >
  <tag id="1" value="109" />
  <tag id="3" value="66" />
</tags>
```

Figure 3-25: Alarms XML response (example 1)

In the response above, tag 2 is not included because it has not changed since cacheid 600.

Example #2

Request: /tags?idfilter=1,2,3&cacheid=100
Response:

```
<tags cacheid="605" >
  <tag id="1" value="109" />
  <tag id="2" value="54.6" />
  <tag id="3" value="66" />
</tags>
```

Figure 3-26: Alarms XML response (example 2)

In the response above, all tags are included because they have all changed since cacheid 100.

Example #3

Request: /tags?idfilter=1,2,3&cacheid=605
Response:

```
<tags cacheid="610" >
</tags>
```

Figure 3-27: Alarms XML response (example 3)

In the response above, no tags are included because they have not changed since cacheid 605. Note that a new cacheid has been returned, this is because other tags have been changed. The client should always use the last returned cacheid for new requests. This is important so that the device can handle the request in the most efficient way possible.

Units & Enumerations

Units encompass all units and enumerations supported by the system. Units themselves also support conversion factors for converting a value from one unit to another.

Unit ID

An unit ID consists is a 32 bit number, split up into 3 parts: {type}{type-id}{item-id}

Type (8 bits)	0: Regular data-type 1: unit-item 2: enum-item 3: unit-type 4: enum-type 5: unit-types and unit-items 6: enum-types and enum-items all: all unit-types, unit-items, enum-types, enum-items
Type-id (16 bits)	Unit or enum **type** ID
Item-id (8 bits)	Unit or enum **item** ID (only used when type is 1 or 2)

This structure was chosen to make is possible to quickly determine to which type an item belongs.

Standard data-types

The following standard data-types are supported by the flow computer:

Unit ID	Description
0	No data type specified (treat as numeric).
1	Text
2	Numeric
3	Date/time
4	Boolean
5	Integer
6	ID of unit item
7	ID of enumeration type
8	IP address

Table 3-20: Standard data-types

Unit types

A unit-type ID starts with the number 3 in the first byte. A unit-type is called a “unit” in English and “grootheid” in Dutch. Some sample unit type ID’s are:

UnitID	Description
0x03000100	Acceleration
0x03000500	Energy
0x03000e00	Kinematic viscosity
. . .	…

Table 3-21: Example unit-type IDs

These unit-types are a resource of the flow computer and can be obtained via a web-service.

Request

Reading unit-types can be done by specifying “/units?type=3” in the URL:

Argument	Description
Filter	Optional type-ID. When omitted all unit-types are returned. The filter can be a specific unit-type ID such as 0x03000500 which when specified will return only that item.
Fields	Optional integer mask which specifies which fields to return. When omitted or when “all” is specified, all fields are returned: 0x0001 ID (unique ID as a number). 0x0004 Name (e.g. “acceleration”) 0x0008 Text (e.g. “Acceleration”)
Language	Optional ID of the requested language (number). When omitted the default language is used.

Table 3-22: Unit types request arguments

Response

```
<units count="4">
  <unit id="3423531" name="energypermass" text="Energy per mass" />
  <unit id="3423532" name="energypermole" text="Energy per mole" />
  <unit id="3423533" name="energypervolume" text="Energy per volume" />
  <unit id="3423534" name="temperature" text="Temperature" />
</units>
```

Figure 3-28: Unit types XML response

Unit items

A unit-item ID starts with the number 1 in the first byte. A unit-type is called a “property” in English and “eenheid” in Dutch. Some sample unit item ID’s are:

UnitID	Description	Factor	Offset
0x01000101	Meters per second squared (m/s2)	1	0
0x01000102	Kilometers per second squared (km/s2)	1000	0
0x01000103	Inch per second squared (in/s2)	0.0254	0
. . .	...		

Table 3-23: Example unit-item IDs

These unit-items are a resource of the flow computer and can be obtained via a web-service.

Request

Reading unit-items can be done by specifying “/units?type=1” in the URL:

Argument	Description
Filter	Optional type-ID. When omitted all unit-items are returned. The filter can be a specific unit-type ID such as 0x03000500 which when specified will return all the unit-items belonging to that type. It can also be a unit-item ID which when specified will return only that item.
Fields	Optional integer mask which specifies which fields to return. When omitted or when “all” is specified, all fields are returned: 0x0001 ID (unique ID as a number). 0x0002 Parent ID (ID of the unit-type which acts as a parent of the item). 0x0004 Name (e.g. “kg_m3_n”) 0x0008 Text (e.g. “kg/m3(n)”) 0x0010 Description (e.g. “kilogram per normal cubic meter”) 0x0020 Conversion factor (e.g. “0.001”) 0x0040 Conversion offset (e.g. “0”)
Language	Optional ID of the requested language (number). When omitted the default language is used.

Table 3-24: Unit items request arguments

Response

```
<units count="3">
  <unit id="15732235" parentid="3423532" name="j_kg"
        desc="joule per kilogram" text="J/kg" factor="1.0" offset="0.0" />
  <unit id="15732236" parentid="3423532" name="kj_kg"
        desc="kilojoule per kilogram" text="KJ/kg"
        factor="1000.0" offset="0.0" />
  <unit id="15732237" parentid="3423532" name="mj_kg"
        desc="megajoule per kilogram" text="MJ/kg"
```

```
        factor="1000000.0" offset="0.0" />
</units>
```

Figure 3-29: Unit items XML response

Enumeration types

A enumeration-type ID starts with the number 4 in the first byte. Some sample enumeration type ID's are:

UnitID	Description
0x04000100	Period
0x04000500	Status
. . .	…

Table 3-25: Example enumeration-type IDs

These enumeration-types are a resource of the flow computer and can be obtained via a web-service.

Request

Reading enumeration-types can be done by specifying "/units?type=4" in the URL:

Argument	Description
Filter	Optional type-ID. When omitted all enumeration-types are returned. The filter can be a specific enumeration-type ID such as 0x04000500 which when specified will return only that item.
Fields	Optional integer mask which specifies which fields to return. When omitted or when "all" is specified, all fields are returned: 0x0001 ID (unique ID as a number). 0x0004 Name (e.g. "period") 0x0008 Text (e.g. "Period")
Language	Optional ID of the requested language (number). When omitted the default language is used.

Table 3-26: Enumeration types request arguments

Response

```
<units count="4">
  <unit id="6423532" name="period" text="Period" />
  <unit id="6423533" name="status" text="Status" />
  …
</units>
```

Figure 3-30: Enumeration types XML response

Enumeration items

An enumeration-item ID starts with the number 1 in the first byte. An enumeration item is analogous to an item in a combo-box. Some sample enumeration item ID's are:

UnitID	Description	Value
0x02000101	Second	1
0x02000102	Minute	2
0x03000103	Hour	3
...	...	

Table 3-27: Example enumeration-item IDs

These enumeration-items are a resource of the flow computer and can be obtained via a web-service.

Request

Reading unit-items can be done by specifying "/units?type=2" in the URL:

Argument	Description
Filter	Optional type-ID. When omitted all enumeration-items are returned. The filter can be a specific unit-type ID such as 0x04000500 which when specified will return all the enumeration-items belonging to that type. It can also be a enumeration-item ID which when specified will return only that item.
Fields	Optional integer mask which specifies which fields to return. When omitted or when "all" is specified, all fields are returned: 0x0001 ID (unique ID as a number). 0x0002 Parent ID (ID of the enumeration-type which acts as a parent of the item). 0x0004 Name (e.g. "second") 0x0008 Text (e.g. "Second") 0x0020 Value (e.g. "1")
Language	Optional ID of the requested language (number). When omitted the default language is used.

Table 3-28: Enumeration items request arguments

Response

```
<units count="5">
  <unit id="35732235" parentid="6423532" name="second"
        text="Second" value="1" />
  <unit id="35732236" parentid="6423532" name="minute"
        text="Minute" value="2" />
  <unit id="35732237" parentid="6423532" name="hour"
        text="Hour" value="3" />
  <unit id="35732238" parentid="6423532" name="day"
        text="Day" value="4" />
```

```
  <unit id="35732239" parentid="6423532" name="month"
        text="Month" value="5" />
</units>
```

Figure 3-31: Enumeration items XML response

WriteTags

The WriteTags web-service allows for setting tag and parameter values. Only those tags are writable that have been marked as writable in the flow computer configuration. These may include communication tags, parameters alarm limits/deadbands/delays and system settings.

Request

Writing tags can be done by specifying "/writetags" in the URL. The following arguments can be specified in the URL:

Argument	Description
Userkey	Required userkey of the logged in user. The tags that are being written must have a level lower or equal to the logged in user.
ErrorDetails	Optional argument that when set to "1", returns detailed information about the result of the write-operation. Since multiple tags can be written through this web-service, it is possible than some write-operations are successful and others aren't. In general, when the write-operation isn't completely successful, an error is returned (most likely 403 – bad request). When ErrorDetails=1, it is possible that 200 is returned together with an XML stream containing the detailed errors.
Writing tags by ID (e.g. "/writetags?tag677=0.78&unit677=167997907")	
Tag[id]	ID and value of the tag to be written. The number followed by the "Tag" text identifies the ID of the tag followed by its new value (e.g. "Tag101=34.4"). This argument can be repeated in order to write multiple tags at once.
Unit[id]	Optional unit-id of the tag being written. When specified, the value of the tag is interpreted in that unit, rather than the unit of the tag it selves. For instance, if the tag is of unit "kg", and the value being written is in "mg", then the server will convert the value from "mg" to "kg". The number followed by the "Unit" text identifies the ID of the tag followed by the unit-ID (e.g. "Unit101=16546743").
Format[id]	Optional format of the tag being written. When specified, the value of the tag is interpreted according to that format. When for instance, specifying a date/time value, you can tell the server in what format it is, so it can be interpreted correctly. The number followed by the "Format" text identifies the ID of the tag followed by the format (e.g. "Format5=dd-mm-yyyy%20hh-mm-ss").
Writing tags by Name (e.g. "/writetags?name=sysglobal!clear_events&value=1")	
Name[seq-nr]	Name of the tag to be written. When writing multiple tags, the 'seq-nr' can be used to specify which name/value/unit/format combinations belong together. Example: "/writetags?name1=sheet1!mytag&value1=365&name2=sheet2!mytag2&value2=478".
Value[seq-nr]	Value of the tag to be written.
Unit[seq-nr]	Optional unit-id of the tag being written. When specified, the value of the tag is interpreted in that unit, rather than the unit of the tag it selves. For instance, if

	the tag is of unit “kg”, and the value being written is in “mg”, then the server will convert the value from “mg” to “kg”.
Format[seq-nr]	Optional format of the tag being written. When specified, the value of the tag is interpreted according to that format. When for instance, specifying a date/time value, you can tell the server in what format it is, so it can be interpreted correctly.

Table 3-29: WriteTags request arguments

Example #1

Request: /writetags?tag76=90&tag77=91&userkey=267AB6

This request writes value “90” to tag with ID 76, and also value “91” to tag with ID 77.

Example #2

Request: /writetags?tag30=76.3&unit30=16783256&userkey=267AB6

Assume that tag 30 uses the unit “kg/s” in the flow computer.
This request writes value “76.3” (which is in the unit “mg/s” = 16783256) to the flow computer. The flow computer translates “76.3” from “mg/s” to “kg/s” and stores the result in the flow computer.

Example #3

Request: /writetags?tag55=06-06-2009%204:16:56&format55=mm-dd-yyyy%20hh:mm:ss
(Userkey omitted for clarity of reading)

This request writes a date/time value in a specific format to tag 55.

Example #4

Request: /writetags?name=sysglobal!clear_events&value=1&userkey=267AB6

This request write ‘1’ to tag ‘sysglobal!clear_events’.

File-request

Besides writing tags through URL commands, it is also possible to send XML data to the server which contains the tags to be written. In order to do this, specify “/writetags” in the URL and submit a form with a file called ‘file’ to the server. The “Userkey” is required in the URL and optionally the “ErrorDetails” argument can be specified in the URL.

Writing a file to the flow computer is not supported through a web browser, but has to be implemented programmatically by posting a file. For example in VB.NET this would look like::

Dim WebClient as New System.Net.WebClient

WebClient.DownloadFileAsync(New System.Uri("http://10.1.7.30/writetags?errordetails=1&userkey=67E555CF0C2F34C4"), "C:\test.xml")

The file should contain a XML document with the following layout:

```
<tags>
  <t id="5" v="67" />
  <t n="mod1_system!ipaddr1" v="168.0.1.2" />
  ...
</tags>
```

Figure 3-32: WriteTags XML request

The value of the tag should be in the unit of the tag on the device. A tag can either be identified through its id or its name.

Response

When 'ErrorDetails=1' and the operation fails (partially), detailed error information is returned. When an operation fails completely, the server can return 400 – Bad Request, indicating that the whole operation failed. When 'ErrorDetails=1' and the server returns 200, the following XML stream is returned. Events are only returned for those items that have failed.

```
<events>
  <event time="01/03/2010 11:35:38.534" sev="err" loc="tags"
         msg="tag 78 (calc_Gas_M_Stn!COMP_IC4_CUR) : is not writable" />
  …
</events>
```

Figure 3-33: WriteTags XML response