ADCON TELEMETRY

addUPI

URL Programming Interface

Version 1.2.2

SMART WIRELESS SOLUTIONS

ADCON TELEMETRY GMBH
INKUSTRASSE 24
A-3400 KLOSTERNEUBURG
AUSTRIA
TEL: +43|2243|38280-0
FAX: +43|2243|38280-6
http://www.adcon.at

ADCON INTERNATIONAL INC
2050 LYNDELL TERRACE
SUITE 120
CA-95616 DAVIS, USA
TEL: +1|530|7531458
FAX: +1|530|7531054
http://www.adcon.at

Release 1.2.2, July 2009

1. Table Of Contents

2. Definitions

2.1. Scope

This paper defines the communication protocol between various TCP/IP capable components of Adcon Telemetry's addNET telemetry system. Some examples of components that may implement this protocol are given below:

- an addVANTAGE Pro client and an addVANTAGE Pro server
- an addVANTAGE Pro client and a Telemetry Gateway (e.g. an A840 base station)
- a Telemetry Gateway and an A740 RTU (remote telemetry unit)

The specification described herein belongs to the application layer according to the ISO OSI layered system.

2.2. Overview

To allow the implementation of the addUPI protocol on a given component, several assumptions are made:

- The underlying component has enough hardware and software resources to implement a TCP/IP stack
- An HTTP implementation exists (a server and/or client, depending on the data direction flow)

The protocol calls for a client and a server. The client will issue HTTP requests to the server and the server will return responses to these requests. The client requests use the GET method while the responses return an XML document. This paper defines the format of the requests and the answers.

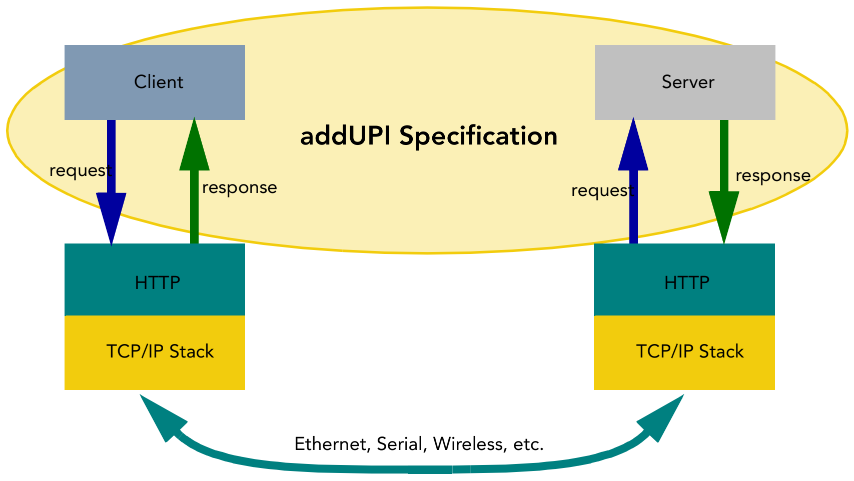

Figure 1. addUPI Specification's scope.

2.3. Naming

There are various objects that need to communicate in an addNET system, no matter if they use this specification or another. The best representation of those objects from a physical and geographical perspective is that of *nodes*. The nodes are hierarchically organized on several levels. A node is an element of the network participating in the movement of the data; some nodes may interface with the outside world (e.g. sensors, actuators). In addition, a node may implement—among other things—a local storage, therefore caching the data from the nodes situated hierarchically below.

A node may be an addVANTAGE server, an Telemetry Gateway, an A733 RTU, an intelligent sensor controller on a bus, a dumb sensor, an actuator or a virtual sensor (a value resulting out of one or more physical or virtual sensors).

Meanwhile, from the addUPI perspective, there are servers and clients. Each node will be assigned a logical ID that must be unique for a given server. If a node is replicated from a server to a client (which can be also a server for other clients), it will get another ID, which will be unique on the new server. There are no special numbering rules that are imposed as long as the IDs remain unique on a given server.

Nodes may have attributes, expose functions and/or issue notifications. The attributes may be read or written, the functions may be invoked in order for the node/tag to perform an operation, while the notifications are generated in case of outstanding situations (events).

A representation of a network can be seen in Figure 2.

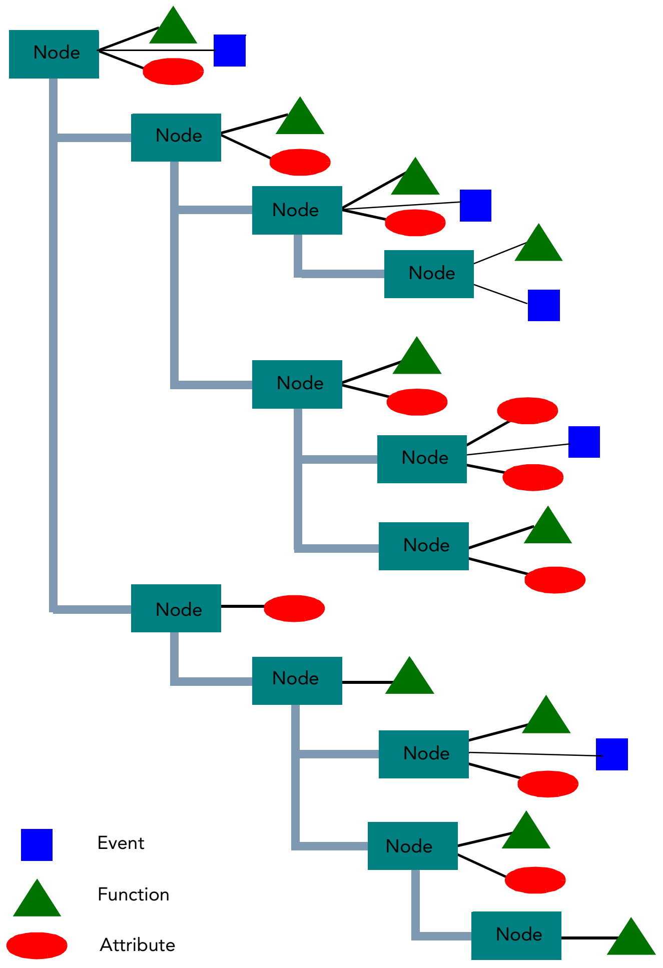

Figure 2. The naming convention used in the addUPI specification.

From the addUPI perspective, only those nodes (servers and clients) supporting a TCP/IP stack are relevant; all other types of devices in a network can be addressed through addUPI only if a proxy for the respective devices exist. The TCP/IP capable nodes are addressed by their standard network name, either using their IP address or their Internet name if a DNS is available. The proxy-ed nodes are addressed by their ID number.

2.4. Getting Connected

A typical session between a client and a server communicating over addUPI follows the steps described below:

1. The client discovers the server: the name of the server (or its IP address) must be known;
2. The client authenticates itself;
3. The client requests the server to return its capabilities: in this context, capability can be a list of nodes that are proxy-ed;
4. Inquire each node about its capabilities: this means getting the list of attributes and/or functions the node has and/or is able to execute (this step can be done simultaneously with step 3 in that the whole configuration of a node is requested, see "getconfig" on page 16 for more details);
5. Further, the client can act upon attributes (get/set), call functions or retrieve data (which is in fact also a particular case of a function call). Conversely, nodes may issue notifications on pending events to the client.

2.5. addUPI Reference

Throughout the following definitions it is expressly stated that for both requests and answers, case counts. This is especially important for functions, events and attributes names (i.e. "Type" and "type" are two different attributes).

2.5.1. General Format of a Request

A request formulated by a client has a generic form as given below

```
http://hostname:port/addUPI?function=fn&session-id=nnnn&param1=p1&
param2=p2&...paramn=pn
```

where

- `hostname` is the address of the server
- `port` is the port number to the server listens upon (if not specified, it is standard 80; it is recommended to use the port 80 for Telemetry Gateways and 8080 for addVANTAGE servers)
- `addUPI` is the name of the handler on the server that will be invoked to handle the request; this may be a servlet, a cgi, etc.
- `function` is the name of the invoked method
- `session-id` is a number identifying a certain client (obtained after authentication)
- `param<i>=pn` are the parameters requested by a particular function

2.5.2. Response Format

The response will return an XML document. There are two main ways to code the result: in plain ASCII, or binary (compressed). The compression method used is the standard Lempel-Ziv coding (LZ77).

Some examples of returned XML files are shown in "addUPI XML Responses" on page 25.

2.5.3. Data types

Following data types are defined in addUPI:

Data Type	Description	Example
int	four-byte signed integer	`<int>-12</int>`
boolean	false or true	`<boolean>true</boolean>`
string	ASCII string	`<string>hello world</string>` or hello world
double	double-precision signed floating point number	`<double>-12.214</double>`
date	date/time	`<date>20000714T12:05:33</date>`
base64	base64-encoded binary	`<base64>eW91IGNhbid0IHJlYWQgdGhpcyE=</base64>`
array	a sequence of objects	`<array> <int>23</int> <double>15.23</double> <string>test</string> </array>`
struct	a collection of key-value pairs	`<struct> <member name="an integer"> <int>2345</int> </member> <member name="a double"> <double>6.3</double> </member> <member name="a string"> <string>baburiba</string> </member> </struct>`

Note: *If no type is indicated, then the type is string by default.*

2.5.4. Date/Time Handling

As the date and time are specified in ASCII formats (ISO 8601), no special precautions are required. The date and time as strings refer generally to the local time. However, all devices will keep internally their time zone variable (TZ) and if required, the daylight saving time (DST). These variables are to be managed by the underlying operating system controlling the respective devices. They are defined as an attribute of the respective node (timeZone).

Alternatively date can also be specified in an ASCII `time_t` format. This is an integer in ASCII representing the number of seconds elapsed since 1 Jan 1970, 0:0:0. However, if a request is made in a certain format, the answer will be provided in the same format. The addvantage of the ASCII `time_t` is that it avoids many problems associated with time zones and daily saving time (DST) conversions.

2.5.5. Notifications

Notifications are used to inform a client that something unexpected (an event) happened with a node. The general concept is that a client makes requests to a server for data. However, in such cases where a server must inform a client that a special event occurred a change of communication direction is required: it's now the server that starts the communication and not the client.

addUPI deals with notifications in a similar manner to the ordinary data transfer. If it wants to accept notifications, an addUPI capable device must implement both an addUPI client as well as an addUPI server. If an addUPI client wants to be notified, it must first add itself as a listener for a particular event to the respective node; if it is the first node added as listener for the respective event, this will in effect enable the target node to send notifications.

Note that in the actual generation of Gateways, it may be that the automatic registration of a addNET participant is not supported because of the restricted memory of the Gateway. In this case, the user must register the receiver of the notification manually, e.g. in the Configuration GUI. If a participant calls the function to register as listener, an error code is returned to tell that this function is not supported.

If the target node is not TCP/IP capable and therefore can't support addUPI, a proxy node that supports addUPI must overtake this function. Thus, the proxy will keep a list with all addUPI nodes that are registered as listeners and will buffer notifications received from the nodes situated hierarchically below.

If a device that has to relay a notification can't contact a registered listener (if the connection is not permanent, e.g. over dial-up), it will wait until it will be contacted, then it will try again to re-send the pending notification(s).

When a device doesn't need to be notified anymore, it must remove itself as listener from the lower level node. This, in turn, must check if no other device is still registered as listener for that particular type of notification: if not, then the slave node will be effectively disabled from sending notifications. When the last registered node de-registers itself, then the slave will cease sending notifications.

There will be two possibilities to test notifications, by performing an immediate notification to a participant that is registered or that the listener tells the notifying party to notify me immediately. In both cases, only test data is transfered. Note that when testing the notifications, the device that later will send the notifications also must perform the test - it is not useful to let e.g. test the Configuration GUI if notifications function properly.

2.5.6. Templates

With addUPI it is possible to retrieve a node's description in its entirety. That means a client can enumerate the attributes, the functions and the events associated with a node. The following information is provided for each attribute:

- name: the internal name by which an attribute is referenced (required);
- display name: the attribute's human-readable name (optional);
- type: the type of the attribute value (string, int, etc.—required);
- description: a short description of the attribute (optional);
- permissions: specifies if an attribute is read-only, read-write or write-only (required);
- visible: this flag is intended to be used by a client that presents the attribute list to the user and wants to filter some attributes (optional).

The following information is provided for each function:

- name: the internal name by which a function is referenced (required);
- description: a short description of the function, i.e. what it is intended to do (optional);
- the parameter list; for each parameter, the following is specified:
  a. name: the internal name by which a parameter is referenced (required);
  b. type: the value type of the parameter (required);
  c. description: a short parameter description (optional)
- the return value; the following information is provided about the return value:
  a. type: the return value type (required);
  b. description: a short description of the return value (optional).

The following information is provided for each event:

- id: the internal id by which an event is referenced (required);
- name: the event name (required);
- description: a short description of the event (optional).

Such a detailed description of a node is required because a client can be a visual tool that can explore and discover an addVANTAGE network at runtime, thus giving the possibility to an user to dynamically get/set attributes or invoke functions. On another hand, to a programmer this information is not useful, because he can get it from a user manual.

From the description above it is clear that a lot of information is required in order to define a node. Because a lot of nodes in an addVANTAGE network are similar (i.e. all the A733 devices share the same definition, only some attribute values are different), and in order to reduce the amount of data flowing over the network, addUPI makes use of *templates*. A template represents the general description of a node (attribute definitions, function definitions, event definitions) and optionally, some of the nodes' attribute values.

It is possible to have all the definitions of the attributes, functions, events either in the definition of the node itself, or to separate the definition of the attributes, functions, events from the node by using templates, or finally, it can be a mixture between the two possibilities above. Anyway, even if a node is completely defined in line (meaning the full definition of attributes, functions, events is in the node itself), there must be a template specified for that node (an empty one).

2.5.7. addUPI Functions

A list of currently defined addUPI functions follows. They are divided in four classes:

- authentication functions
- configuration functions
- data transfer functions
- device specific functions (custom).

The authentication, configuration and data transfer functions are mandatory, while the device specific functions differ from device to device and may or may not be implemented.

Each function call returns an XML document. The DTDs associated with the returned XML documents are specified in the addUPI protocol. Each DTD defines two possible returns: one is the valid response (in case no error has occurred) and the other is the error response. The error response is defined as an integer/string pair. The integer is the error number and the string is the explanation of the error.

When the DTD `rs_functioncall.dtd` is specified as a return type, the type of the return value will be also specified. That is because `rs_functioncall.dtd` is a general format that can be used to describe any value type that conforms to the addUPI data types. As an example, both `login` and `logout` functions return a document of type `rs_functioncall`, but the `login` function returns a string (the `session-id`), while the `logout` function returns no value (`void`).

2.5.7.1. *Authentication Functions*

login

FORMAT

```
addUPI?function=login&user=user-name&passwd=user-password&
host-id=xxx&timeout=xxx&mode=t/z&version=1.2
```

DESCRIPTION

Request authentication from the server.

PARAMETERS

`user` is the user to be authenticated (mandatory).

`passwd` is the password (mandatory).

`host-id` is a unique number generated by the host at installation time.

Note: *The host-id is not used on current known implementations of addUPI and may be deprecated in the final release of the addUPI specification. Use with caution.*

`timeout` is the session duration (in seconds); if missing, the session is either valid until logout or a default timeout is used depending on the server implementation. See the remarks for additional information.

`mode` specifies how the response should be returned; it can be `t` (ASCII text) or `z` (compressed text)—if missing, `t` is implied.

`version` specifies the addUPI version to use for further communication (default: 1.0, supported values: 1.0 and 1.2)

RETURNS

An XML document of the type defined by `rs_functioncall.dtd`. Returned data type is string, and its description is the `session_id`. The `session-id` string will be then used in all subsequent addUPI requests. For more details, see also "Response of a **functioncall** Request" on page 29.

REMARKS

The session-id string must be unique in the context of a given server; its generation could be for example based on the requesting client's address plus a serially incremented number and some random added part. The host-id is required to uniquely identify a client, especially when it adds itself as a listener to a notification. Due to a variety of connecting methods (dial-up, dynamic IP addressing), the host IP address cannot be used for notifications, instead the host-id will be used.

The timeout starts counting after the last request, that is, the timeout counter will be reset which each valid request (invalid or malformed requests are not considered for resetting the timeout counter). In normal communication cases, a client will always execute a logout, so the timeout should never expire; it is however required in cases where the communication breaks up unexpectedly.

A server may (and it is desirable to) have its own timeout. In this case, and if the server built-in timeout is shorter than the timeout proposed by the client, then the server timeout will override the client timeout.

If the client specifies an addUPI protocol version that is not supported by the server, it may choose to return an error instead of a session ID. However, both the A840 and addVANTATE versions 4.x and 5.x will ignore this parameter and just use protocol version 1.0

logout

FORMAT `addUPI?function=logout&session-id=nnnn&mode=t/z`

DESCRIPTION Request de-authentication from the server.

PARAMETERS `session-id` is the result returned by the login function (mandatory).

`mode` specifies how the response should be returned; it can be `t` (ASCII text) or `z` (compressed text)—if missing, `t` is implied.

RETURNS An XML document of the type defined by `rs_functioncall.dtd`. Returned data type is void. For more details, see also "Response of a **functioncall** Request" on page 29.

REMARKS None.

2.5.7.2. Configuration Functions

getconfig

FORMAT
```
addUPI?function=getconfig&session-id=nnnn&id=nnnn&depth=m&
df=date-format&flags=f&mode=t/z
```

DESCRIPTION Returns the configuration of a node and of its underlying nodes, depending on the specified depth.

PARAMETERS `session-id` is the result returned by the login function (mandatory).

`id` is the node's ID— if missing, then the root tree will be returned.

`depth` signifies how many hierarchy levels should the capabilities be searched (and returned)—if not specified, the whole tree will be returned.

`df` is the date format. It can be `iso8601` or `time_t`. If a certain date format is specified, then the answer must be returned using that format. If missing, `iso8601` is implied. The `time_t` format represents the number of seconds elapsed since 1 Jan 1970 0:0:0 relative to the UTC meridian. If the answer is known not to include date values, it can be omitted.

`flags` specifies what should be returned for the specified depth. Following flags are currently defined: `a` (attributes), `e` (events) and `f` (functions). If missing, only the node(s) will be returned. Obviously, any combination of flags may be used, from none to all.

`mode` specifies how the response should be returned; it can be `t` (ASCII text) or `z` (compressed text)—if missing, `t` is implied.

RETURNS An XML document of the type defined by `rs_getconfig.dtd`. For more details, see also "Response to a **getconfig** Request" on page 25.

REMARKS The depth must be interpreted as follows: for a `depth=0`, only the specified node will be returned, but no details for the subordinate nodes. For a `depth=1`, the specified node plus its children nodes will be returned, and so on. For each node, the amount of information returned (i.e. attributes, functions, events) depends on the `flags` parameter.

gettemplate

FORMAT

```
addUPI?function=gettemplate&session-id=nnnn&template=templ&mode=t/z
```

DESCRIPTION

Returns the specified template.

PARAMETERS

`session-id` is the result returned by the login function (mandatory).

`template` points to a specific template—if not specified, all templates known by the inquired server will be returned.

`mode` specifies how the response should be returned; it can be `t` (ASCII text) or `z` (compressed text)—if missing, `t` is implied.

RETURNS

An XML document of the type defined by `rs_template.dtd`. For more details, see also “Response to a **gettemplate** Request” on page 31.

REMARKS

The templates are intended to reduce the traffic over the communication paths. Because the number of defined nodes in a system is finite, a server will specify the template by which a node is defined when returning a `getconfig` document. If a client does not have the template definition, then it can specifically ask for it using the `gettemplate` function call.

getattrib

FORMAT

```
addUPI?function=getattrib&session-id=nnnn&id=nnnn&attrib=name&
df=date-format&cache=y/n&mode=t/z
```

DESCRIPTION

Returns the value of the specified attribute of the specified node.

PARAMETERS

`session-id` is the result returned by the login function (mandatory).

`id` is the node’s ID—mandatory.

`attrib` is the attribute’s name—if not specified, all attributes of the node will be returned.

`df` is the date format. It can be `iso8601` or `time_t`. If a certain date format is specified, then the answer must be returned using that format. If missing, `iso8601` is implied. The `time_t` format represents the number of seconds elapsed since 1 Jan 1970 0:0:0 relative to the UTC meridian. If the answer is known not to include date values, it can be omitted.

`cache` can be `y` or `n` and specifies if the data should be retrieved from the cache of the node or directly from the tag (if the tag supports immediate requests)—if missing, default is `y` (i.e. retrieve from cache).

`mode` specifies how the response should be returned; it can be `t` (ASCII text) or `z` (compressed text)—if missing, `t` is implied.

RETURNS

An XML document of the type defined by `rs_getattrib.dtd`. For more details, see also “Response to a **getattrib** Request” on page 30.

REMARKS — The list of attributes and their meaning is particular to each node and depends on its type. For further details consult the technical manual of the particular node.

setattrib

FORMAT

```
addUPI?function=setattrib&session-id=nnnn&id=nnnn&attrib=name&
value=val&df=date-format&mode=t/z
```

DESCRIPTION — Sets the value of the specified attribute of the specified node.

PARAMETERS

`session-id` is the result returned by the login function (mandatory).

`id` is the node's ID—mandatory.

`attrib` is the attribute's name—mandatory.

`value` is the new value assigned to the specified attribute—mandatory.

`df` is the date format. It can be `iso8601` or `time_t`. If missing, `iso8601` is implied. The `time_t` format represents the number of seconds elapsed since 1 Jan 1970 0:0:0 relative to the UTC meridian. This flag should be used only if a date value must be set.

`mode` specifies how the response should be returned; it can be `t` (ASCII text) or `z` (compressed text)—if missing, `t` is implied.

RETURNS — An XML document of the type defined by `rs_functioncall.dtd`. Returned data type is void. For more details, see also "Response of a **functioncall** Request" on page 29, the XML Example (void).

REMARKS — The list of attributes and their meaning is particular to each node depending on its type. For further details consult the technical manual of the particular node.

2.5.7.3. Notification Functions

addlistener

FORMAT

```
addUPI?function=addlistener&session-id=nnnn&
id=nnnn&event=eventId&callback-url=url&callback-port=nnnnn&
callback-username=username&callback-password=password&mode=t/z
```

DESCRIPTION — Register a host to be notified by a node when a specific event occurs.

PARAMETERS

`session-id` is the result returned by the login function (mandatory).

`id` is the ID of the node that should send the notifications. If missing, then the root node is implied. It is however bad practice to not specify a target node, therefore this should be avoided.

`event` is the type of notification that must be issued (in other words, that will be enabled on the remote node). If missing, the listener will receive all possible notifications from the respective node.

`callback-url` is the URL where the notificatons are delivered. The URL must include the protocol (http:// or https://) and the hostname or IP address. It may include a port and a path (with or without trailing slash). For example, a simple

URL may be "http://apro5.adcon.at", a complex one may be "https://gateway.adcon.at:8089/addvantage/listener/". It is important that the user enters the URL that is reachable for the notifier, as else there might be problems, for example when using NAT.

`callback-port` is the port which is trailing after the URL. Please note that the port needs to be concatenated directly after the hostname, but before the path in the URL, if a path is given.

`callback-username` and `callback-password` are valid login name and password on the notification listener. This is mandatory for security reasons to avoid that any Internet participant could flood a listener with faked notifications.

`mode` specifies how the response should be returned; it can be `t` (ASCII text) or `z` (compressed text)—if missing, `t` is implied.

RETURNS An XML document of the type defined by `rs_functioncall.dtd`. Returned data type is integer, and its description is the registration ID number. The registration ID number will be then used in all subsequent notifications. For more details, see also "Response of a **functioncall** Request" on page 29.

REMARKS See also "notify" on page 19. Please again be aware that currently, some Gateways may not support this function, thus you only could register the notification listener using the Configuration GUI.

notify

FORMAT `addUPI?function=notify&df=date-format&mode=t/z`

DESCRIPTION Issue a notification (following an event) to the notification listener

PARAMETERS `df` is the date format. It can be `iso8601` or `time_t`. If missing, `iso8601` is implied. The `time_t` format represents the number of seconds elapsed since 1 Jan 1970 0:0:0 relative to the UTC meridian. If no date values are included in the parameter list, it can be omitted.

`mode` specifies how the response should be returned; it can be `t` (ASCII text) or `z` (compressed text)—if missing, `t` is implied.

`POST-Data`: in the request, there is also delivered an XML-document that contains the information about the notifications itself. Each notification type is represented by an XML structure, which is defined in a DTD or XSD. The listener is responsible for taking every possible XML that the notifier delivered, even though the message is not understood. It is possible that there are multiple notifications delivered within a single notification call (check out the XML schema).

testlistener

FORMAT
```
addUPI?function=testlistener&session-id=nnnn&
callback-url=url&callback-port=nnnnn&
callback-username=username&callback-password=password&mode=t/z
```

DESCRIPTION	Test from a party that wants to be notified if the notifier could deliver the notification. After issuing this command, the receiver of the command creates a new connection with the given URL / port / username / password and delivers a test notification, which is described in the XML schema part.
PARAMETERS	Same parameters as used in addlistener
RETURNS	An XML document of the type defined by `rs_functioncall.dtd`. Returned data type is string, and its description is either OK or an error code. For more details, see also "Response of a **functioncall** Request" on page 29.

removelistener

FORMAT	`addUPI?function=removelistener&session-id=nnnn&reg-id=xxxx&mode=t/z`
DESCRIPTION	De-register a host to be notified by a node when a specific event occurs.
PARAMETERS	`session-id` is the result returned by the login function (mandatory). `reg-id` is the registration ID number from the host when the client added itself as a listener (see also "addlistener" on page 18). `mode` specifies how the response should be returned; it can be `t` (ASCII text) or `z` (compressed text)—if missing, `t` is implied.
RETURNS	An XML document of the type defined by `rs_functioncall.dtd`. Returned data type is void. For more details, see also "Response of a **functioncall** Request" on page 29.
REMARKS	See also "notify" on page 19. Please again be aware that currently, some Gateways may not support this function, thus you only could register the notification listener using the Configuration GUI.

2.5.7.4. Data Transfer Functions

getdata

FORMAT	`addUPI?function=getdata&session-id=nnnn&id=nnnn&df=date-format&`
	`date=yyyymmddThh:mm:ss&slots=s&cache=y/n&mode=t/z`

DESCRIPTION	Returns data from the specified node(s).
PARAMETERS	`session-id` is the result returned by the login function (mandatory). `id` is the nodes' ID—if missing, an error (5–`node not found`) will be returned. If the specified node has children, then all the data down the tree until the last node will be returned. `df` is the date format. It can be `iso8601` or `time_t`. If a certain date format is specified, then the answer must be returned using the same format. If missing, `iso8601` is implied. The `time_t` format represents the number of seconds elapsed since 1 Jan 1970 0:0:0 relative to the UTC meridian.

`date` specifies the date of the newest stored slot on the client — the data returned will be strictly newer than this date. If missing, the last stored slot will be returned (see also the remarks section for more details).

`slots` is number of slots the client is prepared to accept (per node). The server may return less slots that requested, but never more—if missing, default one slot is implied. If a server returns less slots for a node than requested by the client, then the client assumes that there are no more data available for that node. The maximum number of slots requested by a client must not be greater than the value of the `getdataMaxSlots` attribute of the server (the root node in a `getconfig` response). If a client request more slots that specified by the `getdataMaxSlots` attribute, the server will return an error.

`cache` can be `y` or `n` and specifies if the data should be retrieved from the cache of the node or directly from the tag (if the tag supports immediate requests)—if missing, default is `y` (i.e. retrieve from cache).

`mode` specifies how the response should be returned; it can be `t` (ASCII text) or `z` (compressed text)—if missing, `t` is implied.

RETURNS

An XML document of the type defined by `rs_getdata.dtd`. For more details, see also "Response to a **getdata** Request" on page 28.

REMARKS

The date stamp should be interpreted by a server as follows:

- If the slot requested has a date stamp older than the oldest entry on the server, then a number of slots starting with the oldest stored slot will be returned;
- If the slot requested has a date stamp newer than the newest entry on the server, then an error will be returned (`14–no more data`);
- If data is available, then the slot with the date stamp strictly newer than (but not equal to) the requested date stamp will be returned; as a general rule, a client should always request the next slot with the date stamp retrieved from the last slot it received in a previous request. Example (simplified):
  `getdata(20010101T00:15:30)`
  returns the slot with the date stamp
  `data(20010101T00:19:55)`
  next request will be
  `getdata(20010101T00:19:55)`
  and answer will be
  `data(20010101T00:24:20)`
  and so on.

The "offset" of a measurement value is the timespan between the sample time and the end of the slot interval, the "duration" is the timespan between the start and the end of the slot (for example, a wind maximum at 8:14 for a slot started at 8:00 and stored at 8:15 would result in a timestamp of 8:15, a duration of 900 seconds and an offset of 60 seconds). Note that offset and duration are only returned when the client specified to use addUPI version 1.2 in the login functioncall and the addUPI server supports this protocol version; additionally, these slot elements are not supported by all tag types. If the duration and/or the offset is unknown or not supported, it is disallowed to send the duration/offset in the response. There is no special placeholder for unknown duration/offset, especially “0” is not meant as placeholder! A duration of 0 means a punctual measurement, like “what is the current state?”.

If the specified node has children, then the server will return data from all the children recursively, having the time stamps newer, but not equal to the specified time stamp in the request. For nodes that can’t provide it, an error will be returned (`14– no more data`).

The states for slots are: -99 to -1 for partial data (value is percentage of samples missing, i.e. -23 means that 23 percent of the samples were missing and 77 percent of the samples were taken), 0 for OK (100% of samples were taken), 1 for invalid data and 2 for missing data (0% of samples were taken). Note that slot states other than 0,1 and 2 are only returned when the client specified to use addUPI version 1.2 in the login functioncall.

2.5.7.5. Device Specific Functions

All device specific functions are called by the generic function call described below:

functioncall

FORMAT	`addUPI?function=functioncall&session-id=nnnn&id=nnnn&df=date-format&name=func-name&paramName1=p1&paramName2=p2...&mode=t/z`
DESCRIPTION	Calls a node specific function.
PARAMETERS	`session-id` is the result returned by the login function (mandatory). `id` is the node's or tag's ID—mandatory. `df` is the date format. It can be `iso8601` or `time_t`. If a certain date format is specified, then the answer must be returned using that format. If missing, `iso8601` is implied. The `time_t` format represents the number of seconds elapsed since 1 Jan 1970 0:0:0 relative to the UTC meridian. If the parameter list, or the answer is known not to include date values, it can be omitted. `name` is the name of the function to be invoked (mandatory). `paramName1...` are the parameters (parameter name and type depend on the function). The parameter names are defined in the device's template. `mode` specifies how the response should be returned; it can be `t` (ASCII text) or `z` (compressed text)—if missing, `t` is implied.
RETURNS	An XML document of the type defined by `rs_functioncall.dtd`. For more details, see also "Response of a **functioncall** Request" on page 29. Also the technical manual of the specific device must be consulted.
REMARKS	None

3. addUPI XML Responses

3.1. Scope

This chapter contains the Document Type Description for all addUPI XML document responses. In addition, some XML examples are also given. The examples are imaginary though, and may differ considerably from real life responses.

3.1.1. Response to a getconfig Request

DTD

```
<!--
    RS_GETCONFIG 1.0 01/07/2002

    Copyright 2000-2002 Adcon Telemetry srl

    This DTD describes the response document returned by the “getconfig” command.
-->
<!ELEMENT response (node | error)>
<!ELEMENT node (attribs?, functions?, events?, nodes?)>
<!ATTLIST node
    id CDATA #REQUIRED
    template CDATA #REQUIRED
    name CDATA #REQUIRED
    class CDATA #REQUIRED
    subclass CDATA #IMPLIED
>
<!ELEMENT attribs (attrib*)>
<!ELEMENT attrib (attribdef?, (boolean | int | string | double | date | base64 | array | struct))>
<!ATTLIST attrib
    name CDATA #REQUIRED
>
<!ELEMENT attribdef EMPTY>
<!ATTLIST attribdef
    dispname CDATA #IMPLIED
    type (boolean | int | string | double | date | base64 | array | struct) #REQUIRED
    descr CDATA #IMPLIED
    perm (ro | rw | wo) #REQUIRED
    visible CDATA #IMPLIED
>
<!ELEMENT nodes (node*)>
<!ELEMENT error EMPTY>
<!ATTLIST error
    code CDATA #REQUIRED
    msg CDATA #IMPLIED
>
<!ELEMENT functions (function*)>
<!ELEMENT function (params, return)>
<!ATTLIST function
    name CDATA #REQUIRED
    descr CDATA #IMPLIED
>
<!ELEMENT params (param*)>
<!ELEMENT param EMPTY>
<!ATTLIST param
    name CDATA #REQUIRED
    type (boolean | int | string | double | date | base64 | array | struct) #REQUIRED
    descr CDATA #IMPLIED
>
<!ELEMENT return EMPTY>
<!ATTLIST return
    type (boolean | int | string | double | date | base64 | array | struct | void) #REQUIRED
    descr CDATA #IMPLIED
>
<!ELEMENT events (event*)>
<!ELEMENT event EMPTY>
<!ATTLIST event
    id CDATA #REQUIRED
    name CDATA #REQUIRED
    descr CDATA #IMPLIED
>
<!ELEMENT boolean (#PCDATA)>
<!ELEMENT int (#PCDATA)>
<!ELEMENT string (#PCDATA)>
<!ELEMENT double (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT base64 (#PCDATA)>
<!ELEMENT array ((boolean | int | string | double | date | base64 | array | struct)*)>
<!ELEMENT struct (member*)>
<!ELEMENT member (boolean | int | string | double | date | base64 | array | struct)>
<!ATTLIST member
    name CDATA #REQUIRED
>
```

XML Example

```
<?xml version=”1.0” encoding=”ISO-8859-1” standalone=”no”?>
<!DOCTYPE response SYSTEM “rs_getconfig.dtd”>
<response>
    <node id=”1” template=”A730SD” name=”Root Node” class=”DEVICE” subclass=”A730MD”>
        <attribs>
            <attrib name=”DESCR”>
                <string>This is the root node</string>
            </attrib>
            <attrib name=”CODE”>
                <int>100</int>
            </attrib>
        </attribs>
        <nodes>
            <node id=”2” template=”A730MD” name=”Node 1” class=”DEVICE” subclass=”A730MD”>
                <attribs>
                    <attrib name=”DESCR”>
                        <string>The first node</string>
                    </attrib>
                    <attrib name=”CODE”>
```

```
                        <int>2002</int>
                    </attrib>
                    <attrib name=”CUSTOM_ATTR”>
                        <attribdef dispname=”Custom Attr” descr=”Custom attr defined inline” type=”boolean”
                           perm=”ro” visible=”true”/>
                        <boolean>true</boolean>
                    </attrib>
                </attribs>
                <nodes>
                    <node id=”3” template=”TEMP” name=”Temperature” class=”TAG” subclass=”TEMP”>
                        <attribs>
                            <attrib name=”DESCR”>
                                <string>The temperature sensor</string>
                            </attrib>
                            <attrib name=”ATTR_XYZ”>
                                <boolean>true</boolean>
                            </attrib>
                        </attribs>
                    </node>
                    <node id=”4” template=”HUMID” name=”Humidity” class=”TAG” subclass=”HUMID”>
                        <attribs>
                            <attrib name=”DESCR”>
                                <string>The fabulous tag</string>
                            </attrib>
                            <attrib name=”ATTR_ABC”>
                                <int>2</int>
                            </attrib>
                        </attribs>
                    </node>
                </nodes>
            </node>
            <node id=”5” template=”A720” name=”Node 2” class=”DEVICE” subclass=”A720”>
                <attribs>
                    <attrib name=”DESCR”>
                        <string>Second node</string>
                    </attrib>
                    <attrib name=”CODE”>
                        <int>2003</int>
                    </attrib>
                </attribs>
                <nodes>
                    <node id=”6” template=”TEMP” name=”Temperature” class=”TAG” subclass=”TEMP”>
                        <attribs>
                            <attrib name=”DESCR”>
                                <string>Temperature</string>
                            </attrib>
                            <attrib name=”ATTR_XYZ”>
                                <int>23</int>
                            </attrib>
                        </attribs>
                    </node>
                    <node id=”7” template=”HUMID” name=”Humidity” class=”TAG” subclass=”HUMID”>
                        <attribs>
                            <attrib name=”DESCR”>
                                <string>Humidity</string>
                            </attrib>
                            <attrib name=”ATTR_ABC”>
                                <boolean>false</boolean>
                            </attrib>
                        </attribs>
                    </node>
                </nodes>
            </node>
        </nodes>
    </node>
</response>
```

3.1.2. Response to a getdata Request

DTD

```
<!--
    RS_GETDATA 1.0 01/07/2002
    Copyright 2000-2002 Adcon Telemetry srl

    This DTD describes the response document returned by the “getdata” command.
    t is the timestamp
    s is the status: Partial (-99 to -1): the status + 100 is the % that was measured
      (so status = -23 means -23 + 100 = 77% measured), OK (0); BAD (1); MISSING (2)
    d is the duration which the value is valid. Leave this attribute away if it is unknown
    o is the offset for minimum and maximum values between the point in time when the
      minimum/maximum occured and the time when the observation period ended and the
      value was stored (the timestamp). Leave this attribute away if it is unknown.
          Thus, the following is true:
      1) observation period start = timestamp - duration
      2) minimum/maximum time = timestamp - offset
      3) observation period end = timestamp
      4) 0 <= offset <= duration
    type: if the value was MEASURED (0), CALCULATED (1), MANUALLY (2). If missing, it
      is assumed that it is MEASURED
-->
<!ELEMENT response (node* | error)>
<!ELEMENT node (v* | error)>
<!ATTLIST node
    id CDATA #REQUIRED
>
<!ELEMENT v (#PCDATA)>
<!ATTLIST v
    t CDATA #REQUIRED
    s CDATA #REQUIRED
    d CDATA #REQUIRED
    o CDATA #IMPLIED
    type CDATA #IMPLIED
>
<!ELEMENT error EMPTY>
<!ATTLIST error
    code CDATA #REQUIRED
    msg CDATA #IMPLIED
>
```

XML Example

```
<?xml version=”1.0” encoding=”ISO-8859-1” standalone=”no”?>
<!DOCTYPE response SYSTEM “rs_getdata.dtd”>
<response>
    <node id=”3”>
        <v t=”19990101T00:00:00” s=”0” d=”900”>15.5</v>
        <v t=”+900” s=”0” d=”900”>15.6</v>
        <v t=”+900” s=”0” d=”900”>15.5</v>
        <v t=”+900” s=”0” d=”900” o=”150”>15.9</v>
    </node>
    <node id=”4”>
        <v t=”19990101T00:00:00” s=”0” d=”900”>95.5</v>
        <v t=”+900” s=”0” d=”900”>95.6</v>
        <v t=”+900” s=”0” d=”900” type=”1”>95.5</v>
        <v t=”+300” s=”0” d=”300”>95.9</v>
    </node>
    <node id=”6”>
        <error code=”14”/>
    </node>
</response>
```

3.1.3. Response of a functioncall Request

DTD

```
<!--
    RS_FUNCTIONCALL 1.0 01/07/2002

    Copyright 2000-2002 Adcon Telemetry srl

    This DTD describes the response document returned by the “functioncall” command.
-->
<!ELEMENT response (result | error)>
<!ELEMENT result (boolean | int | string | double | date | base64 | array | struct | void)>
<!ELEMENT boolean (#PCDATA)>
<!ELEMENT int (#PCDATA)>
<!ELEMENT string (#PCDATA)>
<!ELEMENT double (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT base64 (#PCDATA)>
<!ELEMENT array ((boolean | int | string | double | date | base64 | array | struct)*)>
<!ELEMENT struct (member*)>
<!ELEMENT member (boolean | int | string | double | date | base64 | array | struct)>
<!ATTLIST member
    name CDATA #REQUIRED
>
<!ELEMENT error EMPTY>
<!ATTLIST error
    code CDATA #REQUIRED
    msg CDATA #IMPLIED
>
```

XML Example (Generic)

```
<?xml version=”1.0” encoding=”ISO-8859-1” standalone=”no”?>
<!DOCTYPE response SYSTEM “rs_functioncall.dtd”>
<response>
    <result>
        <int>253</int>
    </result>
</response>
```

XML Example (void)

```
<?xml version=”1.0” encoding=”ISO-8859-1” standalone=”no”?>
<!DOCTYPE response SYSTEM “rs_functioncall.dtd”>
<response>
    <result>
        <void/>
    </result>
</response>
```

XML Example (getheard)

```
<?xml version=”1.0” encoding=”ISO-8859-1” standalone=”no”?>
<!DOCTYPE response SYSTEM “rs_functioncall.dtd”>
<response>
    <result>
        <struct>
            <member name=”time”>
                <date>20000101T09:15:24</date>
            </member>
            <member name=”nodes”>
```

```
                <struct>
                    <member name=”1”>
                        <double>6.5</double>
                    </member>
                    <member name=”2”>
                        <double>6.3</double>
                    </member>
                    <member name=”7”>
                        <double>3.5</double>
                    </member>
                </struct>
            </member>
        </struct>
    </result>
</response>
```

3.1.4. Response to a getattrib Request

DTD

```
<!--
    RS_GETATTRIB 1.0 01/07/2002

    Copyright 2000-2002 Adcon Telemetry srl

    This DTD describes the response document returned by the “getattrib” command.
-->
<!ELEMENT response (attrib* | error)>
<!ELEMENT attrib (boolean | int | string | double | date | base64 | array | struct)>
<!ATTLIST attrib
    name CDATA #REQUIRED
>
<!ELEMENT error EMPTY>
<!ATTLIST error
    code CDATA #REQUIRED
    msg CDATA #IMPLIED
>
<!ELEMENT boolean (#PCDATA)>
<!ELEMENT int (#PCDATA)>
<!ELEMENT string (#PCDATA)>
<!ELEMENT double (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT base64 (#PCDATA)>
<!ELEMENT array ((boolean | int | string | double | date | base64 | array | struct)*)>
<!ELEMENT struct (member*)>
<!ELEMENT member (boolean | int | string | double | date | base64 | array | struct)>
<!ATTLIST member
    name CDATA #REQUIRED
>
```

XML Example (success)

```
<?xml version=”1.0” encoding=”ISO-8859-1” standalone=”no”?>
<!DOCTYPE response SYSTEM “rs_getattrib.dtd”>
<response>
    <attrib name=”NAME”>
        <string>Root Node</string>
    </attrib>
    <attrib name=”DESCR”>
        <string>This is the root node</string>
    </attrib>
</response>
```

XML Example (error)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<response>
    <error code="5" msg="Invalid node id."/>
</response>
```

3.1.5. Response to a gettemplate *Request*

DTD

```
<!--
    RS_TEMPLATE 1.0 01/07/2002

    Copyright 2000-2002 Adcon Telemetry srl

    This DTD describes the response document returned by the "gettemplate" command.
-->
<!ELEMENT response (template* | error)>
<!ELEMENT template (attribs?, functions?, events?)>
<!ATTLIST template
    name CDATA #REQUIRED
>
<!ELEMENT attribs (attrib*)>
<!ELEMENT attrib ((boolean | int | string | double | date | base64 | array | struct)?)>
<!ATTLIST attrib
    name CDATA #REQUIRED
    dispname CDATA #IMPLIED
    type (boolean | int | string | double | date | base64 | array | struct) #REQUIRED
    descr CDATA #IMPLIED
    perm (ro | rw | wo) #REQUIRED
    visible CDATA #IMPLIED
>
<!ELEMENT functions (function*)>
<!ELEMENT function (params, return)>
<!ATTLIST function
    name CDATA #REQUIRED
    descr CDATA #IMPLIED
>
<!ELEMENT params (param*)>
<!ELEMENT param EMPTY>
<!ATTLIST param
    name CDATA #REQUIRED
    type (boolean | int | string | double | date | base64 | array | struct) #REQUIRED
    descr CDATA #IMPLIED
>
<!ELEMENT return EMPTY>
<!ATTLIST return
    type (boolean | int | string | double | date | base64 | array | struct | void) #REQUIRED
    descr CDATA #IMPLIED
>
<!ELEMENT events (event*)>
<!ELEMENT event EMPTY>
<!ATTLIST event
    id CDATA #REQUIRED
    name CDATA #REQUIRED
    descr CDATA #IMPLIED
>
<!ELEMENT boolean (#PCDATA)>
<!ELEMENT int (#PCDATA)>
<!ELEMENT string (#PCDATA)>
<!ELEMENT double (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT base64 (#PCDATA)>
<!ELEMENT array ((boolean | int | string | double | date | base64 | array | struct)*)>
<!ELEMENT struct (member*)>
<!ELEMENT member (boolean | int | string | double | date | base64 | array | struct)>
```

```
<!ATTLIST member
    name CDATA #REQUIRED
>
<!ELEMENT error EMPTY>
<!ATTLIST error
    code CDATA #REQUIRED
    msg CDATA #IMPLIED
>
```

XML Example

```
<?xml version=”1.0” encoding=”ISO-8859-1” standalone=”no”?>
<!DOCTYPE response SYSTEM “rs_template.dtd”>
<response>
    <template name=”A730SD”>
        <attribs>
            <attrib name=”NAME” dispname=”Name” type=”string” descr=”Node name” perm=”rw”/>
            <attrib name=”DESCR” dispname=”Description” type=”string” descr=”Node’s short description”
                           perm=”rw”/>
            <attrib name=”CODE” dispname=”Code” type=”int” descr=”Node’s physical code” perm=”rw”/>
            <attrib name=”ATTR_X” dispname=”Attr X” type=”int” descr=”A custom attribute” perm=”rw” visi-
                           ble=”true”/>
            <!--    The next attribute definition shows that we can define static attributes - that have the
                           value specified in the template -->
            <attrib name=”CANROUTE” dispname=”Can route” type=”boolean” descr=”Specifies if the node can
                           route” perm=”rw”>
                <boolean>true</boolean>
            </attrib>
        </attribs>
        <functions>
            <function name=”method1” descr=”The method short description.”>
                <params>
                    <param name=”firstParam” type=”int” descr=”The param description.”/>
                    <param name=”secondParam” type=”double” descr=”The param description.”/>
                </params>
                <return type=”int” descr=”The return description.”/>
            </function>
        </functions>
        <events>
            <event id=”1” name=”Event 1” descr=”Event description”/>
        </events>
    </template>
</response>
```

3.2. Defined Error Numbers

This list of error numbers will be permanently updated.

Error Number	Explanation
1	Invalid request
2	Parameter not found - <parameter name>
3	Invalid parameter - <parameter name>
4	User is already logged in
5	Node not found

Error Number	Explanation
6	Template not found
7	Attribute not found
8	Authentication failed
9	Operation not supported
10	Invalid session ID
11	Maximum number of requested slots exceeded the server's capability (the getdataMaxSlots attribute)
12	Operation not allowed (e.g. set a ro attribute)
13	Not enough privileges
14	No more data available
15	No more data available and connection to the remote temporarily broken.
31	Invalid remote request (equivalent to radio error -1)
32	Command not implemented on remote (equivalent to radio error -2)
33	Remote returned an internal error (equivalent to radio error -3)
34	Invalid parameter sent to remote (equivalent to radio error -5)
35	Radio timeout
36	Radio network busy
40	No notification pending
90 - 99	Error - <general error message>[a]

a. The range 90 to 99 may be used by servers to return specific internal errors; they must be documented by each particular implementation.

4. Templates

4.1. Scope

This section contains the template definitions for all Adcon nodes to date.

4.2. Node Templates

4.2.1. A720 addIT

TEMPLATE NAME: A720

ATTRIBUTES:

Name	Type	Perm	Default Value	Example
id	int	ro	-	1234
template	string	ro	A720	
name	string	ro	addIT	Stammersdorf
class	string	ro	DEVICE	
subclass	string	ro	addIT	
manufacturer	string	ro	Adcon Telemetry AG	
type	string	ro	A720	
version	string	ro	-	ver. 1.7

Name	Type	Perm	Default Value	Example
code	int	ro	-	6480
date	date	ro	-	20000714T12:05:33
timeZone	string	ro	-	Europe/Vienna
batteryLevel	double	ro	-	6.5 (in volt)
internalTemperature	double	ro	-	22.6 (in °C)

FUNCTIONS: None.

EVENTS: None.

XML REPRESENTATION:

```
<?xml version = “1.0” encoding = “ISO-8859-1”?>
<!DOCTYPE response SYSTEM “rs_template.dtd”>
<response>
    <template name = “A720”>
        <attribs>
            <attrib name = “id” type = “int” perm = “ro”/>
            <attrib name = “template” type = “string” perm = “ro”>
                <string>A720</string>
            </attrib>
            <attrib name = “name” type = “string” perm = “ro”>
                <string>addIT</string>
            </attrib>
            <attrib name = “class” type = “string” perm = “ro”>
                <string>DEVICE</string>
            </attrib>
            <attrib name = “subclass” type = “string” perm = “ro”>
                <string>addIT</string>
            </attrib>
            <attrib name = “manufacturer” type = “string” perm = “ro”>
                <string>Adcon Telemetry AG</string>
            </attrib>
            <attrib name = “type” type = “string” perm = “ro”>
                <string>A720</string>
            </attrib>
            <attrib name = “version” type = “string” perm = “ro”/>
            <attrib name = “code” type = “int” perm = “ro”/>
            <attrib name = “date” type = “date” perm = “ro”/>
            <attrib name = “timeZone” type = “string” perm = “ro”/>
            <attrib name = “batteryLevel” type = “double” perm = “ro”/>
            <attrib name = “internalTemperature” type = “double” perm = “ro”/>
        </attribs>
        <functions/>
        <events/>
    </template>
</response>
```

4.2.2. A720 addIT (Series II)

TEMPLATE NAME: A720_V2

ATTRIBUTES:

Name	Type	Perm	Default Value	Example
id	int	ro	-	2345

Name	Type	Perm	Default Value	Example
template	string	ro	A720_V2	
name	string	ro	addIT	Stammersdorf
class	string	ro	DEVICE	
subclass	string	ro	addIT	
manufacturer	string	ro	Adcon Telemetry AG	
type	string	ro	A720	
version	string	ro	-	ver. 2.2
code	int	ro	-	8321
slotInterval	int	rw	900	(in seconds)
samplesPerSlot	int	rw	3	(in units)
date	date	ro	-	20000714T12:05:33
timeZone	string	ro	-	Europe/Vienna
batteryLevel	double	ro	-	6.5 (in volt)
internalTemperature	double	ro	-	22.6 (in °C)
uptime	string	ro	-	105 days, 5 hours, 10 minutes
rssi	int	ro	58	
pmpLow	int	ro	65	(in hundred of millivolts)
pmpHigh	int	ro	72	(in hundred of millivolts)

FUNCTIONS:

Name	Parameters	Return Value
clearcache	none	nothing

EVENTS: None.

XML REPRESENTATION:

```
<?xml version = “1.0” encoding = “ISO-8859-1”?>
<!DOCTYPE response SYSTEM “rs_template.dtd”>
<response>
    <template name = “A720_V2”>
        <attribs>
            <attrib name = “id” type = “int” perm = “ro”/>
            <attrib name = “template” type = “string” perm = “ro”>
                <string>A720_V2</string>
            </attrib>
            <attrib name = “name” type = “string” perm = “ro”>
                <string>addIT</string>
            </attrib>
            <attrib name = “class” type = “string” perm = “ro”>
                <string>DEVICE</string>
            </attrib>
            <attrib name = “subclass” type = “string” perm = “ro”>
                <string>addIT</string>
            </attrib>
            <attrib name = “manufacturer” type = “string” perm = “ro”>
                <string>Adcon Telemetry AG</string>
            </attrib>
            <attrib name = “type” type = “string” perm = “ro”>
                <string>A720</string>
            </attrib>
            <attrib name = “version” type = “string” perm = “ro”/>
            <attrib name = “code” type = “int” perm = “ro”/>
            <attrib name = “slotInterval” type = “int” perm = “rw”>
                <int>900</int>
            </attrib>
            <attrib name = “samplesPerSlot” type = “int” perm = “rw”>
                <int>3</int>
            </attrib>
            <attrib name = “date” type = “date” perm = “ro”/>
            <attrib name = “timeZone” type = “string” perm = “ro”/>
            <attrib name = “batteryLevel” type = “double” perm = “ro”/>
            <attrib name = “internalTemperature” type = “double” perm = “ro”/>
            <attrib name = “uptime” type = “string” perm = “ro”/>
            <attrib name = “rssi” type = “int” perm = “ro”>
                <int>58</int>
            </attrib>
            <attrib name = “pmpLow” type = “int” perm = “ro”>
                <int>65</int>
            </attrib>
            <attrib name = “pmpHigh” type = “int” perm = “ro”>
                <int>72</int>
            </attrib>
        </attribs>
        <functions>
            <function name = “clearcache”>
                <params/>
                <return type = “void”/>
            </function>
        </functions>
        <events/>
    </template>
</response>
```

4.2.3. A723 addIT (Series III) and A733 addWAVE

TEMPLATE NAME: A7X3_V1_4

ATTRIBUTES:

Name	Type	Perm	Default Value	Example
id	int	ro	-	5432
template	string	ro	A7X3_V1_4	
name	string	ro	addWAVE	Adcon Klosterneuburg
class	string	ro	DEVICE	
subclass	string	ro	-	addWAVE
manufacturer	string	ro	Adcon Telemetry AG	
type	string	ro	-	A733
version	string	ro	-	ver. 1.1
code	int	ro	-	14501
slotInterval	int	rw	900	(in seconds)

Name	Type	Perm	Default Value	Example
samplesPerSlot	int	rw	15	(in units)
date	date	ro	-	20000714T12:05:33
timeZone	string	ro	-	Europe/Vienna
batteryLevel	double	ro	-	6.5 (in volt)
internalTemperature	double	ro	-	22.6 (in °C)
uptime	string	ro	-	105 days, 5 hours, 10 minutes
rssi	int	ro	42	
pmpLow	int	ro	65	(in hundred of millivolts)
pmpHigh	int	ro	72	(in hundred of millivolts)
powerOutput	int	ro	155	

FUNCTIONS:

Name	Parameters	Return Value
broadcast	none	nothing
getheard	none	a structure containing the stations heard and their RF level as a result of the last broadcast function issued
clearcache	none	nothing

EVENTS: None.

XML REPRESENTATION:

```
<?xml version = “1.0” encoding = “ISO-8859-1”?>
<!DOCTYPE response SYSTEM “rs_template.dtd”>
<response>
    <template name = “A7X3_V1_4”>
        <attribs>
            <attrib name = “id” type = “int” perm = “ro”/>
            <attrib name = “template” type = “string” perm = “ro”>
                <string>A7X3_V1_4</string>
            </attrib>
            <attrib name = “name” type = “string” perm = “ro”>
                <string>addWAVE</string>
            </attrib>
            <attrib name = “class” type = “string” perm = “ro”>
                <string>DEVICE</string>
            </attrib>
            <attrib name = “subclass” type = “string” perm = “ro”/>
            <attrib name = “manufacturer” type = “string” perm = “ro”>
                <string>Adcon Telemetry AG</string>
            </attrib>
            <attrib name = “type” type = “string” perm = “ro”/>
            <attrib name = “version” type = “string” perm = “ro”/>
            <attrib name = “code” type = “int” perm = “ro”/>
            <attrib name = “slotInterval” type = “int” perm = “rw”>
                <int>900</int>
            </attrib>
            <attrib name = “samplesPerSlot” type = “int” perm = “rw”>
                <int>15</int>
            </attrib>
            <attrib name = “date” type = “date” perm = “ro”/>
            <attrib name = “timeZone” type = “string” perm = “ro”/>
            <attrib name = “batteryLevel” type = “double” perm = “ro”/>
            <attrib name = “internalTemperature” type = “double” perm = “ro”/>
            <attrib name = “uptime” type = “string” perm = “ro”/>
            <attrib name = “rssi” type = “int” perm = “ro”>
                <int>42</int>
            </attrib>
            <attrib name = “pmpLow” type = “int” perm = “ro”>
                <int>65</int>
            </attrib>
            <attrib name = “pmpHigh” type = “int” perm = “ro”>
                <int>72</int>
            </attrib>
            <attrib name = “powerOutput” type = “int” perm = “ro”>
                <int>155</int>
            </attrib>
        </attribs>
        <functions>
            <function name = “broadcast”>
                <params/>
                <return type = “void”/>
            </function>
            <function name = “getheard”>
                <params/>
                <return type = “struct” descr = “A structure containing the stations heard and their RF level
                           as a result of the last broadcast function issued”/>
            </function>
            <function name = “clearcache”>
                <params/>
                <return type = “void”/>
            </function>
        </functions>
        <events/>
    </template>
</response>
```

4.2.4. A733GSM (addWAVE-GSM)

TEMPLATE NAME: A733GSM

ATTRIBUTES:

Name	Type	Perm	Default Value	Example
id	int	ro	-	12345
template	string	ro	A733GSM	
name	string	ro	addWAVE-GSM	Adcon Klosterneuburg
class	string	ro	DEVICE	
subclass	string	ro	addWAVE-GSM	
manufacturer	string	ro	Adcon Telemetry AG	
type	string	ro	A733GSM	
version	string	ro	-	ver. 2.0
code	int	ro	-	33210
phone	string	ro	-	+436641234567
slotInterval	int	rw	900	(in seconds)

Name	Type	Perm	Default Value	Example
samplesPerSlot	int	rw	15	(in units)
date	date	ro	-	20030414T12:05:33
timeZone	string	ro	-	Europe/Vienna
batteryLevel	double	ro	-	6.5 (in volt)
internalTemperature	double	ro	-	22.6 (in °C)
uptime	string	ro	-	105 days, 5 hours, 10 minutes
rssi	int	ro	42	
pmpLow	int	ro	65	(in hundred of millivolts)
pmpHigh	int	ro	72	(in hundred of millivolts)

FUNCTIONS:

Name	Parameters	Return Value
clearcache	none	nothing

EVENTS: None.

EVENTS: None.

XML REPRESENTATION:

```
<?xml version = “1.0” encoding = “ISO-8859-1”?>
<!DOCTYPE response SYSTEM “rs_template.dtd”>
<response>
    <template name = “A733GSM”>
        <attribs>
            <attrib name = “id” type = “int” perm = “ro”/>
            <attrib name = “template” type = “string” perm = “ro”>
                <string> A733GSM</string>
            </attrib>
            <attrib name = “name” type = “string” perm = “ro”>
                <string>addWAVE-GSM</string>
            </attrib>
            <attrib name = “class” type = “string” perm = “ro”>
                <string>DEVICE</string>
            </attrib>
            <attrib name = “subclass” type = “string” perm = “ro”>
                <string>addWAVE-GSM</string>
            </attrib>
            <attrib name = “manufacturer” type = “string” perm = “ro”>
                <string>Adcon Telemetry AG</string>
            </attrib>
            <attrib name = “type” type = “string” perm = “ro”>
                <string> A733GSM</string>
            </attrib>
            <attrib name = “version” type = “string” perm = “ro”/>
            <attrib name = “code” type = “int” perm = “ro”/>
            <attrib name = “phone” type = “string” perm = “ro”/>
            <attrib name = “slotInterval” type = “int” perm = “rw”>
                <int>900</int>
            </attrib>
            <attrib name = “samplesPerSlot” type = “int” perm = “rw”>
                <int>15</int>
            </attrib>
            <attrib name = “date” type = “date” perm = “ro”/>
            <attrib name = “timeZone” type = “string” perm = “ro”/>
            <attrib name = “batteryLevel” type = “double” perm = “ro”/>
            <attrib name = “internalTemperature” type = “double” perm = “ro”/>
            <attrib name = “uptime” type = “string” perm = “ro”/>
            <attrib name = “rssi” type = “int” perm = “ro”>
                <int>42</int>
            </attrib>
            <attrib name = “pmpLow” type = “int” perm = “ro”>
                <int>65</int>
            </attrib>
            <attrib name = “pmpHigh” type = “int” perm = “ro”>
                <int>72</int>
            </attrib>
        </attribs>
        <functions>
            <function name = “clearcache”>
                <params/>
                <return type = “void”/>
            </function>
        </functions>
        <events/>
    </template>
</response>
```

4.2.5. A730MD

TEMPLATE NAME: A730MD

ATTRIBUTES:

Name	Type	Perm	Default Value	Example
id	int	ro	-	4444
template	string	ro	A730MD	
name	string	ro	Adcon RTU	Nappa Valley
class	string	ro	DEVICE	
subclass	string	ro	A730MD	
manufacturer	string	ro	Adcon Telemetry AG	
type	string	ro	A730MD	
version	string	ro	-	ver. C6
code	int	ro	-	2345
date	date	ro	-	20000714T12:05:33
timeZone	string	ro	-	Europe/Vienna

FUNCTIONS: None.

EVENTS: None.

XML REPRESENTATION:

```
<?xml version = “1.0” encoding = “ISO-8859-1”?>
<!DOCTYPE response SYSTEM “rs_template.dtd”>
<response>
    <template name = “A730MD”>
        <attribs>
            <attrib name = “id” type = “int” perm = “ro”/>
            <attrib name = “template” type = “string” perm = “ro”>
                <string>A730MD</string>
            </attrib>
            <attrib name = “name” type = “string” perm = “ro”>
                <string>Adcon RTU</string>
            </attrib>
            <attrib name = “class” type = “string” perm = “ro”>
                <string>DEVICE</string>
            </attrib>
            <attrib name = “subclass” type = “string” perm = “ro”>
                <string>A730MD</string>
            </attrib>
            <attrib name = “manufacturer” type = “string” perm = “ro”>
                <string>Adcon Telemetry AG</string>
            </attrib>
            <attrib name = “type” type = “string” perm = “ro”>
                <string>A730MD</string>
            </attrib>
            <attrib name = “version” type = “string” perm = “ro”/>
            <attrib name = “code” type = “int” perm = “ro”/>
            <attrib name = “date” type = “date” perm = “ro”/>
            <attrib name = “timeZone” type = “string” perm = “ro”/>
        </attribs>
        <functions/>
        <events/>
    </template>
</response>
```

4.2.6. A840 Telemetry Gateway

TEMPLATE NAME: A840_V2

ATTRIBUTES:

Name	Type	Perm	Default Value	Example
id	int	ro	-	302
template	string	ro	A840_V2	
name	string	ro	Telemetry Gateway	Adcon Klosterneuburg
class	string	ro	SERVER	
subclass	string	ro	GATEWAY	
manufacturer	string	ro	Adcon Telemetry AG	
type	string	ro	A840	
version	string	ro	-	ver. 1.1
code	int	ro	-	15555
date	date	ro	-	20000714T12:05:33
timeZone	string	ro	-	Europe/Vienna
batteryLevel	double	ro	-	6.5 (in volt)
getdataMaxSlots	int	ro	200	
uptime	string	ro	-	105 days, 5 hours, 10 minutes

FUNCTIONS:

Name	Parameters	Return Value
broadcast	none	nothing
getheard	none	a structure containing the stations heard and their RF level as a result of the last broadcast function issued
clearcache	none	nothing

EVENTS: None.

XML REPRESENTATION:

```
<?xml version = “1.0” encoding = “ISO-8859-1”?>
<!DOCTYPE response SYSTEM “rs_template.dtd”>
<response>
    <template name = “A840_V2”>
        <attribs>
            <attrib name = “id” type = “int” perm = “ro”/>
            <attrib name = “template” type = “string” perm = “ro”>
                <string>A840_V2</string>
            </attrib>
            <attrib name = “name” type = “string” perm = “ro”>
                <string>Telemetry Gateway</string>
            </attrib>
            <attrib name = “class” type = “string” perm = “ro”>
                <string>SERVER</string>
            </attrib>
            <attrib name = “subclass” type = “string” perm = “ro”>
                <string>GATEWAY</string>
            </attrib>
            <attrib name = “manufacturer” type = “string” perm = “ro”>
                <string>Adcon Telemetry AG</string>
            </attrib>
            <attrib name = “type” type = “string” perm = “ro”>
                <string>A840</string>
            </attrib>
            <attrib name = “version” type = “string” perm = “ro”/>
            <attrib name = “code” type = “int” perm = “ro”/>
            <attrib name = “date” type = “date” perm = “ro”/>
            <attrib name = “timeZone” type = “string” perm = “ro”/>
            <attrib name = “batteryLevel” type = “double” perm = “ro”/>
            <attrib name = “getdataMaxSlots” type = “int” perm = “ro”>
                <int>200</int>
            </attrib>
            <attrib name = “uptime” type = “string” perm = “ro”/>
        </attribs>
        <functions>
            <function name = “broadcast”>
                <params/>
                <return type = “void”/>
            </function>
            <function name = “getheard”>
                <params/>
                <return type = “struct” descr = “A structure containing the stations heard and their RF level
                           as a result of the last broadcast function issued”/>
            </function>
            <function name = “clearcache”>
                <params/>
                <return type = “void”/>
            </function>
        </functions>
        <events/>
    </template>
</response>
```

4.2.7. addVANTAGE Pro Server

TEMPLATE NAME: A4PRO_V4

ATTRIBUTES:

Name	Type	Perm	Default Value	Example
id	int	ro	-	302
template	string	ro	A4PRO_V4	
name	string	ro	A4Pro Server	Adcon Klosterneuburg
class	string	ro	SERVER	
subclass	string	ro	A4PRO	
manufacturer	string	ro	Adcon Telemetry GmbH	
type	string	ro	A4PRO	
version	string	ro	-	ver. 1.1
date	date	ro	-	20000714T12:05:33
timeZone	string	ro	-	Europe/Vienna
getdataMaxSlots	int	ro	1000	
uptime	string	ro	-	105 days, 5 hours, 10 minutes

FUNCTIONS: None.

EVENTS: None.

XML REPRESENTATION:

```
<?xml version = “1.0” encoding = “ISO-8859-1”?>
<!DOCTYPE response SYSTEM “rs_template.dtd”>
<response>
    <template name = “A4PRO_V4”>
        <attribs>
            <attrib name = “id” type = “int” perm = “ro”/>
            <attrib name = “template” type = “string” perm = “ro”>
                <string>A4PRO_V4</string>
            </attrib>
            <attrib name = “name” type = “string” perm = “ro”>
                <string>A4PRO Server</string>
            </attrib>
            <attrib name = “class” type = “string” perm = “ro”>
                <string>SERVER</string>
            </attrib>
            <attrib name = “subclass” type = “string” perm = “ro”>
                <string>A4PRO</string>
            </attrib>
            <attrib name = “manufacturer” type = “string” perm = “ro”>
                <string>Adcon Telemetry AG</string>
            </attrib>
            <attrib name = “type” type = “string” perm = “ro”>
                <string>A4PRO</string>
            </attrib>
            <attrib name = “version” type = “string” perm = “ro”/>
            <attrib name = “date” type = “date” perm = “ro”/>
            <attrib name = “timeZone” type = “string” perm = “ro”/>
            <attrib name = “getdataMaxSlots” type = “int” perm = “ro”>
```

```
                <int>1000</int>
            </attrib>
            <attrib name = “uptime” type = “string” perm = “ro”/>
        </attribs>
        <functions/>
        <events/>
    </template>
</response>
```

4.2.8. *A730SD Base Station*

TEMPLATE NAME: A730SD

ATTRIBUTES:

Name	Type	Perm	Default Value	Example
id	int	ro	-	302
template	string	ro	A730SD	
name	string	ro	Base Station	Adcon Bucharest
class	string	ro	BASE	
subclass	string	ro	RECEIVER	
manufacturer	string	ro	Adcon Telemetry AG	
type	string	ro	A730SD	
version	string	ro	-	ver. 1.1
code	int	ro	-	777
date	date	ro	-	20000714T12:05:33
timeZone	string	ro	-	Europe/Bucharest
batteryLevel	double	ro	-	6.5 (in volt)
uptime	string	ro	-	105 days, 5 hours, 10 min-utes

FUNCTIONS: None.

EVENTS: None.

XML REPRESENTATION:

```
<?xml version = “1.0” encoding = “ISO-8859-1”?>
<!DOCTYPE response SYSTEM “rs_template.dtd”>
<response>
    <template name = “A730SD”>
        <attribs>
            <attrib name = “id” type = “int” perm = “ro”/>
            <attrib name = “template” type = “string” perm = “ro”>
                <string>A730SD</string>
            </attrib>
            <attrib name = “name” type = “string” perm = “ro”>
                <string>Base Station</string>
            </attrib>
            <attrib name = “class” type = “string” perm = “ro”>
                <string>BASE</string>
            </attrib>
            <attrib name = “subclass” type = “string” perm = “ro”>
                <string>RECEIVER</string>
            </attrib>
            <attrib name = “manufacturer” type = “string” perm = “ro”>
                <string>Adcon Telemetry AG</string>
            </attrib>
            <attrib name = “type” type = “string” perm = “ro”>
                <string>A730SD</string>
            </attrib>
            <attrib name = “version” type = “string” perm = “ro”/>
            <attrib name = “code” type = “int” perm = “ro”/>
            <attrib name = “date” type = “date” perm = “ro”/>
            <attrib name = “timeZone” type = “string” perm = “ro”/>
            <attrib name = “batteryLevel” type = “double” perm = “ro”/>
            <attrib name = “uptime” type = “string” perm = “ro”/>
        </attribs>
        <functions/>
        <events/>
    </template>
</response>
```

4.2.9. A720B Low Power Base Station

TEMPLATE NAME: A720B

ATTRIBUTES:

Name	Type	Perm	Default Value	Example
id	int	ro	-	302
template	string	ro	A720B	
name	string	ro	Low Power Base Station	Adcon
class	string	ro	BASE	
subclass	string	ro	RECEIVER	
manufacturer	string	ro	Adcon Telemetry AG	
type	string	ro	A720B	
version	string	ro	-	ver. 1.0
code	int	ro	-	6543
date	date	ro	-	20000714T12:05:33
timeZone	string	ro	-	Europe/Vienna
batteryLevel	double	ro	-	6.5 (in volt)
uptime	string	ro	-	105 days, 5 hours, 10 minutes

FUNCTIONS: None.

EVENTS: None.

XML REPRESENTATION:

```
<?xml version = “1.0” encoding = “ISO-8859-1”?>
<!DOCTYPE response SYSTEM “rs_template.dtd”>
<response>
```

```
    <template name = “A720B”>
        <attribs>
            <attrib name = “id” type = “int” perm = “ro”/>
            <attrib name = “template” type = “string” perm = “ro”>
                <string>A720B</string>
            </attrib>
            <attrib name = “name” type = “string” perm = “ro”>
                <string>Low Power Base Station</string>
            </attrib>
            <attrib name = “class” type = “string” perm = “ro”>
                <string>BASE</string>
            </attrib>
            <attrib name = “subclass” type = “string” perm = “ro”>
                <string>RECEIVER</string>
            </attrib>
            <attrib name = “manufacturer” type = “string” perm = “ro”>
                <string>Adcon Telemetry AG</string>
            </attrib>
            <attrib name = “type” type = “string” perm = “ro”>
                <string>A720B</string>
            </attrib>
            <attrib name = “version” type = “string” perm = “ro”/>
            <attrib name = “code” type = “int” perm = “ro”/>
            <attrib name = “date” type = “date” perm = “ro”/>
            <attrib name = “timeZone” type = “string” perm = “ro”/>
            <attrib name = “batteryLevel” type = “double” perm = “ro”/>
            <attrib name = “uptime” type = “string” perm = “ro”/>
        </attribs>
        <functions/>
        <events/>
    </template>
</response>
```

4.2.10. A440 Wireless Modem

TEMPLATE NAME: A440_V1_4

ATTRIBUTES:

Name	Type	Perm	Default Value	Example
id	int	ro	-	302
template	string	ro	A440_V1_4	
name	string	ro	Wireless Modem	Adcon Boca Raton
class	string	ro	BASE	
subclass	string	ro	RECEIVER	
manufacturer	string	ro	Adcon Telemetry AG	
type	string	ro	A440	
version	string	ro	-	ver. 1.5
code	int	ro	-	17500
date	date	ro	-	20010714T12:05:33
timeZone	string	ro	-	America/New_York
uptime	string	ro	-	105 days, 5 hours, 10 minutes
powerOutput	int	ro	155	

FUNCTIONS:

Name	Parameters	Return Value
broadcast	none	nothing
getheard	none	a structure containing the stations heard and their RF level as a result of the last broadcast function issued

EVENTS: None.

XML REPRESENTATION:

```
<?xml version = “1.0” encoding = “ISO-8859-1”?>
<!DOCTYPE response SYSTEM “rs_template.dtd”>
<response>
    <template name = “A440_V1_4”>
        <attribs>
            <attrib name = “id” type = “int” perm = “ro”/>
            <attrib name = “template” type = “string” perm = “ro”>
                <string>A440_V1_4</string>
            </attrib>
            <attrib name = “name” type = “string” perm = “ro”>
                <string>Wireless Modem</string>
            </attrib>
            <attrib name = “class” type = “string” perm = “ro”>
                <string>BASE</string>
            </attrib>
            <attrib name = “subclass” type = “string” perm = “ro”>
                <string>RECEIVER</string>
            </attrib>
            <attrib name = “manufacturer” type = “string” perm = “ro”>
                <string>Adcon Telemetry AG</string>
            </attrib>
            <attrib name = “type” type = “string” perm = “ro”>
                <string>A440</string>
            </attrib>
            <attrib name = “version” type = “string” perm = “ro”/>
            <attrib name = “code” type = “int” perm = “ro”/>
            <attrib name = “date” type = “date” perm = “ro”/>
            <attrib name = “timeZone” type = “string” perm = “ro”/>
            <attrib name = “uptime” type = “string” perm = “ro”/>
            <attrib name = “powerOutput” type = “int” perm = “ro”>
                <int>155</int>
            </attrib>
        </attribs>
        <functions>
            <function name = “broadcast”>
                <params/>
                <return type = “void”/>
            </function>
            <function name = “getheard”>
                <params/>
                <return type = “struct” descr = “A structure containing the stations heard and their RF level
                           as a result of the last broadcast function issued”/>
            </function>
        </functions>
        <events/>
    </template>
</response>
```

4.2.11. A723 and A733 Digital Tag

TEMPLATE NAME: DIGITAL_TAG_A7X3_V1_4

Templates

ATTRIBUTES:

Name	Type	Perm	Default Value	Example
id	int	ro	-	34567
template	string	ro	DIGITAL_TAG_A7X3_V1_4	
name	string	ro	-	Pump Switch
class	string	ro	TAG	
subclass	string	ro	-	Digital
manufacturer	string	ro	-	ACME Inc.
type	string	ro	-	XYZ2001-12
EUID	int	ro	-	7000
isOutput	bool	rw	FALSE	

FUNCTIONS:

Name	Parameters	Return Value
readValue	none	the tag value (boolean)
writeValue	newValue (boolean)	nothing
xorValue	none	nothing
monostableOff	the OFF and ON times	nothing
monostableOn	the ON and OFF times	nothing
multivibratorOff	the OFF and ON times	nothing
multivibratorOn	the ON and OFF times	nothing

EVENTS:

Name	Description
notificationOnPortChange	Issues a notification when an input port changed state

XML REPRESENTATION:

```
<?xml version = “1.0” encoding = “ISO-8859-1”?>
<!DOCTYPE response SYSTEM “rs_template.dtd”>
<response>
    <template name = “DIGITAL_TAG_A7X3_V1_4”>
        <attribs>
            <attrib name = “id” type = “int” perm = “ro”/>
            <attrib name = “template” type = “string” perm = “ro”>
                <string>DIGITAL_TAG_A7X3_V1_4</string>
            </attrib>
            <attrib name = “name” type = “string” perm = “ro”/>
            <attrib name = “class” type = “string” perm = “ro”>
                <string>TAG</string>
            </attrib>
            <attrib name = “subclass” type = “string” perm = “ro”/>
            <attrib name = “manufacturer” type = “string” perm = “ro”/>
            <attrib name = “type” type = “string” perm = “ro”/>
            <attrib name = “EUID” type = “int” perm = “ro”/>
            <attrib name = “isOutput” type = “boolean” perm = “rw”>
                <boolean>false</boolean>
            </attrib>
        </attribs>
        <functions>
            <function name = “readValue”>
                <params/>
                <return type = “boolean”/>
            </function>
            <function name = “writeValue”>
                <params>
                    <param name = “newValue” type = “boolean”/>
                </params>
                <return type = “void”/>
            </function>
            <function name = “xorValue”>
                <params/>
                <return type = “void”/>
            </function>
            <function name = “monostableOff”>
                <params>
                    <param name = “offTime” type = “int”/>
                    <param name = “onTime” type = “int”/>
                </params>
                <return type = “void”/>
            </function>
            <function name = “monostableOn”>
                <params>
                    <param name = “onTime” type = “int”/>
                    <param name = “offTime” type = “int”/>
                </params>
                <return type = “void”/>
            </function>
            <function name = “multivibratorOff”>
                <params>
                    <param name = “offTime” type = “int”/>
                    <param name = “onTime” type = “int”/>
                </params>
                <return type = “void”/>
            </function>
            <function name = “multivibratorOn”>
                <params>
                    <param name = “onTime” type = “int”/>
                    <param name = “offTime” type = “int”/>
                </params>
                <return type = “void”/>
            </function>
        </functions>
        <events>
            <event id = “0” name = “notificationOnPortChange” descr = “Issues a notification when an input
                           port changed state”/>
        </events>
    </template>
</response>
```

4.2.12. A723 and A733 Analog Tag

TEMPLATE NAME: ANALOG_TAG_A7X3_V1_4

ATTRIBUTES:

Name	Type	Perm	Default Value	Example
id	int	ro	-	123
template	string	ro	ANALOG_TAG_A7X3_V1_4	

Name	Type	Perm	Default Value	Example
name	string	ro	-	Temp 2m
class	string	ro	TAG	
subclass	string	ro	-	Temp
manufacturer	string	ro	-	Vaisala Oy
type	string	ro	-	YA-50
EUID	int	ro	-	3500
minValue	double	ro	-	-20
maxValue	double	ro	-	60
samplingMethod	int	rw	-	3

FUNCTIONS:

Name	Parameters	Return Value
setNotificationParameters	the two thresholds and the type (upper/lower or inside/outside limits)	nothing
getNotificationParameters	none	the two thresholds and the type (upper/lower or inside/outside limits)

EVENTS:

Name	Description
notificationOnThresholdReached	Issues a notification when an analog event occured on an analog channel (see also analog function calls)

XML REPRESENTATION:

```
<?xml version = “1.0” encoding = “ISO-8859-1”?>
<!DOCTYPE response SYSTEM “rs_template.dtd”>
<response>
    <template name = “ANALOG_TAG_A7X3_V1_4”>
        <attribs>
            <attrib name = “id” type = “int” perm = “ro”/>
            <attrib name = “template” type = “string” perm = “ro”>
                <string>ANALOG_TAG_A7X3_V1_4</string>
            </attrib>
            <attrib name = “name” type = “string” perm = “ro”/>
            <attrib name = “class” type = “string” perm = “ro”>
                <string>TAG</string>
            </attrib>
            <attrib name = “subclass” type = “string” perm = “ro”/>
            <attrib name = “manufacturer” type = “string” perm = “ro”/>
            <attrib name = “type” type = “string” perm = “ro”/>
            <attrib name = “EUID” type = “int” perm = “ro”/>
            <attrib name = “minValue” type = “double” perm = “ro”/>
            <attrib name = “maxValue” type = “double” perm = “ro”/>
            <attrib name = “samplingMethod” type = “int” perm = “rw”/>
        </attribs>
        <functions>
            <function name = “setNotificationAndParameters”>
                <params>
                    <param name = “type” type = “int”/>
                    <param name = “limit1” type = “int”/>
                    <param name = “limit2” type = “int”/>
                </params>
                <return type = “void”/>
            </function>
            <function name = “getNotificationAndParameters”>
                <params/>
                <return type = “array” descr = “NotificationType, Limit1, Limit2”/>
            </function>
        </functions>
        <events>
            <event id = “1” name = “notificationOnThresholdReached” descr = “Issues a notification when an
                           analog event occured on an analog channel”/>
        </events>
    </template>
</response>
```

4.2.13. *Generic Digital Tag*

TEMPLATE NAME: DIGITAL_TAG

ATTRIBUTES:

Name	Type	Perm	Default Value	Example
id	int	ro	-	34567
template	string	ro	DIGITAL_TAG	
name	string	ro	-	Pump Switch
class	string	ro	TAG	
subclass	string	ro	-	Digital
manufacturer	string	ro	-	Adcon Telemetry AG
type	string	ro	-	A501
EUID	int	ro	-	7000

FUNCTIONS:

Name	Parameters	Return Value
writeValue	newValue (boolean)	nothing

EVENTS: None.

XML REPRESENTATION:

```
<?xml version = “1.0” encoding = “ISO-8859-1”?>
<!DOCTYPE response SYSTEM “rs_template.dtd”>
<response>
    <template name = “DIGITAL_TAG”>
        <attribs>
            <attrib name = “id” type = “int” perm = “ro”/>
            <attrib name = “template” type = “string” perm = “ro”>
                <string>DIGITAL_TAG</string>
            </attrib>
            <attrib name = “name” type = “string” perm = “ro”/>
            <attrib name = “class” type = “string” perm = “ro”>
                <string>TAG</string>
            </attrib>
            <attrib name = “subclass” type = “string” perm = “ro”/>
            <attrib name = “manufacturer” type = “string” perm = “ro”/>
            <attrib name = “EUID” type = “int” perm = “ro”/>
        </attribs>
        <functions>
            <function name = “writeValue”>
                <params>
                    <param name = “newValue” type = “boolean”/>
                </params>
                <return type = “void”/>
            </function>
        </functions>
        <events/>
    </template>
</response>
```

4.2.14. *Generic Analog Tag*

TEMPLATE NAME: ANALOG_TAG

ATTRIBUTES:

Name	Type	Perm	Default Value	Example
id	int	ro	-	7777
template	string	ro	ANALOG_TAG	
name	string	ro	-	Battery
class	string	ro	TAG	
subclass	string	ro	-	Batt
manufacturer	string	ro	-	Adcon Telemetry AG
type	string	ro	-	BATT-1200
EUID	int	ro	-	3500
minValue	double	ro	-	0
maxValue	double	ro	-	20

FUNCTIONS: None

EVENTS: None.

XML REPRESENTATION:

```
<?xml version = “1.0” encoding = “ISO-8859-1”?>
<!DOCTYPE response SYSTEM “rs_template.dtd”>
<response>
    <template name = “ANALOG_TAG”>
        <attribs>
            <attrib name = “id” type = “int” perm = “ro”/>
            <attrib name = “template” type = “string” perm = “ro”>
                <string>ANALOG_TAG</string>
            </attrib>
            <attrib name = “name” type = “string” perm = “ro”/>
            <attrib name = “class” type = “string” perm = “ro”>
                <string>TAG</string>
            </attrib>
            <attrib name = “subclass” type = “string” perm = “ro”/>
            <attrib name = “manufacturer” type = “string” perm = “ro”/>
            <attrib name = “type” type = “string” perm = “ro”/>
            <attrib name = “EUID” type = “int” perm = “ro”/>
            <attrib name = “minValue” type = “double” perm = “ro”/>
            <attrib name = “maxValue” type = “double” perm = “ro”/>
        </attribs>
        <functions/>
        <events/>
    </template>
</response>
```

4.2.15. SDI-12 Analog Tag

TEMPLATE NAME: ANALOG_TAG_SDI12

ATTRIBUTES:

Name	Type	Perm	Default Value	Example
id	int	ro	-	9999
template	string	ro	ANALOG_TAG_SDI12	
name	string	ro	-	C-Probe 10 cm
class	string	ro	TAG	
subclass	string	ro	-	C-Probe SMC2.0201
manufacturer	string	ro	-	C-Probe Corporation
type	string	ro	-	SMC2.0
EUID	int	ro	-	5301
minValue	double	ro	-	0
maxValue	double	ro	-	100

FUNCTIONS:

Name	Parameters	Return Value
setNotificationParameters	the two thresholds and the type (upper/lower or inside/outside limits)	nothing
getNotificationParameters	none	the two thresholds and the type (upper/lower or inside/outside limits)

EVENTS:

Name	Description
notificationOnThresholdReached	Issues a notification when an analog event occured on an analog channel (see also analog function calls)

XML REPRESENTATION:

```
<?xml version = “1.0” encoding = “ISO-8859-1”?>
<!DOCTYPE response SYSTEM “rs_template.dtd”>
<response>
```

```
    <template name = “ANALOG_TAG_SDI12”>
        <attribs>
            <attrib name = “id” type = “int” perm = “ro”/>
            <attrib name = “template” type = “string” perm = “ro”>
                <string>ANALOG_TAG_SDI12</string>
            </attrib>
            <attrib name = “name” type = “string” perm = “ro”/>
            <attrib name = “class” type = “string” perm = “ro”>
                <string>TAG</string>
            </attrib>
            <attrib name = “subclass” type = “string” perm = “ro”/>
            <attrib name = “manufacturer” type = “string” perm = “ro”/>
            <attrib name = “type” type = “string” perm = “ro”/>
            <attrib name = “EUID” type = “int” perm = “ro”/>
            <attrib name = “minValue” type = “double” perm = “ro”/>
            <attrib name = “maxValue” type = “double” perm = “ro”/>
        </attribs>
        <functions>
            <function name = “setNotificationAndParameters”>
                <params>
                    <param name = “type” type = “int”/>
                    <param name = “limit1” type = “int”/>
                    <param name = “limit2” type = “int”/>
                </params>
                <return type = “void”/>
            </function>
            <function name = “getNotificationAndParameters”>
                <params/>
                <return type = “array” descr = “NotificationType, Limit1, Limit2”/>
            </function>
        </functions>
        <events>
            <event id = “1” name = “notificationOnThresholdReached” descr = “Issues a notification when an
                           analog event occured on an analog channel”/>
        </events>
    </template>
</response>
```

4.3. Known Tags

The following table contains in a compressed form all the tags currently in use on the A72x and A73x series of devices (non-intelligent sensors).

Name (default)	Type	Manufacturer	Type	EUID	minValue	maxValue
Air Pressure	AIRPRESS	Keller AG	PA41	1811	750	1050
Battery	BATT	Adcon Telemetry AG	AA for addITs, C for A730MD and A733	3500	0	20
Global Radiation	GR	Mesa Systems GmbH	MS-020VM	5800	0	1400
Precipitation	RAIN	Pronamic	Rain-O-Matic	2601	0	51
Relative Humidity	RH	Vaisala Oy	50YA	5400	0	100
Soil Temperature	SOILTEMP	Rössler AG	99TG411E	2710	-20	40
Air Temperature	TEMP	Vasiala Oy	50YA	2710	-40	60
Soil Moisture	TENSIOMETER	Tensio-Technic GmbH	1208-STECK16	1803	0	200

Name (default)	Type	Manufacturer	Type	EUID	minValue	maxValue
Soil Moisture	VIRRIB	Amet	VIRRIB	5300	0	50
Soil Moisture	WATERMARK	Irrometer, Inc.	WATERMARK	1803	0	200
Wind Direction	WINDDIR	Schiltknecht AG	566.2S-130	310	0	360
Wind Speed	WINDSPEED	Schiltknecht AG	566.2S-130	601	0	100
Water Level	WLEVEL5	Keller AG	PR36W	3	0	500
Water Level	WLEVEL10	Keller AG	PR36W	3	0	1000
Leaf Wetness	LW	Adcon Telemetry AG	A730WET	5500	0	10
Soil Moisture	WMRAW1	Irrometer, Inc.	WATERMARK	1803	0	200
Soil Moisture	WMRAW2	Irrometer, Inc.	WATERMARK	1803	0	200
Soil Moisture	WMRAW3	Irrometer, Inc.	WATERMARK	1803	0	200
Soil Moisture	CPROBE1	Agrilink Pty	C-Probe	5301	0	100
Soil Moisture	CPROBE2	Agrilink Pty	C-Probe	5301	0	100
Soil Moisture	CPROBE3	Agrilink Pty	C-Probe	5301	0	100
RF Level	RF	Adcon Telemetry AG	(depending on unit, A720, A730MD, etc....)	3502	0	10
Incoming RF Level	RFIN	Adcon Telemetry AG	(depending on unit, A720, A730MD, etc....)	3502	0	10
Outgoing RF Level	RFOUT	Adcon Telemetry AG	(depending on unit, A720, A730MD, etc....)	3502	0	10
Solar Panel	DIGITAL	Adcon Telemetry AG	none	7000	-	-
Soil Moisture	GB1	Waterwise	WATERWISE	5600	0	100
Soil Moisture	GB2	Waterwise	WATERWISE	5600	0	100
Soil Moisture	GB3	Waterwise	WATERWISE	5600	0	100