TOA4/MW Remote Procedure Call (RPC) API

Introduction

The purpose of the TOA4 RPC API is to allow external software applications to exchange data with TOA4 and activate certain TOA4 functions. This document describes the portion of that API for exchanging data with TOA4 Monitor Watch™ (TOA4/MW), an optional extension of TOA4 Online for managing and interpreting online monitor data. **Monitors have to be registered** in TOA4/MW before data for them can be transmitted to TOA4/MW.

The TOA4 RPC API is a simple HTTP-based [HT1, HT2] command interface, with each command represented by a URI. For security, TOA4 RPC should always be accessed using the HTTP Secure protocol (https) [TL1, TL2, TL3].

Data Format

Monitor data uploaded to TOA4/MW is expected to be comma-separated value (.csv) ASCII or UTF8-encoded text, with carriage return - line feed (0x0C 0x0A) separating the rows of text. Here is an example, with the line terminators not shown:

```
monitor_id,sampledate,fluidtempc,airtemp,ch4,c2h4,c2h2,relsaturation
HORS-T1-MAIN,2014-04-03 10:54:46,39,12,24.1,3.4,0.0,10.2
SAAN-T2-DIV,2014-04-03 10:57:10,31,,210.1,356.8,44.2,27.6
```

The **first text row** must consist of comma-separated **column names** recognized by TOA4. The column names are not case-sensitive and can be in any order.

Each text row after the first consists of comma-separated data values in the same order as the corresponding column names. A **missing data value** is represented by an **empty string**, i.e., no characters. Non-numeric data items may optionally be enclosed in double quotes ("). Embedded commas, tabs, line feeds, and special symbols such as degree symbols are not permitted.

Each data row represents one monitor sample record. The **monitor_id**, a customer-assigned text identifier for an individual monitor, is mandatory, as is the **sampledate**, a timestamp in "yyyy-mm-dd hh:mm:ss" format indicating when the sample was collected by the monitor. Numeric data fields commonly used include:

> fluidtempc -- tank oil temperature, preferably top oil temperature, degrees C
> airtemp -- external air temperature, degrees C
> h2, ch4, c2h6, c2h4, c2h2, co, co2, o2, n2 -- gas concentrations, microliters per liter (ppm)
> relsaturation -- % relative saturation of water in the insulating fluid, at the tank temperature
> water -- water concentration, microliters per liter (ppm)

Numeric values can optionally have fractional parts. For example, 15 or 15.3 would be acceptable as a gas concentration. We do not guarantee to use or to preserve any or all of the fractional part for variables with low measurement precision.

Other fields, representing additional temperatures, kiloamperes of load current, and other sensor inputs, may also be supported. Please contact *support@deltaxresearch.com* for information.

Sample records for any particular monitor are accepted only **in chronological order**, i.e., any incoming sample record with a sampledate less than or equal to that of the most recent sample record already accepted by TOA4/MW is rejected.

Protocol

In all cases, the procedure for using a TOA4 RPC command is for the client application to issue an **HTTP request** to a URI similar to the following, where the last element of the URI is the RPC command name:

```
https://www.toa4online.com/toa/rpc/append_monitor_data
```

The server address, or **hostname** portion of the URI, depends on which TOA4 Online server is being addressed. The USA server is *www.toa4online.com*, the Canadian/International server is *ca.toa4.com*, and the test server is *test.toa4online.com*.

Some RPC commands accept **query data** to provide arguments which modify the scope or effect of the command. Regarding HTTP requests, "query data" refers to attribute-value expressions appended to the command URI following a question mark, as in this example:

```
https://www.toa4online.com/toa/rpc/delete_eqp?apprtype=TRN&equipnum=1234
```

The HTTP **request method** (GET or POST) required in each case is specified with the description of each RPC command below. Generally, commands that retrieve information use the GET method, and commands that insert or modify data use the POST method.

Unless specifically indicated otherwise, all RPC HTTP requests must be **simple**, not multi-part.

All RPC requests directed to TOA4 require an **authentication header** for the HTTP Basic user authentication protocol (see the related Appendix below). The RPC login ID and password associated with the command are encrypted in the authentication header. The RPC login ID must be assigned the "rpc_user" **security role** in a TOA4 database in order for TOA4 to accept RPC commands associated with that login ID in that database.

The response to a TOA4 RPC request is an **HTTP response** which, depending on the command, may contain a plain-text status message, a block of .csv data, or a document. The external application which sent the RPC request is responsible for receiving the response and retrieving and interpreting the information in the body of the response.

The Appendix to this document indicates sources of detailed information on HTTP and provides examples of a TOA4 RPC HTTP authentication header and request and response messages.

TOA4/MW RPC Commands

Testing

welcome (GET) - Return text message with RPC login ID and database ID.

Example output: `Welcome 'someco-mwrpc1'.  You are using the 'SOMECO' database.`

Arguments: None

Exceptions: None

timestamp (GET) – Return a plain-text UTC timestamp from the TOA4 server in yyyymmddThhmmssZ format. Example: `20140403T180205Z`.

Arguments: None

Exceptions: None

Please note -- These commands are not to be used for high-frequency pinging of the server. Excessive use of the "welcome" or the "timestamp" RPC may result in blockage of access without warning.

Upload Data

append_monitor_data (POST) - Add new monitor sample records to the database.

Arguments: monitor_id, sampledate (both required)

Request data: A block of .csv data, with field names in the first row.

Response: The response body is a plain-text message reporting the number of new records appended.

Errors: Sample records that are out of chronological sequence with previously accepted data for the same monitor_id are rejected. Records with missing or unrecognized monitor_id and sampledate are rejected.

Reporting

monitor_results (GET) – Return .csv data summarizing latest analysis results for all monitors. If a monitor_id argument is provided, only the results for the designated monitor are returned.

Arguments: monitor_id (optional)

report/monitor_status (GET) - Return the latest status report for the specified monitor, as a PDF document.

Arguments: monitor_id (required)

Appendix: HTTP Examples and Resources

cURL Software for testing HTTP transactions

An open source software application called cURL (see [C1, C3]) is useful for testing RPC commands and for inspecting the HTTP requests and responses related to them. The cURL software is available via free download for Windows, Mac OS, and Linux.

Detailed cURL documentation is provided online [C2], but for convenience here are some examples of how to run TOA4 RPC transactions from a command line using cURL. Substitutions for items such as RPC login ID and password are needed as indicated in the examples by underlining. The command must be typed with no line breaks. In these examples the command is wrapped for display only. The command URL, in double quotes, should be at the end.

If you include a -v argument, cURL echoes the HTTP request sent and the HTTP response received. This is very useful for debugging or for getting examples for learning how to construct requests and parse responses.

cURL Example 1. Run the "welcome" command.

```
> curl -u rpcloginid:rpcpassword "https://www.toa4online.com/toa/rpc/welcome"
```

cURL Example 2. Data upload. Import new laboratory data into TOA4 Online from a csv data file. The data file path must be preceded by an "@" symbol as shown. The POST method is needed because this inserts or modifies data. The header specifies that the file contains text, even though the "data-binary" argument is telling cURL to send everything in the file as-is.

```
> curl -X POST --header "Content-Type: text/plain" --data-binary
@filepath/filename.csv -u rpcloginid:rpcpassword
"https://www.toa4online.com/toa/rpc/append_monitor_data"
```

cURL Example 3. Download monitor results to a file.

```
> curl -u rpcloginid:rpcpassword -o filepath/filename.csv
"https://www.toa4online.com/toa/rpc/monitor_results"
```

cURL Example 4. Download a monitor status report.

```
> curl -X POST -u rpcloginid:rpcpassword  -o filepath/myreport.pdf
"https://www.toa4online.com/toa/rpc/report/monitor_status?monitor_id=CRUM-T1-MAIN"
```

Note: If cURL running in Microsoft Windows has a problem with recognition of the security certification authority (CA) for https, add a -k argument to suppress CA checking. **Do not revert to using the insecure http protocol for RPC transactions!** There is no problem with the TOA4 security certificates. The certification authority for all public TOA4 web sites is Go Daddy Secure Certification Authority, a well-known and widely used CA.

HTTP Protocol Documentation

The HTTP/1.1 protocol, including the detailed specifications for HTTP requests and responses, is documented in RFC2616 [HT1], which is freely downloadable. The Wikipedia article "Hypertext Transfer Protocol" [HT2] contains a good overview of the protocol, examples of request and response messages, and links to related material.

HTTP Basic Authentication

The HTTP "Basic" authentication scheme, used by the TOA4 RPC API, is documented in RFC 2617 [AU1]. The Wikipedia article "Basic access authentication" [AU2] contains discussion and examples.

Every RPC request sent to TOA4 should contain an HTTP Authorization header to avoid receiving a 401 (Unauthorized) response from TOA4. The appearance and structure of such a header is explained by the following example. For the example, assume that the RPC login ID is "someco-rpc1" and the password is "c0nfus1ng".

The first step in constructing the header is to concatenate the RPC login ID, a colon, and the password:

```
someco-rpc1:c0nfus1ng
```

Now that text string is encoded using standard "base64" encoding:

```
c29tZWNvLXJwYzE6YzBuZnVzMW5n
```

The complete header looks like this, where there is a single space character separating the word "Basic" and the encoded RPC login ID and password:

```
Authorization: Basic c29tZWNvLXJwYzE6YzBuZnVzMW5n
```

Note that since the RPC login ID and password are known, the whole header can be generated in advance and stored for use each time an RPC request is sent to TOA4.

When sending RPC HTTP requests to TOA4 via the public Internet, the client should use the secure https protocol [TL1, TL2, TL3, TL4] for data and password security. With https, all traffic between the client and the TOA4 server is strongly encrypted, including both the authorization header and the data. The organization and content of RPC HTTP requests and responses are not changed in any way for https.

Please note that the TOA4 **security certificates** are assigned to the **server hostname**, not to the server IP address, so attempts to communicate via https with TOA4/MW using the server IP address instead of the hostname will fail. Note also that Delta-X Research Inc. does not guarantee that the server IP address will be the same from day to day and does not provide advance notice of server IP address changes.

References

[HT1] W3C, "Hypertext Transfer Protocol -- HTTP/1.1," RFC2616, ftp://ftp.isi.edu/in-notes/rfc2616.txt (2009-11-09).

[HT2] Wikipedia, "Hypertext Transfer Protocol," http://en.wikipedia.org/wiki/HTTP (2009-11-09).

[TL3] Wikipedia, "HTTP Secure," http://en.wikipedia.org/wiki/HTTP_Secure (2009-11-09).

[TL4] W3C, "HTTP Over TLS," RFC2818, http://tools.ietf.org/rfc/rfc2818.txt (2009-11-09).

[AU1] W3C, "HTTP Authentication: Basic and Digest Access Authentication," RFC2617, ftp://ftp.isi.edu/in-notes/rfc2617.txt (2009-11-09).

[AU2] Wikipedia, "Basic access authentication," http://en.wikipedia.org/wiki/Basic_access_authentication (2009-11-09).

[TL1] W3C, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC5246, http://tools.ietf.org/html/rfc5246.txt (2009-11-09).

[TL2] Wikipedia, "Transport Layer Security," http://en.wikipedia.org/wiki/Transport_Layer_Security (2009-11-09).

[C1] cURL web site, http://curl.haxx.se/ (2010-04-15).

[C2] cURL user documentation, http://curl.haxx.se/docs/manpage.html (2010-04-15).

[C3] Wikipedia, "CURL," http://en.wikipedia.org/wiki/CURL (2010-04-15).