API Documentation

TOA Remote Procedure Call API

1 Introduction

The purpose of the TOA RPC API is to allow external software applications to exchange data with TOA and activate certain TOA functions.

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

Access to a TOA database via TOA RPC is available as an add-on which must be purchased from Delta-X Research by the TOA database owner. The price varies depending on the type and volume of expected usage. Fixed prices are provided for RPC access by approved third-party applications.

2 Protocol

In all cases, the procedure for using a TOA 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_test_data**

The prefix "https" indicates that a Secure HTTP connection is to be used, which employs standards-based encryption to protect login credentials and data going both ways. The server address, or **hostname** portion of the URI, depends on which TOA 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 TOA 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 TOA database in order for TOA to accept RPC commands associated with that login ID in that database.

The response to a TOA 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.

Comma-separated value data (CSV) is formatted according to RFC 4180 [C4]. In both the HTTP request and response, the character encoding of textual data is indicated by the "charset" parameter of the "Content-Type" header. If not specified, the default character set is ISO-8859-1. To avoid character encoding issues, it is recommended that the UTF-8 encoding be used. When CSV data is sent to the TOA RPC interface, it is highly recommended that a Content-Type header be provided as part of the HTTP request headers. E.g.

Content-Type: text/csv; charset=utf-8

The value of the "charset" parameter must match the character encoding of CSV data. If set incorrectly, corruption of data can occur. The correct encoding to use will depend on the system that created the CSV data. It is recommended that the UTF-8 encoding be used for text if possible. It is the most widely used Unicode compatible encoding and is efficient for data containing mostly ASCII characters. Other valid character encodings are: Latin_1, ISO-8859-1, UTF-16BE, UTF-16LE, Windows-1252 and many more. Contact Delta-X Research to confirm if a specific encoding is supported.

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

3 RPC Commands

TOA RPC commands are grouped as follows:

- Connection testing
- Context information
- Equipment information
- TOA test data for manually-collected samples
- TOA analysis results
- TOA Monitor Watch^{TM} online monitor data
- TOA Monitor Watch^{TM} analysis results

Each of the command descriptions below shows the portion of the command URI that follows the hostname and describes any query parameters that may be appended to the command.

3.1 Connection Testing

/toa/null (GET) - Test whether TOA server is reachable and online. The response contains a single word ("okay") of plain text. Note that "rpc" is not included in the URI path.

Example output: okay

Arguments: None

Exceptions: None

Note: This is the only command that is acceptable for status "pinging" when the client needs to test whether the TOA server is reachable. Repeated pinging using RPC commands such as "welcome" and "timestamp" is not allowed and may result in denial of access.

3.2 Context Information

/toa/rpc/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 TOA server in

yyyymmddThhmmssZ format.
Example: 20140403T180205Z.

Arguments: 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.

3.3 Equipment information

/toa/rpc/write_eqp (POST) - Create or update equipment information

Arguments: None

Action: Import equipment information from CSV records given in the body of the request. For each data record in the file, create or update a equipment object in the TOA database. Individual equipment objects are identified by either equipnum+apprtype or serialnum+apprtype. The response body is a plain-text message reporting the number of existing equipment objects which were create and updated.

Request data: A block of CSV data. For CSV data format requirements, see "Equipment Information Requirements" (https://www.toa4online.com/toa/help/equipment_info)

Response: The response body is a plain-text message reporting the number of new equipment objects create and updated.

Special cases: In cases where there is already an equipment object in the database with matching identification, update that object's fields with non-null values from the CSV data record.

Errors: If an input equipment record contains a group_name value which does not match the identifier of any access control group to which the RPC login ID belongs, an "equipment access denied" error occurs, and the record is rejected.

/toa/rpc/export_eqp_key (GET) - Return the internal TOA database identifier ('eqp_id') for one or more equipment objects. This is useful for an external application which needs to construct a URI for a TOA user interface page related to a specific equipment item. Note that the key value for an equipment item may change if the item is exported from TOA and then re-imported. This command returns the following as CSV data: equipnum, serialnum, apprtype, external_id, and eqp_id. The same filter arguments as are used by 'export_eqp' (see below) can be used to restrict the scope of 'export_eqp_key'.

/toa/rpc/export_eqp (GET) - Export equipment information in CSV format. The response body is a block of CSV records.

The returned items are filtered according to arguments presented in the query part of the command URI.

Arguments:
- **apprtype:** Apparatus type of equipment
- **equipnum:** Equipment number of equipment
- **serialnum:** Serial number of equipment
- **owner_name:** Owner identifier
- **region_name:** Region identifier
- **substn_name:** Substation identifier
- **posted_since:** UTC timestamp (yyyymmddThhmmssZ)
- **changed_since:** UTC timestamp (yyyymmddThhmmssZ)

Format of CSV data returned:
See "Equipment Information Requirements" (https://www.toa4online.com/toa/help/equipment_info)

/toa/rpc/delete_eqp (POST) - Delete one equipment item from the TOA database, along with all test data belonging to the deleted equipment. The query portion of the command URI must specify a value for the apprtype and also for either the equipnum or the serialnum, uniquely identifying the equipment in the database. Wildcards are not allowed. The response body is a plain-text message reporting either success or the reason (e.g. "Not found") for failure.

Arguments:
- **apprtype:** Apparatus type of equipment
- **equipnum:** Equipment number of equipment
- **serialnum:** Serial number of equipment

/toa/rpc/list_tanks (GET) - Export equipment tank information in CSV format. The response body is a list of CSV records. The returned items are filtered according to the arguments presented in the query part of the URI. Filtering is done as for the export_eqp RPC.

Data returned:
A row of column names, followed by rows of values, in CSV format. The column names are: equipnum, serialnum, apprtype, tank.

3.4 TOA Test Data for Manually Collected Samples

/toa/rpc/append_test_data (POST) - Import test data from CSV records given in the body of the request. The query portion of the command URI may be used to specify a value for dateformat to describe the date format used in the CSV data. For each CSV record, the equipment identification (apprtype+equipnum or apprtype+serialnum) is used to look up an equipment object in the database. If there is no match, the CSV record is rejected. If a matching equipment object is found, the tank identifier (if any) specified in the CSV record is used to look up an existing tank belonging to the selected equipment object, or to create a new tank if no match is found. A default tank identifier (presently 'MAIN') is used if none is specified in the CSV record. The combination of sampledate+container_id is used to look for an oil sample record belonging to the designated tank. If no match is found, the CSV record is used to create a new one; if there is a match, however, and the matching oil sample is not marked as REVIEWED, the CSV record is used to fill in data values only where the existing oil sample record has null values. No non-null values are overwritten. If the matching oil sample is already REVIEWED, the CSV data record is rejected. The response body is a plain-text message reporting the number of tanks created and the number of oil sample objects created or updated.

Arguments:
- **dateformat:** One of 'ymd', 'mdy', or 'dmy' to indicate the order of year, month, and day in the date fields provided in

the CSV data. The default, if dateformat is not specified, is ‘ymd’.
Format of CSV data returned:
See “Equipment Information Requirements” (https://www.toa4online.com/toa/help/equipment_info)
Also see ”TOA4 Data Import File Requirements” (http://www.deltaxresearch.com/docs/toa4_data_file_req.pdf)

/toa/rpc/update_test_data (POST) -Update existing test data from CSV records given in the body of the request. The query portion of the command URI may be used to specify a value for dateformat to describe the date format used in the CSV data. For each CSV record, the equipment identification (apprtype+equipnum or apprtype+serialnum) is used to look up an equipment object in the database. If there is no match, the CSV record is rejected. If a matching equipment object is found, the tank identifier (if any) specified in the CSV record is used to look up an existing tank belonging to the selected equipment object. A default tank identifier (presently ‘MAIN’) is used if none is specified in the CSV record. If no matching tank is found, the CSV record is rejected. If a matching tank is found, the combination of sampledate+container_id is used to look for an oil sample record in the database belonging to the designated tank. If no match is found, the CSV record is used to create a new one; if there is a match, however, and the matching oil sample is not marked as REVIEWED, non-null data values in the CSV record are used to fill in or overwrite data values in the existing oil sample record. If the matching oil sample is already REVIEWED, the CSV data record is rejected. The response body is a plain-text message reporting the number of data records updated.

Arguments:
dateformat: One of ‘ymd’, ‘mdy’, or ‘dmy’ to indicate the order of year, month, and day in the date fields provided in the CSV data. The default, if dateformat is not specified, is ‘ymd’.
Format of CSV data returned:
See “Equipment Information Requirements” (https://www.toa4online.com/toa/help/equipment_info)
Also see ”TOA4 Data Import File Requirements” (http://www.deltaxresearch.com/docs/toa4_data_file_req.pdf)

/toa/rpc/write_test_data (POST) This command combines the effects of append_test_data and update_test_data. For each CSV record in the body of the request, if no matching test data record is found in the database, import the record as new test data. If there is a matching test data record in the database, overwrite it with nonblank field values from the CSVrecord. If the tank identifier does not match a tank already defined for the designated equipment item, an attempt is made to create a tank, with the given identifier, for that equipment item. If the attempt is unsuccessful, the CSV record is rejected with an error. If the equipment identification (equipnum+apprtype or serialnum+apprtype) does not match an equipment object already in the database, the CSV record is rejected with an “equipment not found” error. Non-null values from the CSV record are used to update the matching equipment object, over-writing any values which may already be present. The response body is a plain-text message reporting the number of equipment objects which were created or updated.

Arguments: None
Format of CSV data returned:
See “Equipment Information Requirements” (https://www.toa4online.com/toa/help/equipment_info)

/toa/rpc/export_test_data (GET) - Export test data in CSV format. The response body is a block of CSV records, filtered according to the arguments provided. The returned items are filtered according to arguments presented in the query part of the command URI. The arguments are all optional.

Arguments to filter equipment:
apprtype: Apparatus type of equipment
equipnum: Equipment number of equipment
serialnum: Serial number of equipment
owner_name: Owner identifier

region_name: Region identifier
substn_name: Substation identifier

Arguments to filter tanks:
posted_since: UTC timestamp (yyyymmddThhmmssZ)
changed_since: UTC timestamp (yyyymmddThhmmssZ)
otstatus: REVIEWED or UNREVIEWED

Arguments to filter test data:
sampledate: Date or datetime (e.g. yyyy-mm-dd), only samples after date are returned
labtestdate: Date or datetime (e.g. yyyy-mm-dd), only samples after date are returned
labrecvdate: Date or datetime (e.g. yyyy-mm-dd), only samples after date are returned
labreportnum: Lab report number (case-insensitive string match, no wildcards)
ordernum: Order number (case-insensitive string match, no wildcards)
jobnum: Job/Batch number (case-insensitive string match, no wildcards)
sample_otstatus: REVIEWED or UNREVIEWED

Additional non-filtering arguments:
numeric: 0 or 1 (default is 0, if 1 then trace values will be converted to floating-point zeros, e.g. "< 5" is exported as 0.0)
Format of CSV data returned:
See "Equipment Information Requirements" (https://www.toa4online.com/toa/help/equipment_info)
Also see "TOA4 Data Import File Requirements" (http://www.deltaxresearch.com/docs/toa4_data_file_req.pdf)

/toa/rpc/delete_test_data (POST) - Delete all test data belonging to one equipment item from the TOA database. The query portion of the command URI must specify a value for the apprtype and also for either the equipnum or the serialnum, uniquely identifying the equipment in the database. Wildcards are not allowed. The response body is a plain-text message reporting either success or the reason (e.g. "Not found") for failure.

Arguments:
apprtype: Apparatus type of equipment
equipnum: Equipment number of equipment
serialnum: Serial number of equipment

3.5 TOA Analysis Norms

/toa/rpc/list_norms (GET) - Export the analysis norms in CSV format. The response body is CSV formatted document. All defined norms are returned and no filtering via query parameters is supported.

Format of CSV data returned:
See "Analysis Norms File"' (https://www.toa4online.com/toa/help/norms_file)

3.6 TOA Analysis Results

/toa/rpc/analyze_new (POST) - Analyze all previously-unanalyzed test data with status UNREVIEWED currently in the TOA database. The response body is a plain-text message stating the number of test data records processed and the number of equipment items whose data could not be processed because of a missing or invalid norm_name.

Arguments: None

/toa/rpc/analyze_unreviewed (POST) - Analyze all test data with status UNREVIEWED currently in the TOA database. The response body is a plain-text message stating the number of test data records analyzed.

Arguments: None

/toa/rpc/results (GET) - Export analysis results summary. If the query portion of the command URI specifies a sampledate, return latest test results as of (on or before) that date. If apprtype, equipnum, or serialnum is given, filter test results accordingly. The response body is a block of CSV records containing equipment and tank identifiers and analysis results.

Arguments (all optional, used for filtering):

apprtype: Apparatus type of equipment
equipnum: Equipment number of equipment
serialnum: Serial number of equipment
owner_name: Owner identifier
region_name: Region identifier
substn_name: Substation identifier
sampledate: Date or datetime (e.g. yyyy-mm-dd), only samples after date are returned
posted_since: UTC timestamp (yyyymmddThhmmssZ)
changed_since: UTC timestamp (yyyymmddThhmmssZ)

Data returned:
A row of column names, followed by rows of values, in CSV format. The column names are: equipnum, serialnum, apprtype, designation, external_id, tank_name, sampledate, labtestdate, otstatus, norm_used, dga_result, dga_summary, dga_diagnosis, dga_retestdays, dga_retestdate, fq_diagnosis, fq_result, fq_retestdate, fq_retestdays, fq_summary, pcb_result, moisture_result, moisture_summary, moisture_diagnosis, inhibitor_result, assessment, reviewer, reviewdate, rdga_status, gas_event_hf, gas_event_severity, gas_event_fault_type, gas_event_type, gas_event_co_co2_rel_inc, gas_event_c2h2_inc, gas_event_c2h4_inc, gas_event_c2h6_inc, gas_event_ch4_inc, gas_event_h2_inc
Note that columns for which no values are present in the database may be omitted from the CSV data.
See ”Analysis Variables” (https://www.toa4online.com/toa/help/variables)

/toa/rpc/report/all_tests (GET) - Return the latest analysis report for the specified tank of the specified equipment, as a PDF document. The query portion of the command URI must specify the apprtype; and either the equipnum or the serialnum, uniquely identifying the equipment in the database; and the tank. Wildcards are not allowed. The response body is either a PDF document or a plain-text message reporting the reason (e.g. “Equipment not found”) for failure.

Arguments:

apprtype: Apparatus type of equipment
equipnum: Equipment number of equipment
serialnum: Serial number of equipment
tank: Tank identifier (no wildcards)
rdga: Return the Reliability-based DGA report (rdga=1).

/toa/rpc/report/fault_type (GET) - Return the latest fault type report for the specified tank of the specified equipment, as a PDF document. The query portion of the command URI must specify the apprtype; and either the equipnum or the serialnum, uniquely identifying the equipment in the database; and the tank. Wildcards are not allowed. The response body is either a PDF document or a plain-text message reporting the reason (e.g. “Equipment not found”) for failure.

Arguments:

apprtype: Apparatus type of equipment
equipnum: Equipment number of equipment
serialnum: Serial number of equipment
tank: Tank identifier (no wildcards)

/toa/rpc/export_gas_events (GET) - Export gas event data for gassing events found by the Reliability-based DGA analysis. If apprtype, equipnum, or serialnum parameters are given, filter results accordingly. The response body is a block of CSV records containing equipment and tank identifiers and gas event data.

Arguments (all optional, used for filtering):

apprtype: Apparatus type of equipment
equipnum: Equipment number of equipment
serialnum: Serial number of equipment
owner_name: Owner identifier
region_name: Region identifier
substn_name: Substation identifier
otstatus: REVIEWED or UNREVIEWED
posted_since: UTC timestamp (yyyymmddThhmmssZ)
changed_since: UTC timestamp (yyyymmddThhmmssZ)

Data returned:
A row of column names, followed by rows of values, in CSV format. The column names are: equipnum, serialnum, apprtype, tank, dga_first_date, dga_last_date, gas_event_type, gas_event_start_date, gas_event_end_date, gas_event_start_value, gas_event_end_value, gas_event_num_samples, gas_event_summary, gas_event_severity, gas_event_hf, gas_event_fault_type, gas_event_co_co2_rel_inc
Note that columns for which no values are present in the database may be omitted from the CSV data.

See ”Gas Event Variables” (https://www.toa4online.com/toa/help/gas_event_vars)

3.7 TOA Monitor Watch^{TM} Commands

These RPC commands are available only in connection with the optional Monitor Watch extension to TOA, for exchanging data and information pertaining to online monitors registered in Monitor Watch. **/toa/rpc/append_monitor_data** (POST) - Import online monitor data from CSV records given in the body of the request.

Arguments: None

/toa/rpc/monitor_results (GET) - Export monitor data analysis results summary. If the query portion of the command URI specifies a sampledate, return latest test results as of (on or before) that date. If apprtype, equipnum, or serialnum is given, filter test results accordingly. The response body is a block of CSV records containing equipment and tank identifiers and analysis results.

Arguments: None

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

Arguments:

monitor_id: Monitor ID

Example:

https://www.toa4online.com/toa/rpc/report/monitor_status?monitor_id=MONITOR_ID

3.8 Supplementary Remarks

The following items contain information and advice of generally relevance for RPC data transactions with TOA.

1. Simple wildcard expressions (using '*' only) can be used with all text-type filter arguments for 'export_eqp', 'export_test_data', and 'results' RPC commands.

2. TOA RPC accepts requests using POST even if GET is expected.

3. When errors are encountered importing equipment or test data, the content of the HTTP response contains CSV data representing the rejected records, with an extra column containing the error message for each of those records.

4. To update a field to be blank, even if it is currently nonblank, use the value '$NULL$' for that field in the data sent via an update or write command. Simply leaving the field blank in the input data will not cause the value in the database to become blank.

5. Incoming values of 'apprtype' must match predefined values in the target TOA database. Furthermore, it is possible for the subscriber to define approved values of 'tank' for each apprtype, so that all non-matching values in incoming test data will be rejected.

6. Incoming values of 'apprtype' must match predefined values in the target TOA database. Furthermore, it is possible for the subscriber to define approved values of 'tank' for each apprtype, so that all non-matching values in incoming test data will be rejected.

7. The returned content of web requests contain a list of encountered problems in CSV format. It is recommended that this returned content is passed to a log file and reviewed periodically.

A Appendix: HTTP Examples and Resources

A.1 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 TOA 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.

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, or manually point to the TOA Online certificate (contact support). **Do not revert to using the insecure http protocol for RPC transactions!** There is no problem with the TOA security certificates. The certification authority for all public TOA web sites is Go Daddy Secure Certification Authority, a well-known and widely used CA.

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 “null” command. No user authentication is required

```
> curl "https://www.toa4online.com/toa/null"
```

cURL Example 2. Run the “welcome” command.

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

cURL Example 3. Data upload. Import new laboratory data into TOA 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_test_data"
```

cURL Example 4. Download test data to a file.

```
> curl -u rpcloginid:rpcpassword -o filepath/filename.csv
"https://www.toa4online.com/toa/rpc/export_test_data?equipnum=XYZ123&apprtype=TRN&tank=MAIN"
```

cURL Example 5. Retrieve a PDF copy of the latest lab test data analysis report for a specific equipment item.

```
> curl -u rpcloginid:rpcpassword -o filepath/myreport.pdf
"https://www.toa4online.com/toa/rpc/report/all_tests?equipnum=XYZ123&apprtype=TRN&tank=MAIN"
```

A.2 PowerShell Examples

PowerShell Example 1. Data upload. Write equipment and laboratory data into TOA Online from a CSV data file.

```
$urlbase = 'https://www.toa4online.com/toa/rpc'
$user = 'test1-rpc'
$pass = 'password'
$workdir = "C:\Users\Username\Scripts"
$eqpfile = "$workdir\eqp.csv"
$testdatafile = "$workdir\testdata.csv"

function GetUrl {
   Param ($url, $filePath)
   $pair = "$($user):$($pass)"
   $encodedCreds = [System.Convert]::ToBase64String([System.Text.Encoding]::ASCII.GetBytes($pair))
   $basicAuthValue = "Basic $encodedCreds"
   $Headers = @{
      Authorization = $basicAuthValue
   }
   $resp = Invoke-WebRequest -Headers $Headers -Uri $url -Method Post -InFile $filePath -ContentType
       "text/csv; charset=utf-16"
   return $resp
}

# send equipment data
$r = GetUrl "$urlbase/write_eqp" "$workdir\eqp.csv"
write-host $r.Content

# send test data
$r = GetUrl "$urlbase/write_test_data" "$workdir\testdata.csv"
write-host $r.Content
```

PowerShell Example 2. Data download. Export equipment and laboratory data from TOA Online into a CSV data file.

```
$urlbase = 'https://www.toa4online.com/toa/rpc'
$user = 'test1-rpc'
$pass = 'password'
$workdir = "C:\Users\Username\Scripts"
$timestampfile = "$workdir\timestamp.txt"
$eqpfile = "$workdir\eqp.csv"
$testdatafile = "$workdir\testdata.csv"

function GetUrl {
   Param ($url, $filePath)
   $pair = "$($user):$($pass)"
   $encodedCreds = [System.Convert]::ToBase64String([System.Text.Encoding]::ASCII.GetBytes($pair))
   $basicAuthValue = "Basic $encodedCreds"
   $Headers = @{
      Authorization = $basicAuthValue
   }
   $resp = Invoke-WebRequest -Headers $Headers -Uri $url -Method Post -InFile $filePath -ContentType
       "text/csv; charset=utf-16"
   return $resp
}

#get timestamp from server
$r = GetUrl "$urlbase/timestamp"
$next_timestamp = $r.Content
```

```
# read timestamp from previous run
$timestamp = Get-Content $timestampfile -Raw

# get changed equipment data
$r = GetUrl "$urlbase/export_eqp?changed_since=$timestamp"
$r.Content | out-file -filepath $eqpfile

# get changed test data
$r = GetUrl "$urlbase/export_test_data?changed_since=$timestamp"
$r.Content | out-file -filepath $testdatafile

# save timestamp for next run
"$next_timestamp" | out-file -filepath $timestampfile
```

A.3 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.

A.4 HTTP Basic Authentication

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

Every RPC request sent to TOA should contain an HTTP Authorization header to avoid receiving a 401 (Unauthorized) response from TOA. The appearance and structure of such a header is explained by the following example. For the example, assume that the RPC login ID is “somecorpc1” 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 TOA.

When sending RPC HTTP requests to TOA 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 TOA 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 TOA **security certificates** are assigned to the **server hostname**, not to the server IP address, so **attempts to communicate via https with TOA/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.

A.5 Example of simple RPC HTTP Request

Below is the full text of an HTTP request for a results command.

```
GET /toa/rpc/results?sampledate=2009-10-01&substn_name=BOONYVILLE HTTP/1.1
Authorization: Basic c29tZWNvLXJwYzE6YzBuZnVzMW5n
Host: www.toa4online.com
Accept: */*
```

A.6 Example of RPC Request with data

Below is the full text of an HTTP request for an append_test_data command. Note that the row of column names is wrapped for display in this document only – it is not wrapped (i.e., does not contain a carriage-return or linefeed character) in the text of the HTTP request.

```
POST /toa/rpc/append_test_data?dateformat=ymd HTTP/1.1
Authorization: Basic c29tZWNvLXJwYzE6YzBuZnVzMW5n
Host: www.toa4online.com
Content-Type: text/plain; charset=utf-8
Content-Length: 578
equipnum,apprtype,tank,sampledate,fluidtempc,h2,ch4,c2h6,c2h4,c2h2,
co,co2,o2,n2,acidnum,ift,d1816_2,water
5544B,TRN,MAIN,2000-09-26,50,294,121,137,38,0,223,3004,2340,22698,0.03,30,40,3
5544B,TRN,MAIN,2004-08-01,50,379,194,175,51,0,341,4213,2627,25482,0.15,26,38,18
5544B,TRN,MAIN,2005-03-06,50,689,428,320,109,0,315,1652,685,24333,0.19,22,36,22
5544B,TRN,MAIN,2006-03-28,50,1298,2009,1021,369,0,530,6524,732,24800,0.25,21,34,24
5544B,TRN,MAIN,2008-03-21,50,1360,2554,1332,561,0,554,5952,1027,24651,0.28,21,34,29
```

A.7 Example of simple RPC HTTP Response

Below is the full text of an HTTP response sent back from TOA in reply to an append_test_data command where a single data record was submitted and successfully imported, with no new tanks being created.

```
HTTP/1.1 200 OK
Server: nginx/1.6.2
Content-Type: text/plain; charset=utf-8
Content-Length: 22
Connection: keep-alive
Date: Tue, 01 Mar 2016 19:21:39 GMT
Expires: -1
Cache-Control: max-age=0, must-revalidate
tanks: 0 records: 1
```

A.8 Example of RPC Response with data errors

Below is the full text of an HTTP response sent back from TOA in reply to an append_test_data command when an equipnum specified in the data did not match any equipment in the database. The first line of the response body, ”tanks: 0

records: 0", indicates that in this case no tanks were created and no data records were imported. The rest of the response body contains CSV data indicating which records were rejected and why.

```
HTTP/1.1 200 OK
Server: nginx/1.6.2
Content-Type: text/plain; charset=utf-8
Content-Length: 149
Connection: keep-alive
Date: Tue, 01 Mar 2016 20:07:16 GMT
Expires: -1
Cache-Control: max-age=0, must-revalidate
tanks: 0 records: 0
import_error,equipnum,apprtype,tank,sampledate,fluidtempc,exclude,water
Equipment not found,E14544,TRN,MAIN,2016-03-01,37,1,10
```

A.9 Example of RPC HTTP Response with data

Below is the full text of an HTTP response sent back from TOA in reply to a results command. Note that the rows of column names and data are wrapped for display in this document only – they are not wrapped (i.e., do not contain a carriage-return or linefeed character) in the text of the HTTP response.

```
HTTP/1.1 200 OK
Server: nginx/1.6.2
Content-Type: text/csv; charset=utf-8
Content-Length: 553
Connection: keep-alive
Date: Tue, 01 Mar 2016 20:15:29 GMT
Expires: -1
Cache-Control: max-age=0, must-revalidate
equipnum,serialnum,apprtype,designation,external_id,external_id2,tank,labreportnum,sampledate,labtestdate,
jobnum,ordernum,otstatus,norm_used,dga_result,dga_retestdays,dga_retestdate,dga_remarks,moisture_result,
moisture_remarks,context,dga_refdays,tank_name
E1464,SN1119,TRN,,,,MAIN,,2005-06-09,,2005-06,,REVIEWED,TRN_IEEE_INC_69KV,1,365,2006-06-09,No
    anomalies.,1,The
water content of the oil seems acceptable.,SCREENING,310,MAIN
E1463,SN1119,LTC,,,,MAIN,,2005-06-09,,2005-06,,REVIEWED,LTC_GENERIC,1,365,2006-06-09,No
anomalies.,,,SCREENING,310,MAIN
```

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).

Document Revision Notes

Date	Version	Changes	Author
2017-05-17	1.2	Updated Initial Draft	Neil Schemenauer
2017-07-18	2.0	Updated format, expanded on CSV header information	Steven Herchak
2017-11-30	2.1	Fixed minor typos	Steven Herchak
2018-05-08	2.2	Fixed PowerShell example 2	Steven Herchak
2018-07-13	2.3	Fixed analsysis norm RPC function	Steven Herchak
2018-11-01	2.4	Add export_gas_events	Neil Schemenauer
2020-01-08	2.5	Remove append_eqp and update_eqp	Neil Schemenauer
2022-12-05	2.6	Include specific references to get R-DGA results	Zack Draper
2023-04-20	2.7	Add report/fault_type	Matthew Milholm