TOA4 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.

The TOA4 RPC API is a simple HTTP-based [HT1, HT2] command interface conforming to the design principles of Representational State Transfer (REST) [RES]. For security, TOA4 RPC should be run using the HTTP Secure protocol (https) [TL1, TL2, TL3] whenever the server has a security certificate.

Access to a TOA4 database via TOA4 RPC is an **extra-cost option** which must be purchased from Delta-X Research by the TOA4 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.

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_test_data
```

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
```

If the request is directed to TOA4 running somewhere other than on the main TOA4 Online server, then of course the server address portion of the URI is different from "www.toa4online.com".

The HTTP **request method** (GET or POST) required in each case is specified with the description of each RPC command below. Generally, commands which only retrieve information use the GET method, and commands which 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** like the one used by 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.

RPC Commands

TOA4 RPC commands are grouped as follows:

- Software testing
- Equipment information
- TOA4 test data for manually-collected samples
- TOA4 analysis results
- TOA4 Monitor Watch™ online monitor data
- TOA4 Monitor Watch™ analysis results
- Reports
- Other information

Software Testing

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

Example: `Welcome 'someco-rpc1'.  You are using the 'SOMECO' database.`

Arguments: None

Exceptions: None

See also: The *timestamp* RPC command (below).

Equipment Information

append_eqp (POST)

Arguments: None

Action: Import equipment information from .csv records given in the body of the request. For each .csv data record, create a new equipment object in the database only if the equipment identification (equipnum+apprtype or serialnum+apprtype) is not found in the database.

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 appended and the number of existing equipment objects updated.

Special cases: In cases where there is already an equipment object in the database with matching identification, update that object's null fields (only) with non-null values from the .csv data record. Note that the .csv record does not update any equipment fields which already contain a non-null value.

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.

update_eqp (POST)

Arguments: None

Action: Update existing equipment from .csv records given in the body of the request. For each .csv record, if the equipment identification (equipnum+apprtype or serialnum+apprtype) matches an equipment object already in the database, non-null values from the .csv record are used to update the matching equipment object, over-writing any values which may alreay be present.

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 existing equipment objects which were updated.

Errors: In cases where there is no equipment object in the database with matching equipment identification, an "equipment not found" error occurs.

write_eqp (POST) - This command combines the effects of import_eqp and update_eqp. Import equipment information from .csv records given in the body of the request. For each .csv record, if the equipment identification (equipnum+apprtype or serialnum+apprtype) matches an equipment object already in the database, non-null values from the .csv record are used to update the matching equipment object, over-writing any values which may alreay be present. The response body is a plain-text message reporting the number of existing equipment objects which were updated.

Arguments: None
Format of .csv data:
See "Equipment Information Requirements" (https://www.toa4online.com/toa/help/equipment_info)

export_eqp_key (GET) - Return the internal TOA4 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 TOA4 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 TOA4 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'.

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

delete_eqp (POST) - Delete one equipment item from the TOA4 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

TOA4 Test Data for Manually Collected Samples

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:
See "Data File Requirements" (https://www.toa4online.com/toa/help/datafiles_req)
Also see "TOA4 Data Import File Requirements"
(http://www.deltaxresearch.com/docs/toa4_data_file_req.pdf)

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:

See "Data File Requirements" (https://www.toa4online.com/toa/help/datafiles_req)
Also see "TOA4 Data Import File Requirements"
(http://www.deltaxresearch.com/docs/toa4_data_file_req.pdf)

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 .csv record. 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 a " 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 alreay be present. The response body is a plain-text message reporting the number of existing equipment objects which were updated.

Arguments: None
Format of .csv data:

See "Equipment Information Requirements"
(https://www.toa4online.com/toa/help/equipment_info)

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.

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

Format of .csv data returned:

See "Data File Requirements" (https://www.toa4online.com/toa/help/datafiles_req)
Also see "TOA4 Data Import File Requirements"
(http://www.deltaxresearch.com/docs/toa4_data_file_req.pdf)

delete_test_data (POST) - Delete all test data belonging to one equipment item from the TOA4 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

TOA4 Analysis Results

analyze_new (POST) – Analyze all previously-unanalyzed test data with status UNREVIEWED currently in the TOA4 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

analyze_unreviewed (POST) - Analyze all test data with status UNREVIEWED currently in the TOA4 database. The response body is a plain-text message stating the number of test data records analyzed.

Arguments: None

analyze (POST) - Run analysis against .csv records given in the body of the request. The query portion of the command URI must specify a valid norm_name for analysis of all of the data presented. The query may also specify a value for dateformat to describe the date format used in the .csv data. The response body contains the same .csv records with results inserted. This command does not create or modify records in the TOA4 database and does not require the referenced equipment to be in the database, but it does look up the analysis norms in the TOA4 database. The intended use of the 'analyze' command is for processing monitoring data.

Arguments:
norm_name: (Required) Name of analysis norms to look up in TOA4 database and apply when analyzing all of the data presented.
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:
See "Data File Requirements" (https://www.toa4online.com/toa/help/datafiles_req)
See also "TOA4 Data Import File Requirements" (http://www.deltaxresearch.com/docs/toa4_data_file_req.pdf)

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: "On or before" date for results, in yyyy-mm-dd format

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.`

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)

TOA4 Monitor Watch™ Commands

These RPC commands are available only in connection with the optional Monitor Watch extension to TOA4, for exchanging data and information pertaining to online monitors registered in Monitor Watch.

Note: Monitor Watch is still under development as of the date of this document, so the information below may change by the time Monitor Watch is officially released.

append_monitor_data (POST) - Import online monitor data from .csv records given in the body of the request.

Arguments: None

export_monitor_data (GET) - Export online monitor data. The response body is a block of .csv records, filtered according to the arguments presented in the query part of the command URI.

Arguments: None

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

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

Reports

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

Other Information

timestamp (GET) – Return a plain-text UTC timestamp from the TOA4 server in yyyymmddThhmmssZ format. Example: `20070502T233738Z`.

Arguments: None

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. If you include a -v argument, cURL echoes the HTTP request sent and the HTTP response received. The command URL, in double quotes, should be at the end.

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_test_data"
```

cURL Example 3. 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 4. Delete a specified equipment item, along with all of its test data.

```
> curl -X POST -u rpcloginid:rpcpassword
"https://www.toa4online.com/toa/rpc/delete_eqp?equipnum=XYZ123&apprtype=TRN"
```

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.

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: */*
```

Example of RPC HTTP 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
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
```

Example of simple RPC HTTP Response

Below is the full text of an HTTP response sent back from TOA4 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
Date: Tue, 10 Nov 2009 21:50:42 GMT
Server: Apache/2.2.9 (Debian) mod_scgi/1.12 mod_ssl/2.2.9 OpenSSL/0.9.8g
Expires: -1
Content-Length: 22
Vary: Accept-Encoding
Content-Type: text/plain; charset=utf-8

tanks: 0 records: 1
```

Example of RPC HTTP Response with data errors

Below is the full text of an HTTP response sent back from TOA4 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
Date: Tue, 10 Nov 2009 21:41:16 GMT
Server: Apache/2.2.9 (Debian) mod_scgi/1.12 mod_ssl/2.2.9 OpenSSL/0.9.8g
Expires: -1
Content-Length: 202
Vary: Accept-Encoding
Content-Type: text/plain; charset=utf-8

tanks: 0 records: 0
import_error,equipnum,apprtype,sampledate,fluidtempc,h2,ch4,c2h6,c2h4,c2h2,co,co2,o2,n2,tank_name
Equipment not found,E14522,TRN,2009-11-01,35,23,14,3,0,0,341,2105,5200,16500,MAIN
```

Example of RPC HTTP Response with data

Below is the full text of an HTTP response sent back from TOA4 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
Date: Thu, 08 May 2008 18:00:01 GMT
Content-Length: 607
Content-Type: text/comma-separated-values; charset=ISO-8859-1

equipnum,serialnum,apprtype,tank_name,sampledate,labtestdate,otstatus,norm_used,dga_result,dga_diagnosis,dga
_retestdays,dga_retestdate,fq_result,fq_diagnosis,moisture_result,moisture_diagnosis,dga_summary,fq_summary,
moisture_summary
E1454,SN1090,TRN,MAIN,2005-10-18,,UNREVIEWED,TRN_IEEE_INC_69KV,3,T3,30,2005-11-17,,,,,h2* ch4* c2h6* c2h4*
c2h2*^+ co*,,
E1452,SN0515,TRN,MAIN,2005-10-13,,UNREVIEWED,TRN_IEEE_INC_69KV,3,T2,30,2005-11-12,2,CONTAMINATED,1,,c2h6*
c2h4* c2h2*^,ift*,
E1453,SN0296,TRN,MAIN,2005-04-27,,UNREVIEWED,TRN_IEEE_INC_69KV,3,DT,30,2005-05-27,,,2,WET-OIL,c2h2*^,,water*
relsaturation*
```

References

[RES] Wikipedia, "Representational State Transfer,"
http://en.wikipedia.org/wiki/Representational_State_Transfer (2009-11-09).

[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).