TFRRS API for Desktop Meet Management Software and CSV Results Format

December 30th, 2013

Version 1.8

`

Table of Contents

I. Introduction

The Track & Field Results Reporting System ("TFRRS") is a track & field performance and cross country results aggregation system presented by DIRECTATHLETICS and the U.S. TRACK & FIELD AND CROSS COUNTRY COACHES ASSOCIATION. TFRRS only accepts performances from meet management software; teams may not submit their own performances to TFRRS. This ensures a higher level of efficiency and reliability in performance reporting.

This document describes how results can be submitted to TFRRS from desktop meet management software. Much of the document focuses on the Web Services API (SOAP/REST). Additionally, results may be uploaded in CSV ("comma separated values") formats.

The Web Services API allows meet management software to connect to TFRRS, login, download rosters and submit results. The goal of the API is to enable vendors of meet management software to integrate TFRRS support directly into their software, enabling users to import rosters from TFRRS and submit results to TFRRS without leaving their meet management software.

Web Services

"Web Services" are software systems designed to support machine-to-machine interaction over a network. A Web Service is described in a published Web Service Description Language document ("WSDL"). This document is machine-readable and most software languages have libraries that can read a WSDL and create hooks into that service's methods.

The WSDL for this API is:

http://www.tfrrs.org/tfrrs_mm.wsdl

The above-referenced document is written in XML and conforms to the WSDL specification. It is intended to be consumed by your Web Services library. The WSDL describes 7 methods: get_client_key, validate_client_key, meets, unpublished_meets, get_entries, login, get_rosters and submit_results. Only the last three methods implement the core functionality of TFRRS. The first two methods are necessary to identify your software as valid TFRRS-compatible software. get_rosters retrieves roster data for a meet and get_entries retrieves entry data from TFRRS or DirectAthletics. "unpublished_meets" and "meets" retrieve the meets on which the user may perform actions.

SOAP

This Web Service is a SOAP web service. SOAP is a protocol that dictates the format of requests and responses to and from the Web Service. SOAP is a way of describing

the parameters a method requires and the values it returns. SOAP requests and responses are XML documents that are transmitted via HTTP. If you have a good Web Services/SOAP library you won't need to know the exact format of these XML documents or even ever see them.

REST

Alternatively, the Web Services API is available as a REST service. REST has less strict XML formatting requirements than SOAP and does not require a Web Services/SOAP library to use. However, this means that REST requests must be created and responses must be parsed without the assistance of a pre-existing library. Using SOAP is recommended over REST but if you are unhappy with the Web Services/SOAP libraries available on your development platform you may need to use the REST API. The REST API works almost exactly like the SOAP API. The REST section of this document only covers REST-specific issues. Use the SOAP portion of the document as a reference even if you are using the REST API.

The TFRRS Website

Any meet director can download team rosters, regardless of what online service (if any) is used to collect entries. To do so, a meet director should create a director account on the TFRRS website (www.tfrrs.org). The meet director should then create a TFRRS record for this meet – indicating the meet's name, date and location. The meet director may also select which team rosters, if any, to download. Rosters are available in CSV format and may also be downloaded via this API (see get_rosters).

II. The Track & Field CSV File Format

As an alternative to Web Services, TFRRS users may submit results via CSV file. Despite the name, fields in a CSV file should be tab delimited. It is expected that most CSV files will be generated by Microsoft Excel using "Save As", "Tab Delimited". The columns of a CSV file must match the format described here precisely.

CSV files can be tested for compliance with this format at:

http://www.tfrrs.org/upload_test.html

Examples of valid CSV files are available on this page.

The columns, in order, of a TFRRS CSV file are:

1) bib (bib or TFRRS/DirectAthletics ID is required)
2) TFRRS or DirectAthletics ID
3) team name (leave blank for unattached)
4) team code (leave blank for unattached)
5) first name
6) last name

7) gender (“m” or “f”)
8) year (either the year of graduation, FR,SO,JR,SR or grade number).
9) date of birth (YYYY-MM-DD, though other common format will work)
10) event code
11) event name
12) division name (The division for this team/athlete or event – see Divisions)
13) event min age
14) event max age
15) sub event code (leave this column blank if this is not a sub event result)
16) mark (in seconds, meters, inches, or points)
17) metric (1 if the mark is in meters, otherwise 0)
18) fat (1 if automatic timing, otherwise 0)
19) place
20) score
21) heat
22) heat place
23) round (P, Q,, S, F)
24) points (if this is a sub event of a multi event, otherwise blank)
25) wind (like “-3.3” or “1.2”) (leave blank if no wind reading)
26) relay squad
27) relay athlete 1 first_name
28) relay athlete 1 last_name
29) relay athlete 1 bib
30) relay athlete 1 TFRRS/DirectAthletics ID
31) relay athlete 2 first_name
32) relay athlete 2 last_name
33) relay athlete 2 bib
34) relay athlete 2 TFRRS/DirectAthletics ID
35) relay athlete 3 first_name
36) relay athlete 3 last_name
37) relay athlete 3 bib
38) relay athlete 3 TFRRS/DirectAthletics ID
39) relay athlete 4 first_name
40) relay athlete 4 last_name
41) relay athlete 4 bib
42) relay athlete 4 TFRRS/DirectAthletics ID
43) relay athlete 5 first_name
44) relay athlete 5 last_name
45) relay athlete 5 bib
46) relay athlete 5 TFRRS/DirectAthletics ID
47) relay athlete 6 first_name
48) relay athlete 6 last_name
49) relay athlete 6 bib
50) relay athlete 6 TFRRS/DirectAthletics ID
51) Field Attempt 1 Mark or bar height (meters or inches)
52) Field Attempt 1 Metric (1 if mark/height is in meters, otherwise 0)

53)Field Attempt 1 Status or result ("FOUL" or "XXO")
54)Field Attempt 1 Wind (like "-3.3" or "1.2")
55)Field Attempt 2 Mark or bar height (meters or inches)
56)Field Attempt 2 Metric (1 if mark/height is in meters, otherwise 0)
57)Field Attempt 2 Status or result ("F" or "XXO")
58)Field Attempt 2 Wind (like "-3.3" or "1.2")
59)Field Attempt 3 Mark or bar height (meters or inches)….

[Everything after the 6th relay athlete is a field attempt]

Divisions: Event and Team

The division name field (column #12) is used to provide the division name for either the event (if the meet assigns divisions per event) or for the team/athlete (if the meet is using team division scoring). When uploading a track & field CSV file, the user will be prompted to indicate whether team division scoring was used.

III. The Cross Country CSV File Format

IV. Connecting to the SOAP Service

As stated above, the WSDL for this service is at:

http://www.tfrrs.org/tfrrs_mm.wsdl

This file just describes the API. The location where your software will be directing SOAP messages is:

http://www.tfrrs.org/tracksoap

If your SOAP library is robust enough, you won't actually have to enter this location; your library will find it in the WSDL. But this is not always the case.

The Credentials element

All requests to the service must include a "Credentials" element. The Credentials element can contain several elements within it:

- vendor
- version
- license
- client_key
- session_id

vendor: The credentials element must always contain a valid vendor element. This is a string indicating the vendor of your software. If you haven't been issued a vendor tag from TFRRS, you should request one by emailing support@directathletics.com.

version: The version number of your software. When you request a vendor tag, indicate the version number of your software. This way TFRRS can remain compatible with older versions of your software as improvements are made to TFRRS and your software.

license: A string identifying this copy of your software. This is not required by TFRRS but it may be useful to be able to differentiate copies of your software for debugging purposes.

client_key: A string issued by TFRRS which identifies this software as TFRRS compatible. A client key, once validated, is valid for 24 hours. client key validation is described below.

session_id: A string identifying the session of a logged-in user. This is issued by TFRRS upon successful login.

A Credentials element in XML looks something like:

<Credentials>
<vendor>SoftwareCo</version>
<version>1.3</vesion>
<client_key>sdfdsfsd…</client_key>
</Credentials>

(the actual XML would be much harder to read, I've limited this to tag names and values).

Getting a Client Key

All requests to TFRRS must include a validated client key in the Credentials element, with the exception of requests to the two methods described below (get_client_key and validate_client_key).

To get a client key, call the method get_client_key. The only argument to this method is the Credentials element, and here the Credentials element need only include a valid vendor. The vendor tag "Demo" will be used throughout this document and is available to for use for testing/developing purposes.

The Request:

The entire SOAP request to get_client_key for the vendor "Demo" looks like:

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
```

```
<m:get_client_key xmlns:m="urn:TFRRS">
<Credentials xsi:type="typens:Credentials">
<vendor xsi:type="xsd:string">Demo</vendor>
<version xsi:type="xsd:string">1.0</version>
</Credentials>
</m:get_client_key>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

This is a complete SOAP request. It includes the "Envelope" and "Body" elements as well as type attributes. Note that all the arguments to get_client_key, here just "Credentials", are inside a tag named "get_client_key". The remainder of this document will only refer to the content between the XML tags named after the method. Everything else should be handled by your SOAP library.

You shouldn't have to write this XML request yourself. A good Web Services/SOAP library will allow you to make this request using syntax similar to:

```
#/usr/bin/perl
# Connect to TFRRS SOAP API in Perl:
use SOAP::Lite;

my $soap = SOAP::Lite->service("http://www.tfrrs.org/tfrrs_mm.wsdl");

my $response = $soap->get_client_key( "Credentials" => {vendor => "Demo",
                                                        version => "1.0"});
```

The Response:

If you provided a valid vendor code, the response from get_client_key will look something like:

```
<client_key>yHbExEJnaVXlSFYZlfEmwg</client_key>
<challenge_phrase>F/AIB9G94pH6eqk5arZLHZA9/nEqPGufkHLzs9qjF9Ysg </challenge_phrase>
```

(again, I've removed the Envelope, Body, get_client_keyResponse tags, as well as all type attributes for readability).

The two elements of the response are client_key and challenge_phrase. You should not have to actually parse this XML, your SOAP library will do that for you. To continue the Perl example above, the two elements in the response should be available to you using something similar to:

```
print "client key is $response->{client_key}, challenge phrase is $response->{challenge_phrase}\n";
```

Validating the Client Key

Now you have a client key. However, this client key must be validated before it will grant your software access to the TFRRS API. To validate the client key you must call validate_client_key with the client key and a validation string as arguments (and Credentials, of course).

To create the validation string you will need the challenge phrase returned by get_client_key and your secret key. When you receive your vendor tag you will also receive a secret key.

The secret key for the example vendor Demo is "**sEeKDWK5Yfb7SFp6Na5emA**". Use this key for development/testing in conjunction with the vendor "Demo".

To create the validation string, concatenate your secret key and the challenge phrase. The MD5 hash of the combined string is your validation code. MD5 is a hash algorithm available in almost every software language and is also available as a function in most databases. In Perl:

```
use Digest::MD5;
my $validation = Digest::MD5::md5_hex($secret_key . $challenge_phrase);
```

Or in SQL:

```
sql> select md5("sEeKDWK5Yfb7SFp6Na5emF/AIB9G94pH6eqk5arZLHZA9/nEqPGufkHLzs9qjF9Ysg ") as validation;
+----------------------------------------------------+
| validation                                         |
+----------------------------------------------------+
| 0f79ca05f5e3783218e33debd0b9a0f1  |
+----------------------------------------------------+
```

No matter what implementation you use, you should get the same result. For testing purposes, the challenge phrase is *always* F/AIB9G94pH6eqk5arZLHZA9/nEqPGufkHLzs9qjF9Ysg and the md5 hash of the demo secret key (sEeKDWK5Yfb7SFp6Na5emA) and the validation phrase will always be 0f79ca05f5e3783218e33debd0b9a0f1. This allows you to develop and test even if you are having trouble generating validation strings.

Once you have your validation string, call validate_client_key with your client key and the validation string as arguments:

```
<Credentials>
<vendor>Demo</vendor>
<version>1.0</version>
</Credentials>
<client_key>yHbExEJnaVXlSFYZlfEmwg</client_key>
<validation>0f79ca05f5e3783218e33debd0b9a0f1 </validation>
```

But hopefully your SOAP library allows you do something like:

```
my $response = $soap->validate_client_key(Credentials => {vendor => "Demo",
                                                          version => "1.0"},
                                          client_key => $client_key,
                                          validation => $validation);
```

The response from TFRRS should look like:

```
<validated>1</validated>
<expires>2009-10-13 16:36</expires>
```

(with the value of “expires” being a point 24 hours in the future).

Your client key is now validated. It will allow access to TFRRS for 24 hours. You can get another client key at any time by repeating the above steps.

V. Submitting Results Via The SOAP Service

Now that you have a validated client key you may call login. All of the remaining methods (submit performances, meets, unpublished_meets, get_entries) required a valid session ID. To get a session ID, call login with a valid username and password. The accounts that may login to TFRRS via this API are TFRRS/DirectAthletics accounts that have authority to perform actions on meets (meet directors and administrators). A user does not need to use DirectAthletics for online entries to be able to submit performances via the TFRRS API. The TFRRS website allows a user to create an account for purposes of uploading performances.

For testing/demonstration purposes, the account we will use is:

Username: tfrrsdemo1
Password: test

This account is available to you for testing/development. Performances submitted via this account will not be added to the TFRRS database.

Logging In:

The login method takes a username and password as arguments and returns a session ID and two lists: meets and unpublished_meets.

Request:

```
<Credentials>
<client_key>yHbExEJnaVXlSFYZlfEmwg</client_key>
<vendor>Demo</vendor>
<version>1.0</version>
</Credentials>
<username>tfrrsdemo1</username>
<password>test</password>
```

If your username and password are invalid, your response will look like:

```
<error>Invalid username or password.</error>
<err>1</err>
```

This is the general pattern for all error responses: An error flag “err” and an error message “error”.

If your username and password were valid, your response will be much longer:

```
<session_id>2720416-470627</session_id>
```

```
<meets soapenc:arrayType="xsd:anyType[5]" xsi:type="soapenc:Array">
<item>
<date_end>2010-05-08</date_end>
<registration_end>2010-05-03 23:59:00</registration_end>
<name>Putnam Classic</name>
<sport>track</sport>
<status_message>Registration is in progress.</status_message>
<count_entries>2</count_entries>
<results_exist>false</results_exist>
<registration_begin>2009-09-10 00:00:00</registration_begin>
<date_begin>2010-05-08</date_begin>
<venue>
<name>Christopher Newport</name>
<state>VA</state>
</venue>
<meet_hnd>13590</meet_hnd>
</item>
<item>
<date_end>2009-12-27</date_end>
<registration_end>2009-12-22 23:59:00</registration_end>
<name>API Championship</name>
<sport>xc</sport>
<status_message>Registration is in progress.</status_message>
<count_entries>6</count_entries>
<results_exist>false</results_exist>
<registration_begin>2009-10-06 00:00:00</registration_begin>
<date_begin>2009-12-27</date_begin>
<venue>
<name>Eastern Michigan</name>
<state>MI</state>
</venue>
<meet_hnd>13593</meet_hnd>
</item>

[ several more meets are described here – I've cut them for
readability]

</meets>
<unpublished_meets xsi:type="soapenc:Array">
<item>
<date_end>2009-10-01</date_end>
<name>Waterbury Invitational</name>
<sport>track</sport>
<unpublished_meet_hnd>1985</unpublished_meet_hnd>
-<date_begin>2009-10-10</date_begin>
</item>
</unpublished_meets>
```

The most important part of the response is at the top: The session_id. Put this in the Credentials element for all requests going forward. The session_id is valid until it goes twelve hours without being used for a request.

The other two elements of the response are meets and unpublished_meets. These are arrays. I have left in some of the type attributes so that it is apparent they are different from the XML elements seen so far. Between the <meets> and </meets> tags are several meet definitions, each within its own <item> </item> tags. The elements of a meet are:

- meet_hnd – the unique identifier of this meet in TFRRS
- name
- sport (either "track" or "xc")
- date_begin
- date_end
- count_entries – how many entries this meet has in DirectAthletics
- results_exist - a boolean ("true"or "false") indicating if results have been submitted.
- registration_begin
- registration_end
- status_message – the status of online entries for this meet on DirectAthletics
- venue – an element containing the name and state of the facility of this meet.

meet_hnd is the most important element of a meet.

The unpublished_meets array is almost identical to the meets array. An unpublished meet is a meet that is invisible to all but the user who created it. When a user creates a meet solely for the purpose of submitting performances to TFRRS, the meet will be unpublished until performances are submitted to it. An unpublished meet is identified by its "unpublished_meet_hnd" as opposed to a meet_hnd. These two types of identifiers are not interchangeable.

> Note: The two arrays described above are also returned by the meets and unpublished_meets methods. By including these lists in the login response you are saved two additional SOAP requests. But you may wish to allow your users to refresh their meet lists once they are logged in. Call meets or unpublished_meets with no arguments (except Credentials – which must have a session_id) to get an updated meets or unpublished_meets array.

Submitting Results

Now that the user is logged in and has a list of meets (and unpublished meets) he may submit results. The submit_results method takes two elements:

- *Either* a meet or an unpublished meet.
- An array of teams.

The meet or unpublished meet need only contain the sport ("track" or "xc") and the meet_hnd or unpublished_meet_hnd, as in:

```
<meet><meet_hnd>13590</meet_hnd><sport>track</sport></meet>
```

Or

```
<unpublished_meet>
<unpublished_meet_hnd>1985</unpublished_meet_hnd><sport>track</sport>
</unpublished_meet>
```

Team Division Scoring

If the meet was scored separately per-division, with divisions assigned to teams and athletes, the `team_division_scoring` flag in the `<meet>` element should be set to 1:

```
<meet><meet_hnd>13590</meet_hnd><sport>track</sport>
<team_division_scoring>1</team_division_scoring></meet>
```

Teams array

The teams array is much more complicated. Each team must contain a name. The team may also contain a TFRRS team code:

```
<teams>
<item><name>My Team</name><team_code>MYTEAM</team_code>
<division>Class AA</division>……
```

In addition it must contain one or both of two arrays, "roster" and "relays".

Team Division: If team division scoring is used, the division name for this team should be included as it is in the example above.

The roster Array:

Each item in the roster array is an athlete. An athlete must contain

- first name
- last name
- gender ("m" or "f")
- bib *or* id
- results (an array)

And may additionally contain:

- year – either year of graduation, grade number or FR,SO, JR, SR
- exhibition – a boolean (true or false)
- division – if team division scoring is used and this athlete has a division assignment overriding the team division.

bib or id: TFRRS identifies athletes using a TFRRS ID. A TFRRS ID is a string like "045621DAC*IVY*". A TFRRS ID is created when an athlete is added to TFRRS by his/her coach. Additionally, TFRRS can identify an athlete using a DirectAthletics ID which looks like "28118dx1051b". If the athlete does not have a TFRRS ID or DirectAthletics ID, his/her performances will not be submitted to TFRRS. However, a unique identifier is still required. Instead of using id, put a unique number for this athlete in this meet in bib. As long as the bib number is used only for this athlete in this meet it doesn't matter what numbering scheme is used.

The Results Array (Track & Field)

The results array contains the results of this athlete in this meet. For track & field meets, each result contains:

- round - one of "P", "Q", "S", "F" (prelims, quarter final, semi, final)
- place - place overall in the event
- heat - heat or flight number
- hplace - place in the particular heat or flight
- score - the score for this event
- mark - time in seconds, height/distance in meters or inches, or points for multi-events. Can also be DNS, DNF, FOUL, FS, or DQ.
- metric - 1 if the mark is metric, 0 otherwise (field events only)
- event - an element describing the event
- fat - 1 if automatic timing used, otherwise 0
- wind - in meters per second ("-1.2","+3.2"). Do not include if no wind reading.
- attempts (an array)

attempts: You may submit an array of attempts, in order. Each item in the array may contain:

  - mark- meters, inches, FOUL or PASS
  - metric – 1 if the mark is meters, 0 if inches.
  - wind (do not include if no wind reading)
  - bar – height of the bar in meters or inches (for vertical jumps)
  - bar_metric – 1 if bar height is meters, otherwise 0

event: Each result has an event element describing this event. The event element contains:

- code - a code indicating what kind of event this is
- gender - "m" or "f"
- division - an integer, used only if there are multiple events of the same code & gender.
- min_age - the minimum age if this is an age group meet.
- max age - the maximum age if this is an age group meet.
- name - this name will be used when displaying the results.

code: TFRRS identifies types of events by a string called "code". The list of codes is in Appendix 1.

gender: TFRRS uses "m" or "f" to indicate gender.

division: if there is more than one event of the same code and gender, these events must be distinguished using a division number.

min age and max age: For age group meets.

name: If you include a name element that name will be used for that event. Otherwise TFRRS will just use the gender and type for the event name.

Example of the beginning of the teams array:

```
<teams>
<item>
<name>My Team</name>
<team_code>MYTEAM</team_code>
<roster>
<item>
<first_name>Bob</first_name>
<last_name>Smith</last_name>
<gender>m</gender>
<id>x22ee21-32b4</id>
<results>
<item>
<place>8</place>
<round>F</round>
<score>.00</score>
<event>
<name>Men 1 Mile Run</name>
<gender>m</gender>
<code>1mile</code>
</event>
<mark>266.36</mark>
<heat >2</heat>
<fat >1</fat>
<hplace>3</hplace>
</item>
</results>
[more athletes for this team would follow.]</team>[More teams would follow this team]</teams
```

Multi-Events

When a track & field result is for an event within a multi-event (decathlon, etc.) there is an additional element called "sub_event". This is just like the "event" element but it only contains "code" and "gender". Also, a result for a sub-event does not have a "score" element. Instead there is a "points" element.

Example of a 100m Dash result within a Heptathlon:

<item>
<metric>1</metric>
<place>15</place>
<**points**>501</**points**>
<round>F</round>
<event>
<name>Women **Heptathlon**</name>
<gender>f</gender>
<code>heptathlon</code>
</event>
<**sub_event**>
<code>100</code>
<gender>f</gender>
</**sub_event**>
<mark>17.88</mark>
<wind>1.2</wind>
<heat >6</heat>
<fat>1</fat>
<hplace>4</hplace>
</item>

Note that an athlete's overall result for a multi-event has no sub-event.

The relays Array

The relays array contains all the relay results for a team. Each item in a relays array contains:

- place
- round
- score
- mark
- heat
- fat
- hplace
- event
- athletes (array)

All the elements except "athletes" are the same as in an athlete result. The athletes array contains the relay athletes, in order. Each athlete in this array needs only the bib or id. It is not necessary to repeat the first name, last name or gender (assuming that athlete has an entry in the roster array).

An example relay result:

<item>
<place>5</place>
<round>F</round>
<score>.00</score>
<mark>54.74</mark>
<heat>1</heat>

```
<fat>1</fat>
<hplace>5</hplace>
<event>
<gender>f</gender>
<code>4x100</code>
</event>
<athletes >
<item>
<bib> 58084</bib>
</item>
<item>
<bib>58086</bib>
</item>
<item>
<bib>58074</bib>
</item>
<item>
<bib>58056</bib>
</item>
</athletes>
</item>
```

Unattached Athletes

All results for unattached athletes should be in a team named "Unattached" with a team_code of "UNA".

Submit Your Results

When you have created your teams array, submit that, along with your meet or unpublished meet to submit_results. A complete submit_results request is in Appendix 2.

If your submitted results are accepted the response will contain:

- meet_hnd (if you submitted an unpublished meet, this meet_hnd represents a newly created meet)
- results – the total number of results found
- relays - the total number of relay found
- athletes – the total number athletes found
- matched_athletes – the number of athletes with IDs matching TFRRS or DirectAthletics IDs.

If you submit results using the Demo vendor your results will not actually be imported into TFRRS but the values above should be helpful for development/testing.

VI. Downloading Rosters From The SOAP Service

Meet hosts can download roster information from TFRRS regardless of the online entry provider used. Meets hosts can download roster data from the TFRRS website in CSV format or from a meet manager using the get_rosters function.

Before downloading rosters in either CSV format or via get_rosters, a meet host must first set up the meet on TFRRS and select which rosters are to be downloaded. Any meet host can create an account for setting up meets on www.tfrrs.org.

get_rosters

The get_rosters function takes one argument, "unpublished_meet_hnd". This argument should refer to an unpublished meet in the user's list.

The response from get_rosters contains a meet element and teams array.
The meet element contains only the name of the meet.

Each team in the teams array contains:

- name
- team_code
- roster (an array)

Each athlete in the roster array contains:

- first_name
- last_name
- middle_initial
- gender ("m" or "f")
- year
- id (this is a *one time use id*)

The ID for an athlete in a get_rosters response is valid for submitting performances for this meet only. Include this ID in the athlete entitle in a submit_results request.

VII. Downloading Entries From The SOAP Service

Users who collect entries via DirectAthletics may download entries via the TFRRS API get_entries method. Implementing support for this method will increase the utility of your software for DirectAthletics users.

get_entries

The get_entries method takes one argument, meet_hnd (in addition to a Credentials element with a valid session, client_key and vendor). The meet_hnd must identify a meet in the user's meet array.

The response from get_entries contains a meet element and a teams array.

The meet element contains:

- meet_hnd
- name
- date_end
- date_begin
- registration_begin
- registration_end
- events (an array)
- venue (an element containing the name and state of the track)
- divisions (an array)

The events array contains events in the same format as in submit_results. If the meet is configured in DirectAthletics to use divisions, the divisions array contains a list of divisions used in the meet. Each division contains:

- name
- division (an integer)

The teams Array

Each team in the teams array contains:

- name
- team_code (a unique TFRRS code for this team)
- town
- state
- roster (an array)
- relays (an array)

The roster Array

Each athlete in the roster array contains:

- first_name

- last_name
- year – the year in school of this athlete
- dob – date of birth (YYYY-MM-DD)
- id – the TFRRS or DirectAthletics ID
- gender
- entries (an array)

The entries Array

Each item in an athlete's entries array contains:

- event
- seed
- date
- metric (for field events: true if seed is metric, false if English)
- note

seed: The seed is in seconds for a timed event. For fields events the seed is in meters if metric=1 or inches if metric=0. For multi-events, the seed is points.

note: A string indicating where the seed was achieved.

Date: The date the seed was achieved, formatted like YYYY-MM-DD.

event: The event in an entry is the same as the event in a result. It contains:

  - code
  - gender
  - division
  - min_age
  - max_age

The relays Array

Each item in the relays array contains:

- squad – a letter
- seed – in seconds
- date – the date the seed was achieved (YYYY-MM-DD)
- note – a string indicating where the seed was achieved
- athletes (an array)
- event – same as in submit_results (code, gender, division, min_age, max_age)

Each item in the athletes array of a relay contains:

  - relay_order – an integer indicating the leg this athlete will run
  - id – the TFRRS or DirectAthletics ID of this athlete
  - gender ("m" or "f")

- first_name
- last_name

An athlete appearing in a relay event will also be in the roster array.

Unattached Entries:

Unattached entries will be in a team with a name of Unattached and a team_code of "UNA".

VIII. REST

Developers who prefer a REST interface will find one at:

http://www.tfrrs.org/trackrest

To use the REST interface, put the method name after the URL above, as in:

http://www.tfrrs.org/trackrest/get_entries

Put all Credentials elements in CGI parameters, as in:

http://www.tfrrs.org/trackrest/get_entries?vendor=Demo&version=1&client_key=yHbExEJnaVXlSFYZlfEmwg&session_id=2720416-470627

You may need to escape the client key.

All other parts of the request are sent via POST. A POST request allows the party making the request to send data to the server (in addition to CGI parameters).

The POST data should contain the same XML data that the equivalent SOAP request would have, minus the Credentials, the encapsulating Envelope and Body tags, and the method name tag. So for a REST request to get_entries, send this data in POST:

<meet_hnd>13585</meet_hnd><sport>track</sport>

For submit_results, the entire <teams> XML would be sent in POST.

The response from the REST server is the same as the response from the SOAP server, minus the encapsulating Envelope, body and method response tags. So for get_entries, the <teams> XML would be the entire response from the REST server.

Appendix 1: Track & Field Event Codes

Code	Event Type
40	40m Dash
40y	40y Dash
45y	45y Dash
50y	50y Dash
50	50m Dash
55	55m Dash
60	60m Dash
150	150m Dash
60y	60y Dash
70	70m Dash
75y	75y Dash
80	80m Dash
100y	100y Dash
100	100m Dash
120y	120y Dash
200	200m Dash
220y	220y Dash
300	300m Dash
300y	300y Dash
400	400m Dash
440y	440y Dash
500	500m Dash
600	600m Run
600y	600y Run
800	800m Run

880y	880y Run
1000	1,000m Run
1000y	1000y Run
1200	1,200m Run
1500	1,500m Run
1600	1,600m Run
1mile	1 Mile Run
2000	2,000m Run
2400	2,400m Run
3000	3,000m Run
3200	3,200m Run
2mile	2 Mile Run
3mile	3 Mile Run
5000	5,000m Run
10000	10,000m Run
1500rw	1500m Race Walk
6000	6000m Run
1600rw	1600m Race Walk
1milerw	1 Mile Race Walk
8000	8000m Run
3000rw	3000m Race Walk
5000rw	5000m Race Walk
45yH	45y Hurdles
50H	50m Hurdles
50yH	50y High Hurdles
55H	55m High Hurdles
60H	60m High Hurdles
60yH	60y High Hurdles
65H	65m Hurdles

70H	70m Hurdles
75	75m Hurdles
77h	77m Hurdles
80h	80m Hurdles
100h	100m High Hurdles
110h	110m High Hurdles
110yh	110y Hurdles
120yh	120y Hurdles
220h	220m Hurdles
300h	300m Intermediate Hurdles
330yh	330y Hurdles
400h	400m Intermediate Hurdles
200h	200m Hurdles
900h	900m Hurdles
800steeple	800m Steeplechase
1500steeple	1500m Steeplechase
1600steeple	1,600m Steeplechase
2000steeple	2,000m Steeplechase
4x50	4 x 50m Relay
3000steeple	3,000m Steeplechase
4x60	4 x 60m Relay
4x55	4 x55m Relay
4x100	4 x 100m Relay
4x110y	4 x 110y Relay
4x160	4 x 160m Relay
4x220	4 x 220y Relay
4x236	4 x 236m
4x200	4 x 200m Relay
4x400	4 x 400m Relay

4x440y	Mile Relay
4x800	4 x 800m Relay
200smr	200m Sprint Medley
800smr	800m Sprint Medley
880ysmr	880y Sprint Medley
4x880y	2 Mile Relay
4x1200	4 x 1200m Relay
4x1500	4 x 1,500m Relay
3x100SH	3x100 Shuttle Hurdle
3x1600	3 x 1600m Relay
3x300SH	3 x 300m Shuttle Hurdle
3x55SH	3 x 55 Shuttle Hurdle
4x100sh	4 x 100m Shuttle Hurdle Relay
3x110SH	3 x 110m Shuttle Hurdle
3x140sh	3x140 Shuttle Hurdle
4x50SH	4 x 50m Shuttle Hurdle Relay
4x55SH	4 x 55m Shuttle Hurdle Relay
4x60SH	4 x 60m Shuttle Hurdle Relay
4x65SH	4 x 65m Shuttle Hurdle
4x77sh	4 x 77m Shuttle Hurdle
4x94sh	4 x 94m Shuttle Hurdle
4x70SH	4 x 70m Shuttle Hurdle
4x110sh	4 x 110m Shuttle Hurdle Relay
4x1600	4 x 1,600m Relay
4x150sh	4 x150m Shuttle Hurdle Relay
4x160sh	4 x 160m Shuttle Hurdle
4x1mile	4 x 1 Mile
4x64SH	4 x 64m Shuttle Hurdle
4x50ySH	4 x 50 Yard Shuttle Hurdle Relay

4x60ySH	4 x 60 Yard Shuttle Hurdle Relay
1200smr	1200m Sprint Medley
1600smr	1600m Sprint Medley
900smr	900m Sprint Medley
960smr	960m Sprint Medley
4x1000	4 x 1000m Medley Relay
4x3200	4 x 3200m Relay
1000smr	1000m Sprint Medley
1500smr	1500m Sprint Medley
800dmr	800m Distance Medley
2000dmr	2000m Distance Medley
2600dmr	2600m Distance Medley
3000dmr	3000m Distance Medley
3200dmr	3,200m Distance Medley
4000dmr	4000m Distance Medley
10y	10y Dash
4800dmr	4800m Distance Medley
1600dmr	1600m Distance Medley
4400ydmr	4400y Distance Medley
1120smr	1120m Sprint Medley
hj	High Jump
pv	Pole Vault
lj	Long Jump
tj	Triple Jump
sp	Shot Put
dt	Discus Throw
ht	Hammer Throw
wt	Weight Throw
jav	Javelin

pentathlon	Outdoor Pentathlon
ipentathlon	Indoor Pentathlon
heptathlon	Heptathlon
decathlon	Decathlon
wpentathlon	Weight Pentathlon
triathlon	Triathlon
tetrathlon	Tetrathlon
10000rw	10,000m Race Walk
slj	Standing Long Jump
softball	Softball Throw
8x55y	8 x 55y Relay
halfmarathon	Half Marathon
4x150	4 x 150m Relay
marathon	Marathon
195h	195 Hurdles

Appendix 2: A Complete submit_results Request

Below are SOAP Requests for each step of submitting results.

get_client_key:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<get_client_key xmlns="urn:TFRRS">
<Credentials>
<version xsi:type="xsd:float">1.0</version>
<vendor xsi:type="xsd:string">Demo</vendor>
</Credentials>
</get_client_key>
</soap:Body>
</soap:Envelope>
```

validate_client_key:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<validate_client_key xmlns="urn:TFRRS">
<client_key xsi:type="xsd:string">RsYe+t3s1eslytfdVVJB5A</client_key>
<Credentials>
<version xsi:type="xsd:float">1.0</version>
<vendor xsi:type="xsd:string">Demo</vendor>
</Credentials>
<validation
xsi:type="xsd:string">01112179aa6da4db9ccf63bb78c4719f</validation>
</validate_client_key>
</soap:Body>
</soap:Envelope>
```

login:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<login xmlns="urn:TFRRS">
<password xsi:type="xsd:string">test</password>
```

```
<Credentials>
<client_key xsi:type="xsd:string">RsYe+t3s1eslytfdVVJB5A</client_key>
<version xsi:type="xsd:float">1.0</version>
<vendor xsi:type="xsd:string">Demo</vendor>
</Credentials>
<username xsi:type="xsd:string">tfrrsdemo1</username>
</login>
</soap:Body>
</soap:Envelope>
```

submit_results:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<submit_results xmlns="urn:TFRRS">
<Credentials>
<client_key xsi:type="xsd:string">RsYe+t3s1eslytfdVVJB5A</client_key>
<session_id xsi:type="xsd:string">2718481-863196</session_id>
<version xsi:type="xsd:float">1.0</version>
<vendor xsi:type="xsd:string">Demo</vendor>
</Credentials>
<meet>
<meet_hnd xsi:type="xsd:int">13590</meet_hnd>
<sport xsi:type="xsd:string">track</sport>
</meet>
<teams soapenc:arrayType="xsd:anyType[7]" xsi:type="soapenc:Array">
<item>
<relays soapenc:arrayType="xsd:anyType[4]" xsi:type="soapenc:Array">
<item>
<place xsi:type="xsd:int">1</place>
<round xsi:type="xsd:string">F</round>
<athletes soapenc:arrayType="xsd:anyType[6]" xsi:type="soapenc:Array">
<item>
<bib xsi:type="xsd:int">2212</bib>
</item>
<item>
<bib xsi:type="xsd:int">2166</bib>
</item>
<item>
<bib xsi:type="xsd:int">2211</bib>
</item>
<item>
<bib xsi:type="xsd:int">2210</bib>
</item>
<item>
<bib xsi:type="xsd:int">2169</bib>
</item>
<item>
<bib xsi:type="xsd:int">2192</bib>
</item>
</athletes>
```

```
<score xsi:type="xsd:float">.00</score>
<event>
<max_age xsi:type="xsd:int">109</max_age>
<min_age xsi:type="xsd:int">0</min_age>
<gender xsi:type="xsd:string">f</gender>
<code xsi:type="xsd:string">4x400</code>
</event>
<mark xsi:type="xsd:float">250.42</mark>
<heat xsi:type="xsd:int">1</heat>
<fat xsi:type="xsd:int">1</fat>
<hplace xsi:type="xsd:int">1</hplace>
</item>
<item>
<heat xsi:type="xsd:int">1</heat>
<mark xsi:type="xsd:float">0.00</mark>
<fat xsi:type="xsd:int">1</fat>
<round xsi:type="xsd:string">F</round>
<athletes soapenc:arrayType="xsd:anyType[5]" xsi:type="soapenc:Array">
<item>
<bib xsi:type="xsd:int">2192</bib>
</item>
<item>
<bib xsi:type="xsd:int">2211</bib>
</item>
<item>
<bib xsi:type="xsd:int">2214</bib>
</item>
<item>
<bib xsi:type="xsd:int">2173</bib>
</item>
<item>
<bib xsi:type="xsd:int">2171</bib>
</item>
</athletes>
<score xsi:type="xsd:float">.00</score>
<event>
<max_age xsi:type="xsd:int">109</max_age>
<min_age xsi:type="xsd:int">0</min_age>
<gender xsi:type="xsd:string">f</gender>
<code xsi:type="xsd:string">4x400</code>
</event>
</item>
</relays>
<roster soapenc:arrayType="xsd:anyType[56]" xsi:type="soapenc:Array">
<item>
<bib xsi:type="xsd:int">2180</bib>
<id xsi:type="xsd:string">x22fff2-760</id>
<results soapenc:arrayType="xsd:anyType[1]" xsi:type="soapenc:Array">
<item>
<metric xsi:type="xsd:int">1</metric>
<place xsi:type="xsd:int">16</place>
<round xsi:type="xsd:string">F</round>
<score xsi:type="xsd:float">.00</score>
<event>
<max_age xsi:type="xsd:int">109</max_age>
<division xsi:type="xsd:int">0</division>
<name xsi:type="xsd:string">Men 3000 Meter Run</name>
```

```
<min_age xsi:type="xsd:int">0</min_age>
<gender xsi:type="xsd:string">m</gender>
<code xsi:type="xsd:int">3000</code>
</event>
<mark xsi:type="xsd:float">610.88</mark>
<heat xsi:type="xsd:int">1</heat>
<fat xsi:type="xsd:int">1</fat>
<hplace xsi:type="xsd:int">16</hplace>
</item>
</results>
<gender xsi:type="xsd:string">m</gender>
<last_name xsi:type="xsd:string">Morgan</last_name>
<first_name xsi:type="xsd:string">Ethan</first_name>
</item>
<item>
<bib xsi:type="xsd:int">2199</bib>
<id xsi:type="xsd:string">x23007e-761</id>
<results soapenc:arrayType="xsd:anyType[2]" xsi:type="soapenc:Array">
<item>
<metric xsi:type="xsd:int">1</metric>
<place xsi:type="xsd:int">11</place>
<round xsi:type="xsd:string">P</round>
<score xsi:type="xsd:float">.00</score>
<event>
<max_age xsi:type="xsd:int">109</max_age>
<division xsi:type="xsd:int">0</division>
<name xsi:type="xsd:string">Women 60 Meter Dash</name>
<min_age xsi:type="xsd:int">0</min_age>
<gender xsi:type="xsd:string">f</gender>
<code xsi:type="xsd:int">60</code>
</event>
<mark xsi:type="xsd:float">9.01</mark>
<wind xsi:type="xsd:float">-0.0</wind>
<heat xsi:type="xsd:int">1</heat>
<fat xsi:type="xsd:int">1</fat>
<hplace xsi:type="xsd:int">6</hplace>
</item>
<item>
<metric xsi:type="xsd:int">1</metric>
<place xsi:type="xsd:int">2</place>
<round xsi:type="xsd:string">F</round>
<score xsi:type="xsd:float">.00</score>
<event>
<max_age xsi:type="xsd:int">109</max_age>
<division xsi:type="xsd:int">0</division>
<name xsi:type="xsd:string">Women Long Jump</name>
<min_age xsi:type="xsd:int">0</min_age>
<gender xsi:type="xsd:string">f</gender>
<code xsi:type="xsd:string">lj</code>
</event>
<mark xsi:type="xsd:float">4.34</mark>
<heat xsi:type="xsd:int">1</heat>
<fat xsi:type="xsd:int">1</fat>
<hplace xsi:type="xsd:int">2</hplace>
</item>
</results>
<gender xsi:type="xsd:string">f</gender>
```

```
<last_name xsi:type="xsd:string">Darrow</last_name>
<first_name xsi:type="xsd:string">Jennie</first_name>
</item>
</roster>
</teams>
</submit_results>
</soap:Body>
</soap:Envelope>
```

Appendix 3: Changes To This Document

1.8:

- Cross Country Reporting Support:
    - XML objects of type “meet” and “unpublished_meet” now contain a “sport” element. Valid values are “track” or “xc.” *If sport is omitted, “track” is assumed.*
    - Cross Country CSV Format
    - CSV documentation moved ahead of SOAP/REST interface documentation.
- Team divisions support for track & field