RegattaCentral API V3.0 Cookbook

Contents

1. **Getting Started** 1
2. **Authentication** 2
3. **Transformers** 2
4. **Downloading Regatta Information** 3
5. **Organization Search** 4
6. **Participant Search** 4
7. **Uploading Data** 4
8. **Flush/purge race schedule, race status and race results** 5
9. **Creating and Modifying Events** 5
10. **Creating a New Entry** 5
11. **Race information** 5
12. **Timing Milestones and Results** 6
13. **Lane Draw** 6

Appendix A Example Payloads 8

1. Getting Started

Welcome to the RegattaCentral V3.0 Cookbook. This document describes how to utilize the RegattaCentral ™ API V3.0 to effectively exchange data with RegattaCentral. This allows a timing system to efficiently retrieve the latest registration and regatta information from RegattaCentral and it allows the timing system to report changes and results to RegattaCentral so that races and results can be viewed on RegattaCentral™ by coaches, athletes and spectators moments after the timing system publishes them. RegattaCentral™ allows viewing of these results through mobile device apps and standard web pages.

Complete details regarding the data structures, as well as example requests, can be found at

https://api.regattacentral.com/apiV3.jsp

Results data is provided to RegattaCentral in a “cooked” format – it is up to the timing system to calculate handicaps, penalties, adjusted times and margins and supply labels for timing locations. RegattaCentral reports whatever the timing system supplies so that there is no chance of discrepancy with other results reports, such as timing system generated PDF files.

The timing system will retrieve and import the regatta information. How this entry data is loaded by the timing system will vary. Once loaded, it is assumed that the timing system will maintain and use the specified Event, Entry and Athlete IDs. However, this interface does allow the timing system to add events, entries and athletes on the fly. When the timing system creates a new entity, it must assign a UUID and use this UUID whenever that entity is referenced.   RegattaCentral will use this UUID to create, assign and subsequently locate an internal ID. Whenever a tag has both a non-zero ID and UUID, the ID will be used to reference the RegattaCentral data and the UUID will be ignored.

The timing system needs to obtain an OAUTH2 token from the RegattaCentral API. This uses pre-allocated client ID and client secret in addition to the user's RegattaCentral login credentials. The user must have "staff" access to the regatta on RegattaCentral.

Data may be transferred using the XML or JSON.   Tags may be used multiple times to so that all new data can be transferred at the same time.   Ultimately, all of the data for a regatta could be transferred in a single upload.

The "trigger" to upload data is left up to the timing system.   The timing system could wait and batch the uploads all at once, or after each race, or they can be uploaded every time there is a change in data.   It is important to keep in mind that the more frequently the data is uploaded, the more "realtime" feeling will be sensed by users.   This is especially true for courses that have mid-race split times.

The latest, up to date schema (rc-api.xsd) is available on the API documentation page and can be used to generate classes/methods/routines in your favorite language.

2. Authentication

Obtaining an OAUTH2 token that can be supplied with subsequent requests is performed by a simple HTTP request:

```
$.ajax({ type: "POST",
        url: 'https://api.regattacentral.com/oauth2/api/token',
        data: 'client_id={client_id}&client_secret={client_secret}&username={username}
        &password={password}&grant_type=password',
        success: function (data) { // Process & Display Data },
        error: function () { alert('Error processing request.'); }
});
```

Timing systems that utilize JAVA can contact RegattaCentral support for a utility class which will help them create the HTTP requests.

The token is saved and provided as a HTTP header "Authorization" in all subsequent calls to the API.

The token request also returns an expiration time and a refresh token. You may utilize the refresh token to obtain a new token after the expiration without having to re-prompt the user for his credentials.

3. Transformers

From a programming standpoint, the timing system should use a set of transformer methods which convert data between the RegattaCentral API data model and the timing system's data model.   Because the API data model is hierarchical, every set of race results needs all of the layers above to provide the context. You do *not* have to provide all of the data for every element every time.   As is specified in the schema, the element's ID is usually the only required element. (If the ID is 0, then the UUID must be specified).

Transformers will allow you to easily build each message.

By the same token, transformer methods are needed to process each element during a “/bulk” data transfer which will populate the timing system’s data model from the RegattaCentral model.

A JAVA example of a LaneConstructor is shown below.   It extends the Lane object which was generated from the rc-api.xsd and fills in the data provided by the timing system.

```
public class LaneConstructor extends Lane {

      public LaneConstructor(int laneNumber, long handicapTime,
                              RacingCrewStatusType racingCrewStatusType,
                              int crewId, String crewUuid) {

            this.lane = laneNumber;
            this.displayNumber = Integer.toString(laneNumber);
            this.entryId = crewId;
            this.uuid = crewUuid;
            this.handicap = handicapTime;

            switch (racingCrewStatusType) {
                  case SCRATCHED_LITERAL : this.status = ResultStatusType.SCR;  break;
                  case DISQUALIFIED_LITERAL : this.status = ResultStatusType.DQ; break;
                  case DID_NOT_FINISH_LITERAL : this.status = ResultStatusType.DNF; break;
                  case EXCLUDED_LITERAL : this.status = ResultStatusType.EXC; break;
                  case EXHIBITION_LITERAL : this.status = ResultStatusType.EXH; break;
                  case ELIGIBLE_LITERAL :
                  case FINISHED_LITERAL :
                  case PROTESTING_LITERAL :
                  default:
            }
      }
}
```

```
The object created by this constructor can then easily be added to the <Race> object that
you have created (using its own constructor).
```

4. Downloading Regatta Information

Regatta information (including events, entries, athletes, etc) can be downloaded from the RegattaCentral server using the endpoint:

https://api.regattacentral.com/v3.0/regatta/{RegattaId}/bulk

The RegattaId is the RegattaCentral ID that is unique to the specific regatta. You must supply a valid token and the user account that created the token must have staff privileges to access the regatta.

It is up to the timing system to process this information into the timing system data model, but it is imperative that the timing system maintain the various IDs provided in the structures. These uniquely identify each piece of information and will be utilized during data uploads.

Thought must be given to when this data is loaded and how to handle changes to the data. At this time, there is no way to provide only information updated after a certain time. The timing system must decide how it will SYNC data over time. If changes are allowed to be made on both the timing system and RegattaCentral at the same time, users will need some way to view any conflicting data and resolve which is the correct data to move forward with.   Alternatively, if no entry data is changed on the timing, then the RegattaCentral data can be reloaded into the timing system on subsequent downloads.

5. Organization Search

The timing system can perform organization lookups so that when adding a new entry to the timing system, the correct RegattaCentral organization ID is supplied to RegattaCentral. This is a partial name search of the full organization's name, the short name and the club abbreviation.

https://api.regattacentral.com/v3.0/search/organization/{Term}

6. Participant Search

The timing system can perform athlete lookups so that replacement athletes can be used from the RegattaCentral databases.   This is a partial name search (uses SQL "like" with wildcards after the supplied strings) for all athletes with a given birthdate.

https://api.regattacentral.com/v3.0/search/participant/{Term}/{Birthday}

This call searches the RegattaCentral athlete database and returns data regarding the athlete.   The timing system can then display these names and allow the user to select one of them to place the athlete in the lineup.

The last name (Term) must be at least 2 characters. The birthdate must be in the exact format "yyyy-mm-dd".

7. Uploading Data

All upload (timing system to RegattaCentral) data transfers are performed using XML or JSON semantics. This is performed using a POST – for example..

```
DefaultHttpClient client = new DefaultHttpClient();
HttpPost post = new HttpPost("http://localhost/regatta/v3/{RegattaID}/upload");
post.setHeader("Authorization", token);
StringEntity input = new StringEntity(xml);
input.setContentType("text/xml; charset=ISO-8859-1");
post.setEntity(input);  // the actual XML

HttpResponse resp = client.execute(post);
if (resp.getStatusLine().getStatusCode() == HttpStatus.SC_OK) {
      return true;
}
writeRCConsole(resp.toString());
return false;
```

See the Appendix for sample XML payloads.

8. Flush/purge race schedule, race status and race results

Data provided by the timing system always supersedes earlier data. If an individual piece of data needs to be changed, simply re-issue that information and the old data will be overwritten.

There are times when the timing system needs to start fresh (such as a major change to the race schedule). In this case, all results records for the regatta may be flushed from RegattaCentral using the '`<flush>true</flush>`' tag.

This will cause ONLY the race schedule, draw/lane assignments and race results to be flushed. It does not affect any regatta information nor any entry data.

The '`<flush>true</flush>`' tag can also be specified for a specific race. This will clear the draw and results data for the specified race only. This is often used when a race status is being changed to "Official". To insure consistency of data, when a race status is changed to Official, you can flush the existing results data and upload the entire race data in a single upload.

9. Creating and Modifying Events

Race day operations often include the need to create new events (or splitting an event into 2 events). This can be done by issuing the <event/> tag. It is important that the event_id be set to 0 and that a UUID be created by the timing system for this event. All subsequent references to this event must be made using the UUID. This UUID must also be used when uploading entries and races for this event.

10.Creating a New Entry

Entire entries can be added by the timing system in the same way that events are. The timing system can allow the user to lookup athletes and define the title. When the timing system "Saves" the entry, an XML request should be issued that uses a crew ID of 0 and a UUID. Again, this UUID must be used by all subsequent references to this entry. If this was also a new event, then the eventID would be set to 0 and the UUID would be used to locate the event.

11. Race information

Races and results are maintained using the race_id which is maintained on the timing system. The race is the central piece of data for the results displays.

Each time the race record is sent to RegattaCentral, the data overwrites the existing record. Some data is fairly static but other pieces of data may change. Perhaps the race was shortened to 1500m. Or you want to updated the conditions on the course. But the most important piece of information for results viewing is the status.

The list of possible race statuses is listed below. Regularly updating this status as the regatta progresses is important as each change in status triggers highlights on many of the display pages. After the regatta data is downloaded to the timing system and the race schedule is established, you can publish the schedule with a "Pre-Draw" status. Once the draws have been published, you can set the status to "Draw" so that users can be notified to look up their lane assignments. Setting the status to “Racing” is important to real-time users and should be done when the race actually starts.

```
“PreDraw”,
“Draw”,
“Racing”,
“Stopped”,
“StoppedCollision”,
“StoppedFalseStart”
“StoppedStartZoneDamage”,
“Unofficial”,
“Official”,
“Protest”,
“RemovedCancelled”,
“RemovedConsolidated”,
“RemovedNonEvent”,
“Unknown”
```

12. Timing Milestones and Results

RegattaCentral results displays are currently focused on allowing up to 3 mid-course splits, plus the finish line. So together with the starting line, RegattaCentral supports 5 Timing Milestones. Each milestone can have its own distance and title, but it is important to understand that timingMilestoneId = 0 is always the starting line and timingMilestoneId = 4 is always the finish line.

The starting line milestone establishes the lane draw using the <Lane> tag. The information included with <Lane> will be used to establish the Milestone 0 (starting line) record. No times are recorded with the Lane assignments.

Regardless of how many mid-course splits are in use, timingMilestoneID = 4 is the finish line result.

Every <Result> record must specify the timingMilestoneId.

The <result> record transfers the actual timing data. Note that the crew and event information are NOT included in this record. They key is the race number and lane number. Any lane changes must be reported via <lane> prior to reporting a result.

The <time> attribute is used to report the cumulative time. <splitTime> is used to report the time since the last split location. The <adjusted Time> reports the cumulative time with handicaps and penalties included.

A time of 0 will clear all of the timing field for this race/lane/split.

With multiple races in progress, races and results can be combined into single HTTP requests, or they can be serialized by the timing application.

13. Lane Draw

Lane draws associate crew entries with particular races. It would be common for all of the lanes for a race to be saved at the same time and uploaded together. The lane draw/bow numbers are assigned to each boat

A status may be optionally supplied to indicate the status of this entry – such as:

“SCR”	Scratched
“DNS”	Did Not Start
“DNF”	Did Not Finish
“DSQ”	Disqualified
“RMV”	Removed
“EXC”	Excluded
“NJ”	Not Judged
“INV”	By Invitation

The lanes MUST be reported prior to reporting results.

Appendix A Example Payloads

A1. Flush all Race and Result data

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<regattas xmlns="http://www.regattacentral.com/model">
      <version>3.0</version>
      <timestamp>2015-06-25T13:30:01.987-07:00</timestamp>
      <source>www.regattacentral.com</source>
    <regatta>
        <jobId>3939</jobId>
        <flush>true</flush>
       <name>Anytown API Demo Regatta</name>
    </regatta>
</regattas>
```

A2. Add a race to an event using a UUID

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<regattas xmlns="http://www.regattacentral.com/model">
      <version>3.0</version>
      <timestamp>2014-10-29T10:59:33.462-07:00</timestamp>
      <source>www.regattacentral.com</source>
      <regatta>
            <jobId>3939</jobId>
            <events>
                  <eventId>12</eventId>
                  <races>
                        <raceId>0</raceId>
                        <uuid>123-121111-22348</uuid>
                        <display_number>112</display_number>
                        <progression>Finals Only</progression>
                  </races>
            </events>
      </regatta>
</regattas>
```

A3. Add one event and modify another event (typical if an event is being split by age category)

```
<regattas xmlns="http://www.regattacentral.com/model">
<version>3.0</version>
<timestamp>2014-10-29T10:59:33.462-07:00</timestamp>
<source>www.regattacentral.com</source>
      <regatta>
            <jobId>3939</jobId>
            <events>
                  <eventId>0</eventId>
                  <uuid>123-121111-22356</uuid>
                  <title>Mixed Master 8x D-F</title>
                  <sequence>0</sequence>
                  <label>1</label>
                  <code/>
                  <gender>both</gender>
```

```
                    <minAthleteAge>0</minAthleteAge>
                    <maxAthleteAge>0</maxAthleteAge>
                    <minAvgAge>0</minAvgAge>
                    <maxAvgAge>0</maxAvgAge>
                    <athleteClass/>
                    <athleteCountExcludingCox>4</athleteCountExcludingCox>
                    <sweep>false</sweep>
                    <coxed>true</coxed>
                    <cost>0.0</cost>
                    <defaultRaceUnits>meters</defaultRaceUnits>
                    <maxEntriesPerClub>0</maxEntriesPerClub>
                    <reqCoxswain>false</reqCoxswain>
                    <reqSeeds>true</reqSeeds>
                    <lateFee>0</lateFee>
                </events>
                <events>
                    <eventId>2</eventId>
                    <sequence>2</sequence>
                    <label>1</label>
                    <title>Mixed Masters 8x A-C</title>
                    <code/>
                    <gender>both</gender>
                    <minAthleteAge>0</minAthleteAge>
                    <maxAthleteAge>0</maxAthleteAge>
                    <minAvgAge>0</minAvgAge>
                    <maxAvgAge>0</maxAvgAge>
                    <athleteClass/>
                    <athleteCountExcludingCox>2</athleteCountExcludingCox>
                    <cost>0.0</cost>
                    <defaultRaceUnits>meters</defaultRaceUnits>
                    <maxEntriesPerClub>0</maxEntriesPerClub>
                    <reqSeeds>true</reqSeeds>
                    <lateFee>0</lateFee>
                </events>
            </regatta>
        </regattas>
```

A4. Results for a single boat at the finish line after the placements and margins have been computed. (timingMilestone = 4). All of the layers above need to be specified so that the complete context for the result record is defined (jobId/eventId/RaceId/laneId/milestoneId)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<regattas xmlns="http://www.regattacentral.com/model">
    <version>3.0.1</version>
    <source>www.regattacentral.com</source>
    <regatta>
        <jobId>3939</jobId>
        <flush>false</flush>
        <events>
            <eventId>43</eventId>
            <Races>
                <raceId>65</raceId>
                <lanes>
                    <lane>2</lane>
                    <results>
                        <splitLocation>
                            <timingMilestoneId>4</timingMilestoneId>
                        <label>Finish Line</label>
                        <distance>1500.0</distance>
                    </splitLocation>
                    <elapsedTime>352470</elapsedTime>
                    <place>5</place>
                    <marginToFirst>60020</marginToFirst>
                    <marginToPrevious>0</marginToPrevious>
                    <adjustedTime>352470</adjustedTime>
                    <adjustedMarginToFirst>60020</adjustedMarginToFirst>
                    <adjustedMarginToPrevious>0</adjustedMarginToPrevious>
                    <margin>00:17.06</margin>
                    <penaltyTime>0</penaltyTime>
                </results>
            </lanes>
        </Races>
    </events>
    <venue>
        <venueId>0</venueId>
        <name>Schuylkill River</name>
    </venue>
</regatta>
</regattas>
```