

English Chess Grading Data Web Services API

Version: 0.3

Last updated: 21 Jun 2009

Abstract

This document details a functioning programming language independent API, providing direct program access to the English Chess Federation's grading data over the Internet, with examples in multiple programming languages.

Table of Contents

1	Introduction.....	3
2	Document Conventions.....	3
3	References.....	3
4	Legal.....	4
4.1	This Document.....	4
4.2	Data Copyright.....	4
4.3	Trademarks.....	4
4.4	Disclaimer.....	4
5	Getting Started.....	5
6	Filters.....	6
7	Results.....	7
7.1	Grade Record.....	7
7.2	Club Record.....	8
8	Input Validation.....	8
8.1	Names.....	8
8.2	Grading References.....	8
8.3	Dates.....	8
8.4	Parameters and Values.....	8
9	Grade Data Interface Functions.....	9
9.1	getByName(name, date, filter).....	9
9.2	getByFamilyName(surname, date, filter).....	9
9.3	getByClub(clubname, date, filter).....	10
9.4	getByIDs(refs, date).....	10
9.5	getHistoryByID(iref).....	11
9.6	getDates().....	11
9.7	getClubs(clubname, filter).....	11
10	Exception Handling.....	12
11	Java Client Support.....	12
11.1	Third-Party Products.....	12
11.2	Examples.....	13
12	AJAX Client.....	14

1 Introduction

A Web Services API has been developed that enables programs to directly retrieve ECF grade data over the Internet. Use of this API by Web developers and other Chess software developers reduces maintenance costs, because they no longer need to manually download the whole database in CSV format and load it into their own systems in order to access the latest grading information. Direct access facilitates more frequent updates, since software using this API will pick up changes instantly. Use of the interface makes it easier to use ECF grading references, which should somewhat reduce the occurrence of false name matches.

The interface is currently in Beta status and runs on the testing site <http://testgrading.bcfservices.org.uk>. The interface will eventually run on <http://grading.bcfservices.org.uk>.

This document is intended as an introduction for software programmers (a basic knowledge of software development is assumed) though it contains information of the data, which may be of some use to others.

Based on the Simple Object Access Protocol (SOAP), the interface is programming-language independent. The rest of this document shows how to access grading information via the popular Web programming language PHP. The examples starting from section 5 below are based on the built-in SOAP extension in PHP 5. Section 11 shows how to use the interface with the Java programming language and section 12 shows how to access the data using JavaScript.

2 Document Conventions

Code fragments are written using a `small monospace font`.

Hyperlinks are in the usual [underscored dark-blue](#).

3 References

1. Client code examples: <http://testgrading.bcfservices.org.uk/Clients03.zip>

4 Legal

4.1 This Document

This document is distributed under the terms set forth in the [Creative Commons Attribution-ShareAlike license v3](#). This means that you are free to copy, distribute and adapt the work as long as it is made available under a compatible license. The author does not require attribution. Example code fragments in this document may be used without restriction.

4.2 Data Copyright

The data retrieved via this interface is © Copyright English Chess Federation. Use and re-distribution are subject to terms and conditions available from the ECF Web site.

4.3 Trademarks

Java™ and JavaScript™ are registered trademarks of Sun Microsystems, Inc, in the United States and other countries. The author is independent of Sun Microsystems, Inc. and has no relationship, formal or informal.

Eclipse™ are trademarks of Eclipse Foundation, Inc. The author is independent of Eclipse Foundation, Inc. and has no relationship, formal or informal.

4.4 Disclaimer

This document is provided free on an “as is” basis without warranty of any kind, express or implied. The author does not assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, software, or design disclosed in this document or data obtained via the interface itself.

5 Getting Started

The most common operation is to get the grading information for a given name. The sample code fragment below shows how this can be achieved:

```
<?php
# Display a full record
#
function display_results_text($result)
{ ... }

# Initialisations
#
$opts = Array();
$opts['compression'] = SOAP_COMPRESSION_ACCEPT | SOAP_COMPRESSION_GZIP;
$client = new SoapClient('http://grading.bcfsservices.org.uk/EnglishChess_GradeData.wsdl', $opts);

# The Query
try {
    print "Search by Name with filter:\n";

    # Return only results with "lon" in the club name (member of london clubs)
    #
    $parm = new stdClass(); $parm->parameter = "club"; $parm->value = "lon";
    $parameters[] = $parm;

    # Call the service (look for all records with "John" in their name)
    #
    $res = $client->getByName("John", "2009-01-20", $parameters);
    display_results_text($res);

} catch (SoapFault $exception) {
    print $exception;
}
?>
```

For illustrative purposes, the code that displays the results will be shown later (see section 7.1). Initialisation involves setting the compression options and connecting to the service with the SoapClient call.

The getByName() function takes as input, a part of a name, a date for the grade data (usually today's date) and a set of filter criteria. The example, allows only results that have “lon” somewhere in one of the club names. In summary the code returns all the data valid at the time of 20th January 2009 for all “John”s playing for clubs with “lon” somewhere in their name.

6 Filters

A filter is a collection of filter parameters and each parameter is a name-value pair of strings. Multiple filter parameters can be applied by simply adding to the parameters array. The full set of filter parameters is as follows:

- club - a string that matches any part of a club's name. The clubs listed are those the player has played for in the games on which his grade has been calculated. This is currently limited to 6 clubs.
- clubcode - a string that matches any part of a club's 4-character identifier.
- county – this is a 4-character county code, which is an attribute of a club. The code must be 4 characters in length and may be specified only once. The purpose of this filter is to select players who have played for clubs in a particular county.
- sex - "M" or "F". Many players have no gender specified, they will be eliminated if this filter is used.
- age - a number that matches the age of the player. Many players have no age specified, they will be eliminated if this filter is used.
- age_min - a number specifying the minimum age of a player (inclusive). Many players have no age specified, they will be eliminated if this filter is used.
- age_max- a number specifying the maximum age of a player (inclusive). Many players have no age specified, they will be eliminated if this filter is used.
- standardgrade - a number matching the players' grade.
- standardgrade_min - a number specifying the minimum grade (inclusive).
- standardgrade_max - a number specifying the maximum grade (inclusive).
- rapidgrade - a number matching the players' grade.
- rapidgrade_min - a number specifying the minimum grade (inclusive).
- rapidgrade_max - a number specifying the maximum grade (inclusive).
- current – either “0” or “1”. A “1” selects only those grade records that are considered valid for the season (the rest are historical).
- results_max - the maximum number of results to return. This cannot be used to increase the number beyond the interface limit of 200.

The parameter names and values are not case sensitive. Nearly all parameter may be specified twice and in this case the filter is an intersection of the two filters (logical “and”). This makes sense only for name filters.

7 Results

7.1 Grade Record

The default maximum number of rows returned is 100 (see the `results_max` parameter in section 6), results are truncated if they exceed this. The result is an array of records containing name-value pairs. The records contain the following fields:

- `ECFref` - The ECF grading reference.
- `FIDeref` - The FIDE player's reference.
- `Nation` - FIDE Nation Code.
- `GradeDate` - The date for which this grading is correct.
- `FirstNames` - Space separate list of forenames.
- `FamilyName` - Surname.
- `Sex` - "M" or "F", often empty.
- `Age` - an integer, often empty.
- `StandardGrade`
- `StandardCateg` - The standard grade category classification.
- `StandardGames` - The number of standard games played in the season. This is only available for the latest season, it is empty for previous seasons¹.
- `RapidGrade`
- `RapidCateg` - The rapidplay grade category classification.
- `RapidGames` - The number of rapidplay games played in the season. This is only available for the latest season, it is empty for previous seasons¹.
- `Clubs` - A comma separated list of club names that the player has played for previously, on which his grade has been calculated. This is currently limited to 6 clubs. Extraneous white-spaces are stripped from the result.
- `ClubCodes` - A comma separated list of ECF references for clubs the player played for previously.
- `Counties` - a comma separated list of county codes. This too is derived from the clubs, of which it is an attribute.

The following shows how they can be accessed from PHP:

```
function display_results_text($result)
{
    print "<pre>";
    foreach($result as $res) {
        print "\tName - " . $res->FirstNames . " " . $res->FamilyName . "\n";
        print "\tREF - " . $res->ECFref . "\n";
        if (($f = $res->FIDeref))
            print "\tFIDE - $f (" . $res->Nation . ") \n";
        print "\tGrade Date - " . $res->GradeDate . "\n";
        if (($s = $res->Sex))
            print "\tSex - $s";
        if (($a = $res->Age)) {
            if ($s)
                print ", ";
            print "\tAge - $a";
        }
        if ($s || $a)
            print "\n";
        if (($g = $res->StandardGrade))
            print "\tGrade - $g (cat " . $res->StandardCateg . ", " . $res->StandardGames . " games)\n";
        if (($r = $res->RapidGrade))
            print "\tRapid - $r (cat " . $res->RapidCateg . ", " . $res->RapidGames . " games)\n";
        print "\tClubs - " . $res->Clubs . "\n\n";
    }
    print "</pre>";
}
```

The 'foreach' iterates through each record and each record is contained within a PHP array of objects. Almost all of the get functions of this interface use the same result format.

¹ The history data provided by the ECF does not hold game counts -hopefully this will be rectified in the future.

7.2 Club Record

The default maximum number of rows returned is 100 (see the results_max parameter in section 6), results are truncated if they exceed this. The result is an array of records containing name-value pairs. The records contain the following fields:

- ClubCode - The ECF club reference.
- ClubName - The Club's full name.
- CountyCode – The county that the club resides in.
- UnionID – ? - more description required.

The following shows how they can be accessed from PHP:

```
function display_clubs_text($result)
{
    print "<pre>";
    foreach($result as $res) {
        print "\tClub      - " . $res->ClubName . "\n";
        print "\tCode       - " . $res->ClubCode . "\n";
        print "\tCounty     - " . $res->CountyCode . "\n";
        print "\tUnion      - " . $res->UnionID . "\n\n";
    }
    print "</pre>";
}
```

8 Input Validation

8.1 Names

The length of a name must be at least 2 characters in length and cannot exceed 50 characters in length. Names cannot contain the following meta-characters: '.\+*?[^](\$)'.

8.2 Grading References

The length of an ID must be at least 5 characters in length and cannot exceed 8 characters in length. IDs cannot contain the following meta-characters: '.\+*?[^](\$)'.

8.3 Dates

If no date is supplied (NULL or empty string), the current date is used. If a date is supplied it must be in the ISO 8601 format of 'YYYY-MM-DD'.

8.4 Parameters and Values

Unrecognised parameters will throw an exception.

The length of a parameter value must be at least 1 character/digit in length and cannot exceed 50 characters/digits in length.

Names cannot contain the following meta-characters: '.\+*?[^](\$)'.

Age related parameters must be numeric and only values between 4 and 110 (inclusive) are considered valid.

Grade related parameters must be numeric and only values between 0 and 300 (inclusive) are considered valid.

9 Grade Data Interface Functions

This section describes all the functions supported this API.

9.1 getByName(name, date, filter)

Arguments

- name - part of the name to be found. The longer the string, the more exact the match.
- date - the date for which the results are to be valid for. Use today's date (or NULL) for the latest results.
- filter - an array of objects containing a parameter/value-pair of strings.

Result

This function returns an array of grade records.

9.2 getByFamilyName(surname, date, filter)

Arguments

- surname - the exact Family Name to be found.
- date - the date for which the results are to be valid for. Use today's date (or NULL) for the latest results.
- filter - an array of objects containing a parameter/value-pair of strings.

Result

This function returns an array of grade records.

Example

```
try {
    print "Search by family-name with club-name filter:\n";
    $parm = new stdClass(); $parm->parameter = "club"; $parm->value = "lon";
    $parameters[] = $parm;
    $res = $client->getByFamilyName("Jones", "2009-01-20", $parameters);
    display_results_text($res);
} catch (SoapFault $exception) {
    print $exception;
}
```

9.3 getByClub(clubname, date, filter)

Arguments

- clubname - part of the club name to be found. The longer the string, the more exact the match. The clubs are those the player has played for in the games on which his grade has been calculated. This is currently limited to 6 clubs.
- date - the date for which the results are to be valid for. Use today's date (or NULL) for the latest results.
- filter - an array of objects containing a parameter/value-pair of strings.

Result

This function returns an array of grade records.

Example

```
try {
    print "Search by Club with standard grade filter and limit number of results:\n";
    $parm = new stdClass(); $parm->parameter = "standardgrade_min"; $parm->value = "120";
    $parameters[] = $parm;
    $parm = new stdClass(); $parm->parameter = "results_max"; $parm->value = "5";
    $parameters[] = $parm;
    $res = $client->getByClub("west ", "2009-01-19", $parameters);
    display_results_text($res);
} catch (SoapFault $exception) {
    print $exception;
}
```

9.4 getByIDs(refs, date)

Arguments

- refs - an array of strings containing ECF grading references. The number of IDs cannot exceed the interface maximum of 200.
- date - the date for which the results are to be valid for. Use today's date (or NULL) for the latest results.

Result

This function returns an array of grade records.

Example

```
try {
    print "Search by IDs:\n";
    $IDs[] = "108726F";
    $IDs[] = "105817E";
    $res = $client->getByIDs($IDs, "2009-01-21");
    display_results_text($res);
} catch (SoapFault $exception) {
    print $exception;
}
```

9.5 getHistoryByID(iref)

Arguments

- iref - one individual ECF grading reference.

Result

This function returns an array of grade records covering the history for one person.

9.6 getDates()

Arguments

None

Result

This function returns an array of dates for which grade data exists in ascending order.

Example

```
try {
    print "Dates:<pre>";
    $result = $client->getDates();
    foreach ($result as $res) {
        print $res . "\n";
    }
    print "</pre>";
} catch (SoapFault $exception) {
    print $exception;
}
```

9.7 getClubs(clubname, filter)

Arguments

- clubname - part of the club name to be found. The longer the string, the more exact the match.
- filter - an array of objects containing a parameter/value-pair of strings. The only filters valid for this function are “ClubCode” and “County”.

Result

This function returns an array of club records.

Example

```
try {
    print "Search Clubs:\n";
    $spec = array('parameter' => "county", 'value' => "CMID");
    $parameters = NULL;
    $parameters[] = $spec;
    $res = $client->getClubs("West", $parameters);
    display_clubs_text($res);
} catch (SoapFault $exception) {
    report_fault($exception);
}
```

10 Exception Handling

The SOAP protocol defines a standard set of return codes and exceptions thrown from the server end always have the code “Server”. To help handle exceptions appropriately, this interface embeds codes into the beginning of the exception text. The code is always the first three letters of the text followed by a colon.

The first letter can be a “V” indicating an input validation error or a “R”, indicating a problem with the results. The codes are listed below:

```
VIS: Value is too short/small
VIL: ID is too long
VIM: ID contains meta-characters
VIG: Value is greater than the maximum
VIN: Value is non-numeric
VDI: Dates not in ISO 8601 format of YYYY-MM-DD
VNS: Name is too short
VNL: Name is too long
VNM: Name contains meta-characters
VPZ: Parameter value is empty
VPL: Parameter value is too long
VPM: Parameter value contains meta-characters
VFU: Filter can only be specified once
VFC: Invalid code
VFI: Invalid filter parameter
RIZ: Unknown ID
RDZ: No results
RNZ: Unknown name
RCZ: Unknown club
```

11 Java Client Support

Java interfaces can be produced using Apache AXIS 1.4². To make development easier, code for the client stubs is provided with the code examples.

11.1 Third-Party Products

The example code in this section was developed using Java 6 from Sun Microsystems, so this version (or newer) is the recommended version.

A number of libraries are required to run the client:

Apache AXIS 1.4 JARs :- axis-ant.jar, axis.jar, commons-discovery-0.2.jar, commons-logging-1.0.4.jar, jaxrpc.jar, log4j-1.2.8.jar, saaj.jar, wsdl4j-1.5.1.jar

Other JARs :- activation.jar, mail.jar (from jaf-1.1.1 and javamail-1.4.2 available online)

These 10 JARs must be added to the Java classpath (In Eclipse: Project→Properties→Java Build Path→Libraries, Add External JARs).

²To-date, Apache AXIS 2 does not have adequate support for RPC/Encoded messages, in particular for array types -see Appendix A: Technicalities.

11.2 Examples

```
private static void addFilter(FilterSpec[] filter, int i, String f, String v)
{
    FilterSpec spec = new FilterSpec(f, v);
    filter[i] = spec;
}

static public void main(String args[])
{
    Date now = new Date();
    GradeData_portType ptype = null;
    EnglishChess_GradeData_ServiceLocator locator;

    // Initialise the connection
    try {
        locator = new EnglishChess_GradeData_ServiceLocator();
        ptype = locator.getGradeData_Port();
    } catch (Exception e) {
        reportFault(e);
        System.exit(5);
    }

    // Client call logic

    // Get by name, using filter
    try {
        FilterSpec[] filter = new FilterSpec[1];
        addFilter(filter, 0, "club", "lon");
        GradeData[] data = ptype.getBy_name("John", now, filter);
        displayResultsText(data);
    } catch (Exception e) {
        reportFault(e);
    }

    // Get by club
    try {
        FilterSpec[] filter = new FilterSpec[1];
        addFilter(filter, 0, "county", "CMID");
        GradeData[] data = ptype.getByClub("West", now, filter);
        displayResultsText(data);
    } catch (Exception e) {
        reportFault(e);
    }

    // Get clubs
    try {
        FilterSpec[] filter = new FilterSpec[1];
        addFilter(filter, 0, "county", "CMID");
        ClubData[] data = ptype.getClubs("West", filter);
        displayClubsText(data);
    } catch (Exception e) {
        reportFault(e);
    }

    // Get by IDs
    try {
        String IDs[] = new String[2];
        IDs[0] = "108726F";
        IDs[1] = "105817E";
        GradeData[] data = ptype.getByIDs(IDs, now);
        displayResultsText(data);
    } catch (Exception e) {
        reportFault(e);
    }

    // Get dates
    try {
        String n = new String("");
        Date[] dates = ptype.getDates(n);
        for (int i=0; i < dates.length; i++) {
            System.out.println(dates[i].toString());
        }
    } catch (Exception e) {
        reportFault(e);
    }
    System.exit(0);
}
```

In addition of course, the reportFault() and display results methods will need to be defined -see the code examples.

12 AJAX Client

This method of accessing grade data is of limited use at the moment because of browser limitations. In an attempt to crack down on browser security vulnerabilities, some browsers prevent cross-site scripting altogether -see browser compatibility.

AJAX stubs are not generated automatically by Apache AXIS 1.4, so a hand-crafted JavaScript library is supplied with the code examples in a file called “gd_fns.js”. The client HTML then uses this library to fetch grade data interactively.

The client works by registering a library-supplied function as an event handler for events such as “onchange” and “onclick”. Example HTML fragment:

```
...
<head>
...
  <script src="gd_fns.js" type="text/javascript"></script>
</head>
...
<script type="text/javascript"><!--
...

  function gdr_gdata(response_obj, target_ids, suffix)
  {
    ...
  }

  function gdr_gerr(response_obj, target_ids, suffix)
  {
    ...
  }

--></script>
...

Name: <input type="text" id="gdr_name" onchange="req_name(target_ids, suffixes)"/>
...
```

The event handler fetches inputs from HTML elements tagged with specific Input IDs, then calls either a data-handling function or error-handling function. In the case of multiple input fields, a suffix may be supplied, which is appended to Input ID names in order to distinguish them. The suffixes argument can be an array; the first element is the suffix for the Input ID and the second element is the suffix for the Filter IDs (see below). The target_ids argument is passed “as is” to the handling functions but only the first element of suffixes is passed through.

The call-back functions must be supplied by the user. The following table details the handler functions supplied:

Function Name	Input IDs	Filter IDs	Data Handler	PHP Equivalent
req_name	gdr_name, gdr_date	Y	gdr_gdata, gdr_gerr	getByName
req_fam_name	gdr_fam_name, gdr_date	Y	gdr_gdata, gdr_gerr	getByFamilyName
req_ids	gdr_ids, gdr_date	N	gdr_gdata, gdr_gerr	getByIDs
req_history	gdr_id	N	gdr_gdata, gdr_gerr	getHistoryByID
req_club	gdr_club_name, gdr_date	Y	gdr_gdata, gdr_gerr	getByClub
req_clubs	gdr_clubs_name	Y	gdr_cdata, gdr_cerr	getClubs
req_dates	gdr_opts	N	gdr_gdates, gdr_derr	getDates

Filter parameter and values are identified using gdr_fparm_<n> and gdr_fval_<n>, where n is 1-5. If a filter suffix is specified, they are identified by gdr_fparm_<suffix>_<n> and gdr_fval_<suffix>_<n> respectively.

The call-back function, when called, populate HTML elements, which are tagged with IDs in the usual way. This part specific to each client application but a simple pre-formatted text handler is shown in the code examples.

Appendix A: Technicalities

SOAP

The interface uses the RPC/Encoded method of transfer. Currently, this has the widest support in the software industry, although there is a general trend toward the Document/Literal format.

PHP is one of the most popular Web development languages, so from a practical standpoint, the lack of support for Document/Literal in PHP 5 is a barrier to its wider adoption.

Cross-Site XMLHttpRequest

AJAX browser compatibility:

Browser	Response
Firefox 3.x	Fails quietly
Firefox 3.5	Works
Opera 9.x	Fails quietly
Konqueror 4.1.x	Fails quietly
IE 6	Prompts once for access
IE 7	Unknown
IE 8	Will work