Xfcc – XML Web Services for Correspondence Chess

XfccBasic Specification v1.0 Fourth Draft

by Martin Bennedik, bennedik@gmx.net, www.bennedik.com, www.xfcc.org

Abstract

This specification defines the first Xfcc web service, called XfccBasic. XfccBasic is published by the server, which is a server for playing correspondence chess. XfccBasic is consumed by the client, which can be any kind of chess software, which wants to interoperate with a server. XfccBasic defines two methods, GetMyGames which allows the client to query the server for the current games of the user, and MakeAMove, which allows the user to make a move using the client.
The focus of XfccBasic is on simplicity, in order to make the implementation as easy as possible. Advanced features are not supported in XfccBasic, but will be considered for other definitions in the Xfcc framework.
XfccBasic is based on Web services [2] and PGN standards [5, 6].
Thanks to ICCF, Chessbase, and Austin Lockwood for their help.

GetMyGames

Parameters:

- **username** a string containing the player's username
- **password** a string containing the player's password

Return value:
GetMyGames returns a list of games. Each game consists of several fields describing the game. Some of these fields are based on the tags defined in the PGN standard. There are additional fields, because PGN tags are not intended for the description of a game in progress. Some of the fields are required. Every XfccBasic compliant server of client has to support these fields. Some fields are optional. XfccBasic compliant servers may choose which of the optional fields it wants to send. Clients may ignore optional fields, but should not crash when those fields are present.

Field name	Datatype	Required	PGN	Description
id	int	yes	no	Identifies the game. The same id must be used when committing a move for this game
white	string	yes	yes	Identifies the player with the white pieces.
black	string	yes	yes	Identifies the player with the black pieces.
event	string	yes	yes	Identifies the event or tournament.
site	string	yes	yes	Identifies the site, or in case of a correspondence game, the organisation running the event.

myTurn	bool	yes	no	True, if it is the player's turn in the game.
hasWhite	bool	yes	no	True, if the player has the white pieces in this game.
daysPlayer	int	yes	no	This and the following two values specify the amount of thinking time left to the player until the next time control.
hoursPlayer	int	yes	no	
minutesPlayer	int	yes	no	
daysOpponent	int	yes	no	This and the following two values specify the amount of thinking time left to the opponent until the next time control.
hoursOpponent	int	yes	no	
minutesOpponent	int	yes	no	
moves	string	yes	yes	Contains all the moves of this game in PGN notation (see details below). The moves string may contain additional PGN features, such as comments, which the client may or may not display, as defined in the PGN standard. Clients should not crash when they discover unsupported PGN features, but handle them gracefully.
drawOffered	bool	yes	no	Indicates if the opponent has offered a draw to the player with his last move.
message	string	no	no	Containins any message to the player that the opponent may have send with his last move. This string should contain only messages sent with the last move. Messages sent with previous moves or in between moves can be added to the moves field as PGN comments.
serverInfo	string	no	no	A free form text field with additional information from the server. For example, conditionals are not supported by XfccBasic, but in case the server supports open conditionals, it can supply this information to the user via the serverInfo field. Clients should display this field to the user unparsed.
gameLink	string	no	no	A URL as a link to the game on the server. This link is provided to the user in case he wants to use a feature that is not supported by the client. The client should present the link to the user, and when clicked should open a browser with the URL. The URL provided by the server should open the game. The URL should be complete, e.g. http://www.myserver.com/mygame.html?gameid=1234.
whiteTitle	string	no	yes	Title of white player.
blackTitle	string	no	yes	Title of black player.
whiteElo	int	no	yes	Rating of white player as rated by the organization specified under site.
blackElo	int	no	yes	Rating of black player as rated by the organization specified under site.
whiteNA	string	no	yes	Email address of white player.
blackNA	string	no	yes	Email address of black player.
eventDate	string	no	yes	Starting date of event in the format YYYY.MM.DD

eventSponsor	string	no	yes	Sponsor of the event.
section	string	no	yes	Section of the event.
stage	string	no	yes	Stage of the event.
board	string	no	yes	Board of the event.
timeControl	string	no	yes	Time control as defined in the PGN standard. The PGN standard uses times based on seconds, while typical correspondence time controls are measured in days. Clients should expect very big values here, and display them to the users conveniently, i.e. in days, hours, and minutes, not just in seconds. The PGN standard does not define all possible time controls, therefore this field is optional.
setup	bool	no	yes	True, if the game starts from a non-standard starting position, e.g. Fischer-Random chess. In that case the fen field has to be set to the starting position, as described in the PGN standard. Do **not** use the setup/fen fields in case of a thematic event. Because these are optional values, clients may not support it. Instead, in a thematic, just list the start moves in the moves field.
fen	string	no	yes	Starting position in case setup is set to true. See PGN standard for details.
result	enum	no	no	The server can leave a finished game on the list, so that players do not wonder about suddenly disappearing games. In that case, the result field should be used. Result can be one of the following values: Ongoing, WhiteWins, BlackWins, Draw, WhiteWinAdjudicated, BlackWinAdjudicated, DrawAdjudicated, WhiteDefaulted, BlackDefaulted, BothDefaulted, Cancelled, and AdjudicationPending. If the result field is missing, a result of Ongoing is assumed.

MakeAMove

Parameters:

- **username** a string containing the player's username
- **password** a string containing the player's password
- **gameId** an integer identifying the game to the server
- **resign** a boolean indicating if the player resigns
- **acceptDraw** a boolean indicating if the player accepts the draw offered to him. The server must check the validity of the player's acceptance.
- **movecount** an integer identifying the move number of the move the player is commiting. This is neccessary to avoid problems if multiple messages get sent.
- **myMove** a string containing the move in PGN notation
- **offerDraw** a boolean indicating if the player offers a draw with his move
- **claimDraw** a boolean indicating if the player claims a draw with his move. The server must check if the 3-fold-repetition, 50-moves or insufficient-mating-material rules apply.
- **myMessage** a string containing a message the player wishes to send to his opponent

Make a move can return any of the following values:

- **Success** the move was committed successfully
- **ServerError** the server had an internal error, the message should be resend later
- **AuthenticationFailed** the user supplied an invalid username/password combination
- **InvalidGameID** the game id does not represent a valid game
- **NotYourGame** the game id does not represent a game where the player is involved
- **NotYourTurn** the game id does not represent a game where the player has to move
- **InvalidMoveNumber** the supplied move number does not match the current move number in the game. This value will be returned if the same message gets sent twice.
- **InvalidMove** the supplied move is not valid in the current position
- **NoDrawWasOffered** acceptDraw was true, but no draw was offered
- **LostOnTime** the move was not accepted, because the player lost on time
- **YouAreOnLeave** the move was not accepted, because the player is on leave, and was not allowed to send moves during leave
- **MoveIsAmbigous** the move was not accepted, because the supplied notation was ambiguous (e.g. Re1 instead of Rfe1)

Move representation

Moves in the myMove parameter of MakeAMove and the moves value of the result set of GetMyGames have to conform to the PGN standard [6]. This means:

- Castling is written with capital O, not with zeros.
- In the moves field of GetMyGames, PGN comments can be used to supply additional information to the client. For example, messages sent between the players can be displayed as comments.
- Additional characters, such as + or # may or may not be present in the notation, and should not cause an error in the parsing of the moves field on the client or the myMove field on the server, if present (or absent).
- The PGN enhancement provides a standard to define commands inside the PGN comments [5]. An additional PGN command was defined by Chessbase and Martin Bennedik to support correspondence time stamps:
  `%ccsnt yyyy.mm.dd[,hh:mm[:ss]]`
  ccsnt stands for Correspondence Chess Sent, followed by the time stamp when the move was sent, consisting of the 4 digit year, the 2 digit month, the two digit day, and (optionally) followed by a coma, and the 2 digits hour and minute, and (again, optionally), a 2 digit second.
- The Xfcc client has to be able to handle the complete moves field of GetMyGames without line breaks. The PGN standard requires line breaks, but this differes here, as it is part of an XML document.

For example, a valid value for the moves field of GetMyGames would be:

1.e4 {[%ccsnt 2004.08.31,09:30]} {Welcome to the game.} e5 {[%ccsnt 2004.09.07,16:28]} 2.Nf3 {[%ccsnt 2004.09.07,17:18]} Nc6 {[%ccsnt 2004.09.07,21:03]}

Message and other embedded texts

The message and other embedded texts should be plain text, and confirm to the XML standard, i.e. the following character entities have to be used:

Character	Entity
&	&amp;
‘	'

“	&quot;
<	&lt;
>	&gt;

Note on user authentication

The current proposal for Xfcc uses a username and password as parameters to its methods. The W3C has proposed a standard for user authentication called web services security, however this does not seem to have widespread tool support yet. To make it as easy as possible for the implementors, I therefore propose the simple username and password parameters. We can augment this later (when tool support is widely available) with an additional XfccBasicSecure interface that is identical but uses web services security instead of the parameters. For more information about web services security, see [3].
To increase security, SSL can be used at the moment.

About Martin Bennedik

Martin Bennedik has worked as lead developer and technical director for the software WebSuxess, which sold more than 200.000 times, and as technical architect and developer in projects for the Deutsche Bank, Deutsche Bahn, and the government of the United Kingdom. Recently he launched www.iccf-webchess.com for the Iccf. His website is www.bennedik.com [4], and he can be reached via email at bennedik@gmx.net.

References

[1] http://www.xfcc.org Xfcc website
[2] http://www.w3.org/2002/ws/ Web services spec
http://msdn.microsoft.com/webservices/ Microsoft website about web services
http://www.ibm.com/developerworks/webservices/ IBM website about web services
[3] http://msdn.microsoft.com/webservices/building/security/default.aspx
Microsoft website about web service security
http://www-106.ibm.com/developerworks/webservices/library/ws-secure/
IBM website about web service security
[4] www.bennedik.com Martin Bennedik’s website
[5] http://www.enpassant.dk/chess/palview/enhancedpgn.htm PGN enhancement
[6] http://scid.sourceforge.net/doc/standard.txt PGN standard

Sample WSDL

```
<?xml version="1.0" encoding="utf-8" ?>
- <definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:s0="http://www.bennedik.com/webservices/XfccBasic"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    targetNamespace="http://www.bennedik.com/webservices/XfccBasic"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
- <types>
- <s:schema elementFormDefault="qualified"
    targetNamespace="http://www.bennedik.com/webservices/XfccBasic">
- <s:element name="GetMyGames">
- <s:complexType>
```

```
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="username" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="password" type="s:string" />
    </s:sequence>
    </s:complexType>
    </s:element>
- <s:element name="GetMyGamesResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="GetMyGamesResult"
     type="s0:ArrayOfXfccGame" />
    </s:sequence>
    </s:complexType>
    </s:element>
- <s:complexType name="ArrayOfXfccGame">
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="unbounded" name="XfccGame" nillable="true"
     type="s0:XfccGame" />
    </s:sequence>
    </s:complexType>
- <s:complexType name="XfccGame">
- <s:sequence>
  <s:element minOccurs="1" maxOccurs="1" name="id" type="s:int" />
  <s:element minOccurs="0" maxOccurs="1" name="white" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="black" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="event" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="site" type="s:string" />
  <s:element minOccurs="1" maxOccurs="1" name="myTurn" type="s:boolean" />
  <s:element minOccurs="1" maxOccurs="1" name="hasWhite" type="s:boolean" />
  <s:element minOccurs="0" maxOccurs="1" name="moves" type="s:string" />
  <s:element minOccurs="1" maxOccurs="1" name="drawOffered" type="s:boolean" />
  <s:element minOccurs="0" maxOccurs="1" name="message" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="serverInfo" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="whiteTitle" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="blackTitle" type="s:string" />
  <s:element minOccurs="1" maxOccurs="1" name="whiteElo" type="s:int" />
  <s:element minOccurs="1" maxOccurs="1" name="blackElo" type="s:int" />
  <s:element minOccurs="0" maxOccurs="1" name="whiteNA" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="blackNA" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="eventSponsor" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="section" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="stage" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="board" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="timeControl" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="setup" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="fen" type="s:string" />
  <s:element minOccurs="1" maxOccurs="1" name="result" type="s0:Result" />
  <s:element minOccurs="1" maxOccurs="1" name="daysPlayer" type="s:int" />
  <s:element minOccurs="1" maxOccurs="1" name="hoursPlayer" type="s:int" />
  <s:element minOccurs="1" maxOccurs="1" name="minutesPlayer" type="s:int" />
  <s:element minOccurs="1" maxOccurs="1" name="daysOpponent" type="s:int" />
  <s:element minOccurs="1" maxOccurs="1" name="hoursOpponent" type="s:int" />
  <s:element minOccurs="1" maxOccurs="1" name="minutesOpponent" type="s:int" />
  <s:element minOccurs="0" maxOccurs="1" name="gameLink" type="s:string" />
```

```
<s:element minOccurs="0" maxOccurs="1" name="eventDate" type="s:string" />
  </s:sequence>
  </s:complexType>
- <s:simpleType name="Result">
- <s:restriction base="s:string">
  <s:enumeration value="Ongoing" />
  <s:enumeration value="WhiteWins" />
  <s:enumeration value="BlackWins" />
  <s:enumeration value="Draw" />
  <s:enumeration value="WhiteWinAdjudicated" />
  <s:enumeration value="BlackWinAdjudicated" />
  <s:enumeration value="DrawAdjudicated" />
  <s:enumeration value="WhiteDefaulted" />
  <s:enumeration value="BlackDefaulted" />
  <s:enumeration value="BothDefaulted" />
  <s:enumeration value="Cancelled" />
  <s:enumeration value="AdjudicationPending" />
  </s:restriction>
  </s:simpleType>
- <s:element name="MakeAMove">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="username" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="password" type="s:string" />
  <s:element minOccurs="1" maxOccurs="1" name="gameId" type="s:int" />
  <s:element minOccurs="1" maxOccurs="1" name="resign" type="s:boolean" />
  <s:element minOccurs="1" maxOccurs="1" name="acceptDraw" type="s:boolean" />
  <s:element minOccurs="1" maxOccurs="1" name="movecount" type="s:int" />
  <s:element minOccurs="0" maxOccurs="1" name="myMove" type="s:string" />
  <s:element minOccurs="1" maxOccurs="1" name="offerDraw" type="s:boolean" />
  <s:element minOccurs="1" maxOccurs="1" name="claimDraw" type="s:boolean" />
  <s:element minOccurs="0" maxOccurs="1" name="myMessage" type="s:string" />
  </s:sequence>
  </s:complexType>
  </s:element>
- <s:element name="MakeAMoveResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="1" maxOccurs="1" name="MakeAMoveResult"
    type="s0:MakeAMoveResult" />
  </s:sequence>
  </s:complexType>
  </s:element>
- <s:simpleType name="MakeAMoveResult">
- <s:restriction base="s:string">
  <s:enumeration value="Success" />
  <s:enumeration value="ServerError" />
  <s:enumeration value="AuthenticationFailed" />
  <s:enumeration value="InvalidGameID" />
  <s:enumeration value="NotYourGame" />
  <s:enumeration value="NotYourTurn" />
  <s:enumeration value="InvalidMoveNumber" />
  <s:enumeration value="InvalidMove" />
  <s:enumeration value="NoDrawWasOffered" />
```

```
<s:enumeration value="LostOnTime" />
<s:enumeration value="YouAreOnLeave" />
<s:enumeration value="MoveIsAmbigous" />
  </s:restriction>
  </s:simpleType>
  </s:schema>
  </types>
- <message name="GetMyGamesSoapIn">
<part name="parameters" element="s0:GetMyGames" />
  </message>
- <message name="GetMyGamesSoapOut">
<part name="parameters" element="s0:GetMyGamesResponse" />
  </message>
- <message name="MakeAMoveSoapIn">
<part name="parameters" element="s0:MakeAMove" />
  </message>
- <message name="MakeAMoveSoapOut">
<part name="parameters" element="s0:MakeAMoveResponse" />
  </message>
- <message name="MakeAMove2SoapIn">
<part name="parameters" element="s0:MakeAMove2" />
  </message>
- <message name="MakeAMove2SoapOut">
<part name="parameters" element="s0:MakeAMove2Response" />
  </message>
- <portType name="XfccBasicSoap">
- <operation name="GetMyGames">
<input message="s0:GetMyGamesSoapIn" />
<output message="s0:GetMyGamesSoapOut" />
  </operation>
- <operation name="MakeAMove">
<input message="s0:MakeAMoveSoapIn" />
<output message="s0:MakeAMoveSoapOut" />
  </operation>
  </portType>
- <binding name="XfccBasicSoap" type="s0:XfccBasicSoap">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"
  />
- <operation name="GetMyGames">
<soap:operation
   soapAction="http://www.bennedik.com/webservices/XfccBasic/GetMyGames"
   style="document" />
- <input>
<soap:body use="literal" />
  </input>
- <output>
<soap:body use="literal" />
  </output>
  </operation>
- <operation name="MakeAMove">
<soap:operation
   soapAction="http://www.bennedik.com/webservices/XfccBasic/MakeAMove"
   style="document" />
- <input>
<soap:body use="literal" />
```

```
    </input>
- <output>
  <soap:body use="literal" />
    </output>
    </operation>
    </binding>
- <service name="XfccBasic">
- <port name="XfccBasicSoap" binding="s0:XfccBasicSoap">
  <soap:address location="http://testarea.iccf-webchess.com/XfccBasic.asmx" />
    </port>
    </service>
    </definitions>
```