STANDARD DATA TRANSFER FORMAT

This document describes the first version of a data transfer standard format for data generated through Electronic Performance and Tracking Systems (EPTS). In a joint effort between FIFA and F.C. Barcelona, we have moved towards the creation of a data format or schema that allows providers and clubs to exchange EPTS data in an adaptable, modular and standardized way.

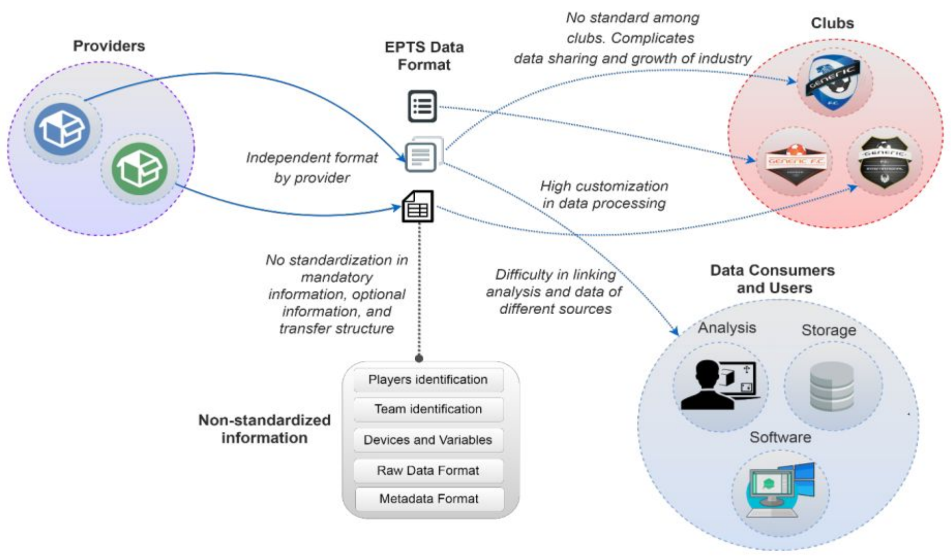
This document explains the conceptual and technical characteristics of the developed standard. First section provides the reasoning and background for this project, highlighting the main issues of the current state of EPTS data transfer mechanism, and the objectives of the proposed standard. The second section covers technical details of the format itself, including an outline of the its structure and detailed description of each parameter.

Why a standard format for EPTS data exchange?

The adoption of EPTS devices in professional and non-professional sections of sports teams has grown considerably during the last 5 years. From wearable devices with multiple types of sensors, to computer vision-driven tracking technology, EPTS devices are providing a great amount of performance data in daily-basis. Alongside the increasing adoption of these devices, the variety of brands and vendors have also increased up to tens of providers in this timeframe. While more data is being generated, the operational maintenance and integration of this information becomes everyday more complex. A central issue is that each different vendor defines its own format and specification for the data that is being provided. While is typical to also provide software to interpret this data, the increasing availability of sources of information and the need of data centralization makes very difficult for clubs to keep the growing pace of this industry, regarding the needs for continuous integration and maintainability of data. Also, the ad-hoc nature of provided formats makes harder to integrate information from different sources, making progress in this area slower.

In the near future is also natural to envision an increased maturity of tracking and performance which would lead to demands of better integrability, faster implementation, and the ability to be transferred in a standardized way. An example of this is the collaboration between clubs, universities and companies to develop new mechanisms to obtain more refined insight from this data. This is also includes the possibility of clubs exchanging player performance information once a player is bought on the transfer market.

The main objective of this project is to help providers and clubs to find better and easier ways to communicate and integrate technology, by reaching a consensus standard for transferring EPTS data. The project is focused exclusively on the technical formatting of data, and not on the specific characteristics of derived performance variables, which are left for the development of each provider on its own.


Providers
EPTS Data Format
No standard among clubs. Complicates data sharing and growth of industry
Clubs
Independent format by provider
High customization in data processing
No standardization in mandatory information, optional information, and transfer structure
Difficulty in linking analysis and data of different sources
Data Consumers and Users
Analysis
Storage
Software
Non-standardized information
Players identification
Team identification
Devices and Variables
Raw Data Format
Metadata Format

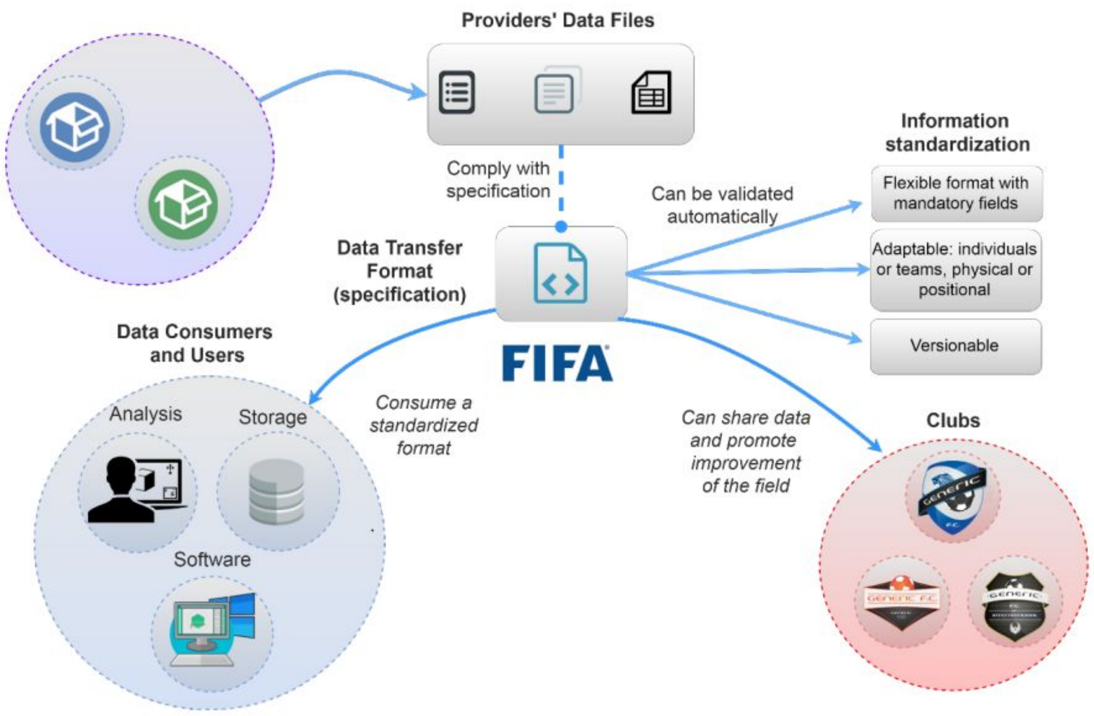
The image above presents the current state of EPTS data transfer exchange. Here, the main issues of lack of standardization are covered, highlighting the following:

- **Dispersion**: Independent format by provider and lack of standardization among clubs.
- **Customized Integration:** Ad-hoc data integration process for multiple sources of data, carrying the load of maintenance when changes are applied.
- **Missing information:** There is no standardization for mandatory information.

Principles of the proposed standard format

Having the issues above in mind, we have developed a standardized format for the exchange of EPTS data. The main design principles that were taken into account are:

- **Standard Structure:** The format needs to define a clear programmatic structure, that allows a deterministic or standardized parsing of any individual data file.
- **Flexibility:** Given the wide variety of field-specific application cases, the increasing number of technology providers and the development of the technology itself, the proposed standard must account for both mandatory and optional parameters, providing adaptability to different scenarios.
- **Extensibility:** The format structure must be able to evolve through extension and modifications, that maintain as much as possibly backward compatibility. It must be versionable.
- **Adaptability:** It can be adapted to multiple kinds of EPTS data, covering GPS, ultra-wide band, and computer-vision driven technologies, among others


Providers' Data Files
Comply with specification
Information standardization
Can be validated automatically
Flexible format with mandatory fields
Adaptable: individuals or teams, physical or positional
Versionable
Data Transfer Format (specification)
FIFA
Data Consumers and Users
Analysis
Storage
Software
Consume a standardized format
Can share data and promote improvement of the field
Clubs

The format is thought for transferring data of derived variables with a relatively high frequency (25Hz, for example), which we will call from now on *Raw Data*. However, the intention is not to transfer the lowest level raw data, such as sensors raw signal output, but pre-built variables of each vendor. An example variable could be "travelled distance" which is provided 25 times per second. However, the format does not explicitly limit the frequency of values either, but allows for flexibility in the frequency, type of data and even general structure.

Based on these guidelines we propose a standardized specification for EPTS data exchange, that is composed by two main components: metadata and data format specification. The metadata defines mandatory and optional elements to include in each specific data file, while the data format specification allows to define a format for parsing a raw data file.

Both are defined within a XML-Schema (XSD) document. This XSD document allows providers to define and describe their own format as an XML document that complies with the mentioned schema. **Is critical to observe that providers do not need to provide the raw data in XML format. The XSD allows to define a structure for a raw data file (data format specification) which can be of any format prefered by the provider. This will be explained in**

detail through this document, but as summary providers will need to hand in the following two documents when delivering data:

- **Format Specification: An XML file including the metadata and data format specification. This files complies with the XSD specification**
- **Raw data file: The actual data file which is in a format that can be parsed line by line, where each line has a free and flexible format that can be defined by each provider.**

Technical Details

In this section we explain in detail the complete structure of the format, making emphasis in both the theoretical conceptualization of each parameter and its specific technical structure. The schema is defined following the guidelines of XSD version 1.1 schema (click this link to find detailed information on XSD specification and available software for visualization and processing).

Along the definition of the schema you will find two main types of elements:

- **Specified Parameter:** This refers to any kind of element that has a predefined name and structure, and makes direct reference to a concept in the EPTS technology landscape.
- **Additional Provider Parameter**: Although it might receive different generic names, its a type of element designed to allow providers to add any additional information that has not been previously defined within the format.

Through these two types of elements its possible to both define and conceptualize recurrent and relevant data in advance, while also providing a mechanism to allow vendors or providers to add additional information they might consider important, or that is missing in the predefined elements.

General Structure

The schema is focused on providing a multi-target (e.g. players, ball) and multi-sensor (e.g. accelerometer, video tracking position) structure, allowing for time-based measurements of each sensor, associated to a given target.

The schema is divided in two main blocks:

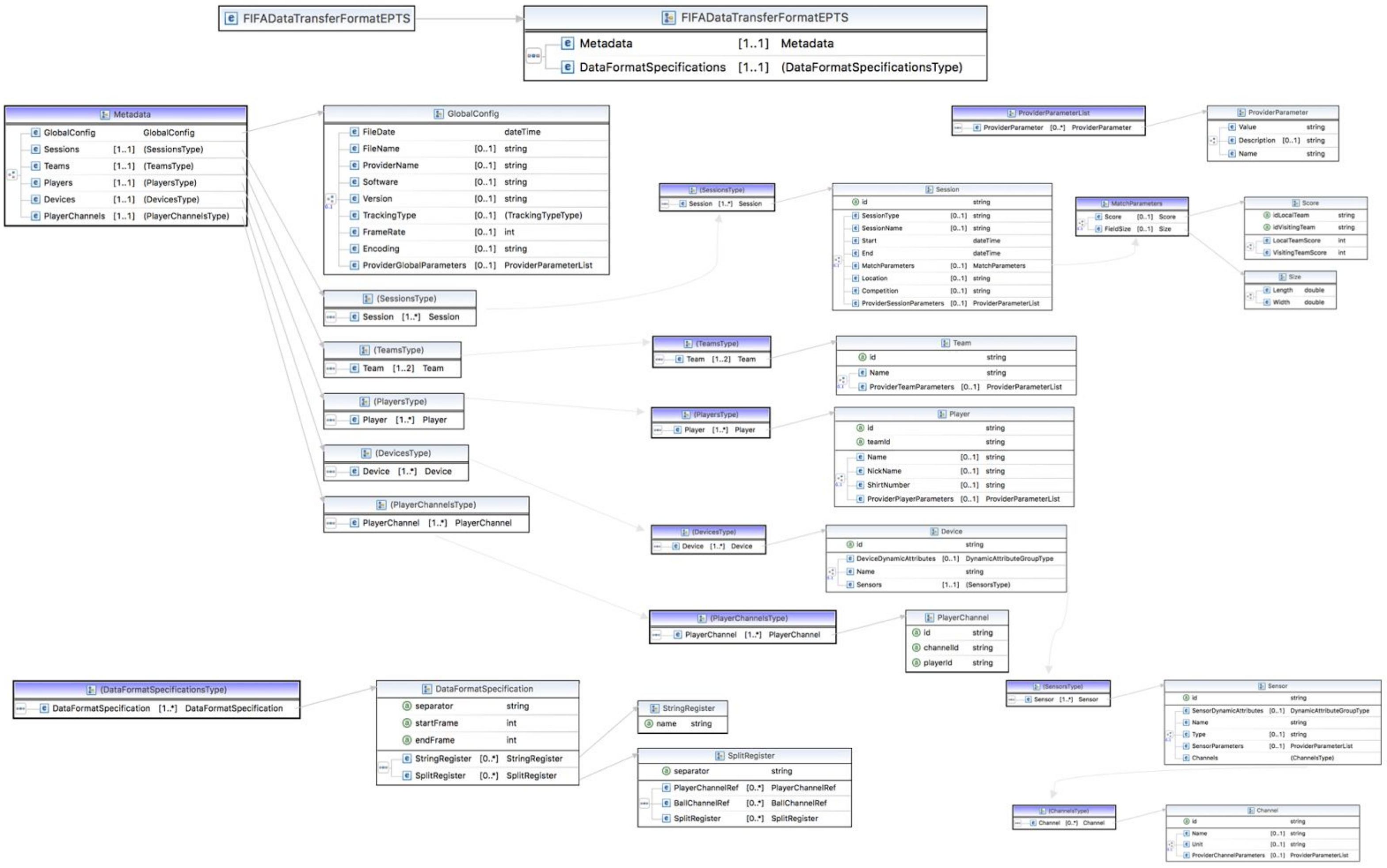
1. **Metadata**: Includes all the information required for contextualizing the information to be provided within the format file, specifying detailed information of players/targets, devices, sensors, location, time, registered phases and complementary information.
2. **DataFormatSpecifications**: This section allows the provider to define the parsing format for a time-based raw data text file with information about the targets (e.g. players, ball) and its corresponding sensor measures (e.g. distance, speed). This meta-format allows anyone to interpret or parse any data file that complies with the standard, while also allowing a wide flexibility to the provider for defining a custom format. Each *DataFormatSpecification* defines the parsing format for a frame which must be replicated

in the frame count range specified in it. These elements are contained in a list that allows the provider to combine frame formats for different periods of the session to handle player substitutions. The meta-format consists of a tree of separators (e.g. semi colon, comma) between string variables (e.g. to represent frame count) and references to player-channel or ball-channel measures.

```
<element name="FIFADataTransferFormatEPTS" type="tns:FIFADataTransferFormatEPTS"></element>

<complexType name="FIFADataTransferFormatEPTS">
    <sequence>
        <element name="Metadata" type="tns:Metadata" minOccurs="1" maxOccurs="1"></element>
        <element name="DataFormatSpecifications" minOccurs="1" maxOccurs="1">
            <complexType>
                <sequence>
                    <element name="DataFormatSpecification" type="tns:DataFormatSpecification" minOccurs="
                </sequence>
            </complexType>
        </element>
    </sequence>
</complexType>
```

In the following image, there is a diagram that shows every element in the schema:


FIFADataTransferFormatEPTS
FIFADataTransferFormatEPTS
Metadata [1..1] Metadata
DataFormatSpecifications [1..1] (DataFormatSpecificationsType)
Metadata
GlobalConfig GlobalConfig
Sessions [1..1] (SessionsType)
Teams [1..1] (TeamsType)
Players [1..1] (PlayersType)
Devices [1..1] (DevicesType)
PlayerChannels [1..1] (PlayerChannelsType)
GlobalConfig
FileDate dateTime
FileName [0..1] string
ProviderName [0..1] string
Software [0..1] string
Version [0..1] string
TrackingType [0..1] (TrackingTypeType)
FrameRate [0..1] int
Encoding [0..1] string
ProviderGlobalParameters [0..1] ProviderParameterList
ProviderParameterList
ProviderParameter [0..*] ProviderParameter
ProviderParameter
Value string
Description [0..1] string
Name string
(SessionsType)
Session [1..*] Session
Session
id string
SessionType [0..1] string
SessionName [0..1] string
Start dateTime
End dateTime
MatchParameters [0..1] MatchParameters
Location [0..1] string
Competition [0..1] string
ProviderSessionParameters [0..1] ProviderParameterList
MatchParameters
Score [0..1] Score
FieldSize [0..1] Size
Score
idLocalTeam string
idVisitingTeam string
LocalTeamScore int
VisitingTeamScore int
Size
Length double
Width double
(SessionsType)
Session [1..*] Session
(TeamsType)
Team [1..2] Team
(TeamsType)
Team [1..2] Team
Team
id string
Name string
ProviderTeamParameters [0..1] ProviderParameterList
(PlayersType)
Player [1..*] Player
(PlayersType)
Player [1..*] Player
Player
id string
teamId string
Name [0..1] string
NickName [0..1] string
ShirtNumber [0..1] string
ProviderPlayerParameters [0..1] ProviderParameterList
(DevicesType)
Device [1..*] Device
(DevicesType)
Device [1..*] Device
Device
id string
DeviceDynamicAttributes [0..1] DynamicAttributeGroupType
Name string
Sensors [1..1] (SensorsType)
(PlayerChannelsType)
PlayerChannel [1..*] PlayerChannel
(PlayerChannelsType)
PlayerChannel [1..*] PlayerChannel
PlayerChannel
id string
channelId string
playerId string
(SensorsType)
Sensor [1..*] Sensor
Sensor
id string
SensorDynamicAttributes [0..1] DynamicAttributeGroupType
Name string
Type [0..1] string
SensorParameters [0..1] ProviderParameterList
Channels (ChannelsType)
(ChannelsType)
Channel [0..*] Channel
Channel
id string
Name [0..1] string
Unit [0..1] string
ProviderChannelParameters [0..1] ProviderParameterList
(DataFormatSpecificationsType)
DataFormatSpecification [1..*] DataFormatSpecification
DataFormatSpecification
separator string
startFrame int
endFrame int
StringRegister [0..*] StringRegister
SplitRegister [0..*] SplitRegister
StringRegister
name string
SplitRegister
separator string
PlayerChannelRef [0..*] PlayerChannelRef
BallChannelRef [0..*] BallChannelRef
SplitRegister [0..*] SplitRegister

Metadata

The *Metadata* element comprises information on general/global configuration information (*GlobalConfig*), characteristics and times of the registered session (*Session*), referential information of associated teams (*Teams*) and (*Players*), the information about the general devices that the players wear (*Devices*) and a list to match each player with a channel that represents the measure taken from him (*PlayerChannels*).

Each element is explained in detail below:

```
<complexType name="Metadata">
        <sequence>
            <element name="GlobalConfig" type="tns:GlobalConfig" />
            <element name="Sessions" minOccurs="1" maxOccurs="1">
                <complexType>
                    <sequence>
                        <element name="Session" type="tns:Session" minOccurs="1"
                            maxOccurs="unbounded" />
                    </sequence>
                </complexType>
            </element>
            <element name="Teams" minOccurs="1" maxOccurs="1">
                <complexType>
                    <sequence>
                        <element name="Team" type="tns:Team" minOccurs="1"
                            maxOccurs="2" />
                    </sequence>
                </complexType>
            </element>
            <element name="Players" minOccurs="1" maxOccurs="1">
                <complexType>
                    <sequence>
                        <element name="Player" type="tns:Player" minOccurs="1"
                            maxOccurs="unbounded" />
                    </sequence>
                </complexType>
            </element>
            <element name="Devices" minOccurs="1" maxOccurs="1">
                <complexType>
                    <sequence>
                        <element name="Device" type="tns:Device" minOccurs="1"
                            maxOccurs="unbounded" />
                    </sequence>
                </complexType>
            </element>
            <element name="PlayerChannels" minOccurs="1" maxOccurs="1">
                <complexType>
                    <sequence>
                        <element name="PlayerChannel" type="tns:PlayerChannel" minOccurs="1"
                            maxOccurs="unbounded" />
                    </sequence>
                </complexType>
            </element>
        </sequence>
</complexType>
```

GlobalConfig

Contains the data of the prameters related to the file and the provider.

```xml
<complexType name="GlobalConfig">
    <all minOccurs="0">
        <element name="FileDate" type="dateTime" />
        <element name="FileName" type="string" minOccurs="0" />
        <element name="ProviderName" type="string" minOccurs="0" />
        <element name="Software" type="string" minOccurs="0" />
        <element name="Version" type="string" minOccurs="0" />
        <element name="TrackingType" minOccurs="0">
            <simpleType>
                <restriction base="string">
                    <enumeration value="Optical" />
                    <enumeration value="GPS" />
                    <enumeration value="RF" />
                </restriction>
            </simpleType>
        </element>
        <element name="FrameRate" type="int" minOccurs="0" />
        <element name="Encoding" type="string" minOccurs="0" />
        <element name="ProviderGlobalParameters" type="tns:ProviderParameterList"
            minOccurs="0" />
    </all>
</complexType>

<complexType name="ProviderParameterList">
    <sequence>
        <element name="ProviderParameter" type="tns:ProviderParameter"
            minOccurs="0" maxOccurs="unbounded" />
    </sequence>
</complexType>

<complexType name="ProviderParameter">
    <all>
        <element name="Value" type="string" />
        <element name="Description" type="string" minOccurs="0" />
        <element name="Name" type="string" />
    </all>
</complexType>
```

Element Name	Description	Type	Opt	Limits
FileDate	timestamp of the file creation date	dateTime		1 - 1
FileName	name of the file	string	x	0 - 1
ProviderName	name of the provider	string	x	0 - 1
Software	name or information about the software	string	x	0 - 1
Version	number of software release	string	x	0 - 1
TrackingType	has to choose between: Optical, GPS and RF	string	x	0 - 1
FrameRate	in case of video analysis, number of frames per second used	int	x	0 - 1

Encoding	character encoding used (e.g. UTF-8)	string	x	0 - 1
ProviderGlobalParameters	provider information, list of optional and not specified parameters	Provider ParameterList	x	0 - 1
ProviderGlobalParameters > ProviderParameter	Not Specified parameter from provider	Provider Parameter		1- no limit
ProviderParameter > Value	value of the parameter	string		1 - 1
ProviderParameter > Description	description of the parameter	string	x	0 - 1
ProviderParameter > name	name of the parameter	string		1 - 1

Sessions

The Session element refers to a specific phase within the event (e.g. training session or match). A training session might be conformed by multiple phases or subsections, that can be represented by individual Session elements, respectively. The same can be done for matches, where Session instances might refer to first and second half respectively.

```
<complexType name="Session">
    <all minOccurs="0">
        <element name="SessionType" type="string" minOccurs="0" />
        <element name="SessionName" type="string" minOccurs="0" />
        <element name="Start" type="dateTime" />
        <element name="End" type="dateTime" />
        <element name="MatchParameters" type="tns:MatchParameters"
            minOccurs="0" />
        <element name="Location" type="string" minOccurs="0" />
        <element name="Competition" type="string" minOccurs="0" />
        <element name="ProviderSessionParameters" type="tns:ProviderParameterList"
            minOccurs="0" />
    </all>
    <attribute name="id" type="string" use="required" />
</complexType>

<complexType name="MatchParameters">
    <all minOccurs="0">
        <element name="Score" type="tns:Score" minOccurs="0" />
        <element name="FieldSize" type="tns:Size" minOccurs="0" />
    </all>
</complexType>

<complexType name="Score">
    <all>
        <element name="LocalTeamScore" type="int" />
        <element name="VisitingTeamScore" type="int" />
    </all>
    <attribute name="idLocalTeam" type="string" use="required"></attribute>
    <attribute name="idVisitingTeam" type="string" use="required"></attribute>
</complexType>

<complexType name="Size">
    <all>
        <element name="Length" type="double" />
        <element name="Width" type="double" />
    </all>
</complexType>
```

Element Name	Description	Type	Opt	Limits
id	unique ID of the session	attribute: string	x	0 - 1
SessionType	string to specify the kind of session	string	x	0 - 1
SessionName	name of the session	string	x	0 - 1
Start	dateTime of the phase start	dateTime		1 - 1
End	dateTime of the phase end	dateTime		1 - 1
MatchParameters	frequent parameters suggested by the schema	MatchParameters	x	0 - 1
MatchParameters > Score	scores of the teams	Score	x	0 - 1
Score > idLocalTeam	id of the local team	attribute: string		1 - 1
Score > idVisitingTeam	id of the visiting team	attribute: string		1 - 1
Score > LocalTeamScore	score of the local team	int		1 - 1
Score > VisitingTeamScore	score of the visiting team	int		1 - 1
MatchParameters > FieldSize	length and width of the field	Size	x	0 - 1
FieldSize > Length	length of the field in meters or pixels	double		1 - 1
FieldSize > Width	width of the field in meters or pixels	double		1 - 1
Location	place of the session	string	x	0 - 1
Competition	name of the competition: training, league, champions, world cup,...	string	x	0 - 1
ProviderSessionParameters	provider information, list of optional and not specified parameters	ProviderParameterList	x	0 - 1
ProviderSessionParameters > ProviderParameter	Not Specified parameter from provider	ProviderParameter		1- no limit
ProviderParameter > Value	value of the parameter	string		1 - 1
ProviderParameter > Description	description of the parameter	string	x	0 - 1
ProviderParameter > name	name of the parameter	string		1 - 1

Teams

Contains the data of the team. Can be one or two.

```
<complexType name="Team">
    <all minOccurs="0">
        <element name="Name" type="string" />
        <element name="ProviderTeamParameters" type="tns:ProviderParameterList"
            minOccurs="0" />
    </all>
    <attribute name="id" type="string" use="required" />
</complexType>
```

Element Name	Description	Type	Opt	Limits
id	unique ID of the team	attribute:string		1 - 1
Name	name of the team	string	x	0 - 1
ProviderTeamParameters	provider information, list of optional and not specified parameters	ProviderParameterList	x	0 - 1
ProviderTeamParameters > ProviderParameter	Not Specified parameter from provider for the team	ProviderParameter		1- N
ProviderParameter > Value	value of the parameter	string		1 - 1
ProviderParameter > Description	description of the parameter	string	x	0 - 1
ProviderParameter > name	name of the parameter	string		1 - 1

Players

List element to reference the information of every tracked player during the registered sessions

```
<complexType name="Player">
    <all minOccurs="0">
        <element name="Name" type="string" minOccurs="0" />
        <element name="NickName" type="string" minOccurs="0" />
        <element name="ShirtNumber" type="string" minOccurs="0" />
        <element name="ProviderPlayerParameters" type="tns:ProviderParameterList"
            minOccurs="0" />
    </all>
    <attribute name="id" type="string" use="required" />
    <attribute name="teamId" type="string" use="required" />
</complexType>
```

Element Name	Description	Type	Opt	Limits
id	unique ID of the player	attribute:string		1 - 1
teamId	unique ID of the team the player belongs to	attribute:string		1 - 1
Name	full name of the player	string	x	0 - 1
NickName	nickname of the player	string	x	0 - 1
ShirtNumber	shirt number of the player	string	x	0 - 1
ProviderPlayerParameters	provider information, list of optional and not specified parameters	ProviderParameterList	x	0 - 1
ProviderPlayerParameters > ProviderParameter	Not Specified parameter from provider for the player	ProviderParameter		1- no limit
ProviderParameter > Value	value of the parameter	string		1 - 1
ProviderParameter > Description	description of the parameter	string	x	0 - 1
ProviderParameter > name	name of the parameter	string		1 - 1

Devices

We provide a three level conceptualization of the information-gathering devices. A Device is understood as a specific physical or virtual element that is able to capture information from different sources.

Each Device is conformed by Sensor elements. For typical EPTS devices, example sensors would be GPS, Accelerometer, Magnetometer and Gyroscope. However, the concept of Sensor is an abstraction of a information-gathering gadget, so it does not have to refer to a specific physical sensor available within the physical device but an abstraction of a source of information. For optical tracking systems, which are completely virtual, a sensor could be the element which identifies a relative position in the field, while another one might be the absolute position in the field.

Sensors contain multiple elements of type Channel, which correspond to the last level of information gathering. Channels allow to distribute the different types of specific sources of information a sensor might have. For the optical tracking example, channels might be XPosition and YPosition. For actual sensors channels might be all or a subset of the channels available on the physical device.

In this section, devices are specified just one time (as general devices) to avoid repetition of the same information but they are associated with each player (This information will be in *PlayerChannels*)

Element Name	Description	Type	Opt	Limits
id	unique ID of the device	attribute:string		1 - 1
DeviceDynamicAttributes			x	0 - 1
Name	name of the device given by the provider	string		1 - 1
Sensors	list of sensors for the device	List		1 - 1
Sensors > Sensor	information about the sensor	Sensor		1 - 1
Sensor > SensorDynamicAttributes			x	0 - 1
Sensor > Name	name of the sensor	string		1 - 1
Sensor > Type	typen of sensor used	string	x	0 - 1
Sensor > SensorParameters	provider information, not specified parameters	ProviderParameterList	x	0 - 1
SensorParameters > ProviderParameter	Not Specified parameter from provider for the player	ProviderParameter		1- no limit
ProviderParameter > Value	value of the parameter	string		1 - 1
ProviderParameter > Description	description of the parameter	string	x	0 - 1
ProviderParameter > name	name of the parameter	string		1 - 1
Sensor > Channels	list of channels for the sensor	List		1 - 1
Channels > Channel	information about the channel	Channel		1 - 1
Channel > id	unique ID of the channel	attribute:string		1 - 1
Channel > Name	name of the channel	string	x	0 - 1
Channel > Unit	units of the measure	string	x	0 - 1
Channel > ProviderChannelParameters	provider information	ProviderParameterList	x	0 - 1
ProviderChannelParameters > ProviderParameter	Not Specified parameter from provider	ProviderParameter		1- no limit
ProviderParameter > Value	value of the parameter	string		1 - 1
ProviderParameter > Description	description of the parameter	string	x	0 - 1
ProviderParameter > name	name of the parameter	string		1 - 1

PlayerChannels

List of channel and player pairs. In this section, each channel from the general devices described in *Devices* will be associated with each player and a identification will be assigned to the pair. This id will identify the measure per channel for the player.

```
<complexType name="PlayerChannel">
    <attribute name="id" type="string" use="required" />
    <attribute name="channelId" type="string" use="required" />
    <attribute name="playerId" type="string" use="required" />
</complexType>
```

Element Name	Description	Type	Op t	Limits
id	unique ID to identify the channel-player pair	attribute:string		1 - 1
channelId	ID to identify the channel	attribute:string		1 - 1
playerId	ID to identify the player	attribute:string		1 - 1

DataFormatSpecification

The *DataFormatSpecification* element presents a flexible structure to define the parsing format for time-based data registers in a separate file. It consists of several *StringRegister* followed by a tree of *SplitRegister*. Both *DataFormatSpecification* and *StringRegister* contain a *separator* param that allows to set the characters for separating variables (e.g. semi colon, comma).

The main idea of this element is allowing the provider to define the parsing format for every row in a .txt document containing the raw time-based data of a set of *Sensor*. Defining a meta-format this way allows anyone to interpret or parse any data file that complies with the standard, while also allowing a wide flexibility to the provider for defining a custom format.

Each element is explained in detail below:

- *DataFormatSpecification*: Element that contains a the format for a frame with its initial *separator*, a frame count range that indicates for which frames the format is applicable
- *StringRegister*: Allows to specify parameters that are in the file but do not represent a measure from a sensor
- *SplitRegister*: It is a recursive element that allows to represent a tree of *separators* between *PlayerChannelRef* or *BallChannelRef*
- *PlayerChannelRef*: Is a reference to the *PlayerChannels* object id that represents the measure of a channel for a player
- *BallChannelRef*: Is a reference to the *Channel* object id and it is used to represent measures from the ball

```
<complexType name="DataFormatSpecification">
     <sequence>
         <element name="StringRegister" type="tns:StringRegister" minOccurs="0"
             maxOccurs="unbounded"></element>
         <element name="SplitRegister" type="tns:SplitRegister" minOccurs="0"
             maxOccurs="unbounded"></element>
     </sequence>
     <attribute name="separator" type="string" use="required"></attribute>
     <attribute name="startFrame" type="int" use="required"></attribute>
     <attribute name="endFrame" type="int" use="required"></attribute>
 </complexType>

 <complexType name="StringRegister">
     <attribute name="name" type="string" use="required"></attribute>
 </complexType>

 <complexType name="PlayerChannelRef">
     <attribute name="playerChannelId" type="string" use="required"></attribute>
 </complexType>

 <complexType name="BallChannelRef">
     <attribute name="channelId" type="string" use="required"></attribute>
 </complexType>

 <complexType name="SplitRegister">
     <sequence>
         <element name="PlayerChannelRef" type="tns:PlayerChannelRef" minOccurs="0"
             maxOccurs="unbounded"></element>
         <element name="BallChannelRef" type="tns:BallChannelRef" minOccurs="0"
             maxOccurs="unbounded"></element>
         <element name="SplitRegister" type="tns:SplitRegister" minOccurs="0"
             maxOccurs="unbounded"></element>
     </sequence>
     <attribute name="separator" type="string" use="required"></attribute>
 </complexType>
```

Element Name	Description	Type	Opt	Limits
separator	Format initial separator	attribute:string		1 - 1
startFrame	First frame included in the format	attribute:int		1 - 1
endFrame	Last frame included in the format	attribute:int		1 - 1
StringRegister	Element to represent a string variable in the format	element	x	0 - N
StringRegister > name	Variable name	attribute:string	x	1 - 1
SplitRegister	Element to represent a separator between elements	element	x	0 - N
StringRegister > PlayerChannelRef	Reference to a player-channel measure	element	x	0 - N
StringRegister > PlayerChannelRef > playerChannelId	PlayerChannel id	attribute:string		1 - 1
StringRegister > BallChannelRef	Reference to a ball-channel measure	element	x	0 - N
StringRegister > BallChannelRef > channelId	Channel id	attribute:string		1 - 1

Keys and Reference keys

The schema includes some rules to validate uniqueness of the identifiers and reference integrity for the foreign ids.

- Unique keys
  - Team id
  - Player id
  - Channel id
  - PlayerChannel id

- Reference integrity
  - The *teamId* in Player must reference the Team id attribute
  - The *idLocalTeam* in *Session > MatchParameters > Score* must reference the Team id attribute
  - The idVisitingTeam in *Session > MatchParameters > Score* must reference the Team id attribute
  - The *channelId* in *PlayerChannel* must reference the Channel id
  - The player in *PlayerChannel* must reference the Player id
  - The *playerChannelId* in *DataFormatSpecification > SplitRegister > PlayerChannelRef* must reference the PlayerChannel id
  - The *channelId* in *DataFormatSpecification > SplitRegister > BallChannelRef* must reference the Channel id

Example

In the folder, you will find an example (in the file called **example.xml**) and the raw txt data for it (in the file called **exampleRawData.txt**). It is a simple example for 3 players and 5 sensors with its corresponding channels:

- Position (x, y, z)

- Distance (total distance)
- Speed (Average and maximum speed)
- Acceleration (Current and maximum acceleration)
- Heartbeat (Current and maximum heartbeat)

The *DataFormatSpecification* defines the format in the *exampleRawData.txt* file. That is, for each frame:

frameCount:x,y,z,distance,avg_speed,max_speed,acceleration,max_acceleration,heartbeat,max_heartbeat;:ballX,ballY,ballZ:

Where you can find:

- The frame count for the tracking data separated by **:**

- The player data, with each player info separated by **;**
- For each player, all channels separated by **,** that is: x,y,z,distance,avg_speed,max_speed,acceleration,max_acceleration,heartbeat,max_heartbeat
- After the players data, the ball info separated by **:** and then each channel for the ball (x, y, z) separated by **,**