Betradar Unified Odds

Integration information for development

March 27th - 2019

Table of contents

Introduction

Betradar Unified Odds ensures you can uniformly access all odds for all events (matches, races, outrights) that Betradar supports in a consistent and fast manner. There are two ways to access Betradar Unified Odds-related information: either through the UnifiedFeed Software Development Kit (SDK) library in Java and C#, or directly at the protocol level.

The preferred and recommended mechanism is the UnifiedFeed SDK, as it helps simplify protocol handling; in particular, things like caching of descriptive localized meanings of markets, outcomes, players, teams, etc. Unless fine-grained control is a requirement, you can focus on the UnifiedFeed SDK and can safely disregard the protocol specification.

Throughout Unified Odds (whether you use the SDK or the protocols) Betradar upholds various unifying concepts: names and terms are the same across our different endpoints (odds, fixtures, teams or sports) and datatypes are consistent.

1. Unified Odds – General information

In this chapter you will find some general information about our Unified Odds product. Hereunder information about the different ways of access, concepts used in unified feed, the different environments we provide, language support, and more.

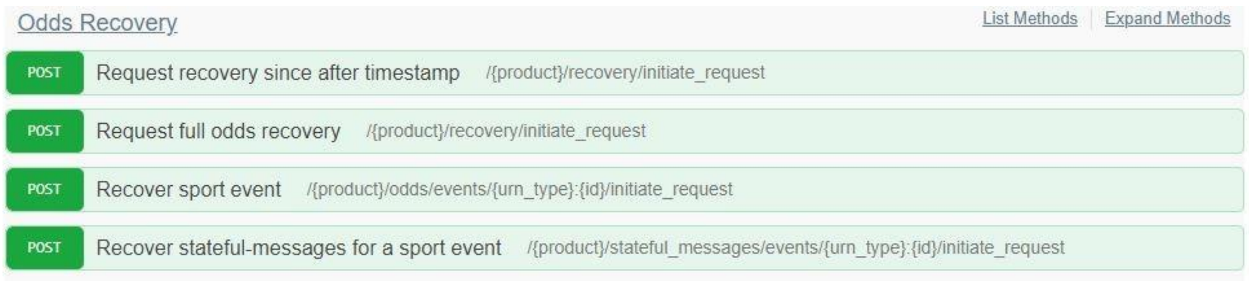
1.1. Access restrictions for odds recovery

For security reasons we only allow connections from IP addresses that have been added to our whitelist. It is therefore necessary that you provide us with the relevant IP addresses from which you are accessing our server before using the production server. Otherwise your login requests will not be successful.

There is a limitation on how many requests you can send to the server.

In general, these limits apply to all odds producers, but only to the Odds Recovery endpoints found in our self-service documentation (usually at the top of the page).

The odds recovery endpoints will typically be found at the top of each API on https://iodocs.betradar.com/


Odds Recovery

List Methods | Expand Methods

POST	Request recovery since after timestamp	/{product}/recovery/initiate_request
POST	Request full odds recovery	/{product}/recovery/initiate_request
POST	Recover sport event	/{product}/odds/events/{urn_type}:{id}/initiate_request
POST	Recover stateful-messages for a sport event	/{product}/stateful_messages/events/{urn_type}:{id}/initiate_request

Table 1 - Recovery categories

Recovery length	Max requests per 10 minutes	Max requests per hour
`< 30 minutes ago`	20	60
`>= 30 minutes & <1 day`	4	10
`>= 1 day (24 hours)`	2 per 30 minutes	4 per 2 hours
`Single events`	100	300

An example would be:

When a customer is rate limited, we only block client from doing new recoveries. Ordinary messages still come through as usual, any already accepted recoveries will still complete. If the rate is X requests over Y minutes, we will at maximum block a client for Y minutes. The table above illustrates 4 different categories of recovery requests. A timestamp is used when performing a recovery, and will add the recovery request to the appropriate category. If you do a quick recovery after a disconnection ~5 minutes ago, the request will count towards the "< 30 minutes ago" category limits.

Please note

If you happen to hit the max requests per hour limit you will not be rate limited for an hour, but rather for 10 minutes if i.e. You made 10 recoveries in 6 minutes (< 30 minutes ago), you will be limited for 10 minutes from the point the last request was made.

An example of how the rate limits work (">= 30 minutes & < 1 day"):

This category has 4 requests per 10 minutes (seen in the above *Table.1*).

Table 2 - Rate limit timeline example

Timeline (in minutes)	Numer of Requests made	Rate limit status
0	1 (1 request made – 10 minutes and 1 hour timers started)	Open (3/4 requests left)
5	1 (2 requests made)	Open (2/4 requests left)
6	1 (3 requests made)	Open (1/4 reqests left)
7	1 (4 requests made)	**Closed** (0/4 requests left)
9	1 (5 requests made)	**Rejected**. Is not counted (0/4 requests available)
10	0	**Reopened** (1/4 requests left)
12	1 (6 requests made)	Closed (0/4) request left
15	0	**Reopened** (1/4 requests left)
16	0	Open (2/4 requests left)
17	1 (7 requests made)	Open (2/4 requests left)
18	1 (8 requests made)	Open (1/4 requests left)

This means that we keep track of the time (the legal) requests were made, and remove them once they are outside of the 10-minute window.

There is also a limit on the number of messages:

- 20 000 messages in a single queue
- 400 000 messages in all queues together

If you exceed these limits, your account will be automatically suspended for ~60 minutes. The account can be reactivated by our support department immediately, once the spamming has been fixed.

If you need information outside of what is provided in this document, or something is unclear, please feel free to contact our support team.

EMAIL: Support@betradar.com
PHONE: +41 71 517 72 00

1.2. UnifiedFeed SDK background

Betradar provides an extension to the Bookmaker SDK for accessing the Unified Odds in Java or C#.

Read more about how to use the SDK using our online documentation available at http://sdk.sportradar.com/, in the "*UnifiedFeed SDK*" section.

If you cannot work with the SDK, the protocols can be used instead. This will, however, add more complexity.

1.3. Protocol/API background

Betradar Unified Odds are provided through two protocol mechanisms: Messaging over *Advanced Message Queuing Protocol* (AMQP), and a *HTTP/XML-based application program interface* (API). The messages are designed to be lightweight and only include important changes. Additional information such as match-details (team names, player names, etc.), and localized versions of such information, are obtained through the API. This design enables Betradar to keep the messages as small as possible for best possible performance and latency.

When programming directly against the protocol, you must handle additional elements and attributes where specified in the corresponding schemas. This enables Betradar to add new functionality when it becomes available, and for you to take advantage of such information when convenient. No existing required data is removed or renamed. If such changes are made, they do not show up in the existing feed, and the existing feed works for at least 24 months after such a change has been made.

- Unified Odds Messages provide fast real-time updates to key information, such as odds over AMQP/XML.
- The Unified Odds API is a request-based RESTful API for additional information (player names, current scores, market descriptions, etc.). For further information, view the API section on https://iodocs.betradar.com/.

1.4. Access to proxy AMQP servers from Asia

For customers in Asia that experience latency issues, it may be beneficial to try Betradar's proxy AMQP servers. These have been shown to reduce latencies for some Asian customers. Instead of connecting to mq.betradar.com, Asian customers can try to connect to either of these two servers instead for improved performance:

- mq.ap-southeast-1.betradar.com (Singapore)
- mq.ap-northeast-1.betradar.com (Tokyo)

For more information about AMQP topic handling, see THIS dedicated section.

1.5. Frequently asked questions

This section will list some commonly asked questions related to the use of Unified Feed.

Q1: **Why am I not getting any odds_change messages, after I connect?**

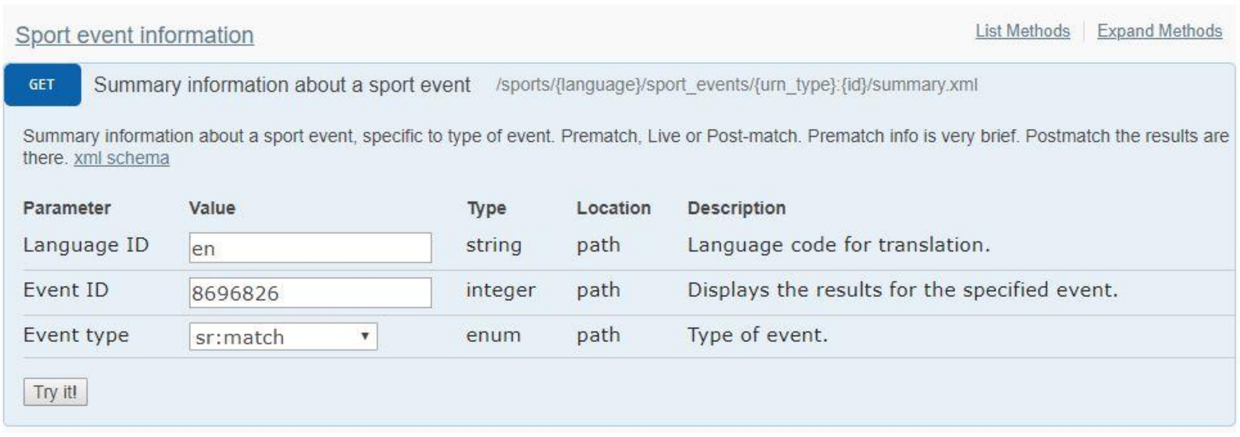
A: If this is the first time you connect or if you have not been disconnected, you first need to initiate a recovery sequence over the API, to start the odds message subscription. The SDK handles this recovery automatically for you if you are using it.

Q2: Why do I receive multiple updates to markets with the same id in the odds_change message?

A: A market is uniquely defined by its *id* AND *specifiers*. Many markets have multiple lines, and the Betradar system provides odds updates for multiple lines in the same odds-update (e.g. the totals markets). More details on markets and specifiers may be found in the concept section.

Q3: **Why am I not receiving the final score/result of a match (match_result)?**

A: The final score (match_result) is not available through the prematch (CTRL) producer, and is only available within the last odds_change message from the Live Odds producer. To receive the result, you will need to have booked the match **and** be subscribed to the Live Odds producer (and the match has to be covered). If these criteria are not met, you will need to pull the match_result though the API. This can be done using the summary.xml endpoint.



Sport event information

List Methods | Expand Methods

GET Summary information about a sport event /sports/{language}/sport_events/{urn_type}:{id}/summary.xml

Summary information about a sport event, specific to type of event. Prematch, Live or Post-match. Prematch info is very brief. Postmatch the results are there. xml schema

Parameter	Value	Type	Location	Description
Language ID	en	string	path	Language code for translation.
Event ID	8696826	integer	path	Displays the results for the specified event.
Event type	sr:match	enum	path	Type of event.

Try it!

XML example of a returned result:

```
<match_summary
<sport_event_status status="4" home_score="2" away_score="2"
status_code="4">
    <period_scores>
        <period_score home_score="2" away_score="2" type="regular_period"
number="1"/>
        <period_score home_score="0" away_score="0" type="regular_period"
number="2"/>
    </period_scores>
</sport_event_status>
<coverage_info level="bronze" live_coverage="false">
    <coverage includes="basic_score"/>
</coverage_info>
</match_summary>
```

Q4: Why do I sometimes receive a bet_cancel message instead of a bet_settlement message with void factor?

A bet_cancel message is sent when we can’t settle a market as voided for the whole time frame. As an example; a producer offers a market in error on an event, but then a live producer correctly offers the market. This means that the first producers can’t send a bet_settlement with the market voided.

1.6. Concepts

This section of the documentation describes different important concepts and abbreviations one might find when reading this document. Some are related to the SDK, others to sports betting, or general terms used throughout Unified Feed. In this chapter you will also find information about the different markets and sport events offered in Unified Feed, coming from multiple producers. You will also find all of our available language support at the end of this chapter.

1.6.1. Sport events

A sport event is a match, race and/or outright. Every message contains information about only one sport event. For example, an odds change message typically includes all changes to any markets for a particular sport event. All sport events have a unique ID that identifies the sport event in the messages. The API then provides endpoints that enable you to look up fixture information for a given sport event ID (https://iodocs.betradar.com). More information about sport events in context can be found HERE.

XML example:

```
<odds_change product="1" event_id="sr:match:14011583"
timestamp="1522772129383">
  <sport_event_status status="0" match_status="0"/>
  <odds>
.... Info about markets....
 </odds>
</odds_change>
```

Match status

The *match_status* of an event gives an indication of which context the current match is in. This is illustrated in the match_status element. For a complete list of all match statuses please consult our match_status.xml endpoint in the API found HERE. More information about match statues can also be found in the messages section.

XML example:

```
<match_status_descriptions response_code="OK">
    <match_status id="0" description="Not started"/>
    <match_status id="41" description="1st extra"/>
    <match_status id="2" description="2nd period" period_number="2"/>
    <match_status id="70" description="Cancelled"/>
</match_status_descriptions>
```

1.6.2. Specifiers

In Live Odds there is a concept called *Special Odds Value*, the equivalent in Unified Odds are called *specifiers*. The Unified Odds specifiers is a cleaned up version of the special odds values that provides a uniform and descriptive way of specifying additional parameters that uniquely identifies a market.

The differences are that legacy Live Odds only sends the values with a single separator (usually a / symbol), while Unified Feed sends a key/value pair separated by the | symbol. The following example is from a cricket match, where the over/under bet of 2.5 (total=2.5) is set to the 5th "over" in the cricket match, represented by the "overnr=5" value.

- Legacy special odds value: "5/2.5"
- Unified feed specifiers: "overnr=5|total=2.5"

1.6.3. Extended specifier

In addition to the specifier, some markets have an extended specifier attribute to add an extra layer of information. This extended specifier is NOT a unique way to identify a market, but simply more extra information added to certain markets:

```
<market status="0" id="8" specifiers="goalnr=2"
extended_specifiers="score=0:1"/>
<market favourite="1" status="-1" id="8" specifiers="goalnr=3"
extended_specifiers="score=1:1">
  <outcome id="6" odds="2.95" probabilities="0.3" active="1"/>
  <outcome id="7" odds="6.0" probabilities="0.14" active="1"/>
  <outcome id="8" odds="1.65" probabilities="0.56" active="1"/>
</market>
```

Please note that not all markets have extended specifiers, and this attribute should not be used to identify a unique market. Only the market ID and the normal market specifier should be used for this purpose.

1.6.4. Markets

Betradar Unified Odds utilizes markets and market lines. Each market is a bet type identified with a unique ID and within a market, multiple different lines are often provided.

Each of these lines is uniquely identified by additional specifiers. An example would be "*Total Goals 2.5*" is the same market as "*Total Goals 1.5*", but these are two different market lines. The market ID for both are the same, but the first one has the specifier *goals=2.5*, and the other one has a specifier *goals=1.5* that uniquely identifies them.

XML example of the same market with 2 different market lines.

```
<market name="Total" id="18" specifiers="total=2.5" status="1">
    <outcome name="over 2.5" active="1" id="12" odds="2.3"/>
    <outcome name="under 2.5" active="1" id="13" odds="1.55"/>
</market>

<market name="Total" id="18" specifiers="total=1.5" status="1">
    <outcome name="under 1.5" active="1" id="13" odds="2.1"/>
    <outcome name="over 1.5" active="1" id="12" odds="1.65"/>
</market>
```

The market lifetime for a sport event typically begins when Betradar provides pre-match odds – often well before the match starts (and the same market continues to live). If Betradar does not cover a sport event live, or the market for some reason is not suitable live, and the market is closed once the match starts.

In general, markets in the Unified Odds model span pre-match and live. They also use the same IDs and specifiers if the sport event hasn't started yet, is live, or has ended.

Life cycle of odds

Life cycle of odds (LCoO) is a term used for markets which are offered pre-match only. These markets are closed before the start of a match, and are settled as soon as the necessary result is reached, e.g. "*How many points will LeBron James score tonight?*".

Market status

The market status is an important concept, and Betradar provides different statuses for a market, the intended market status behaviour is illustrated in the following *Figure.1*:

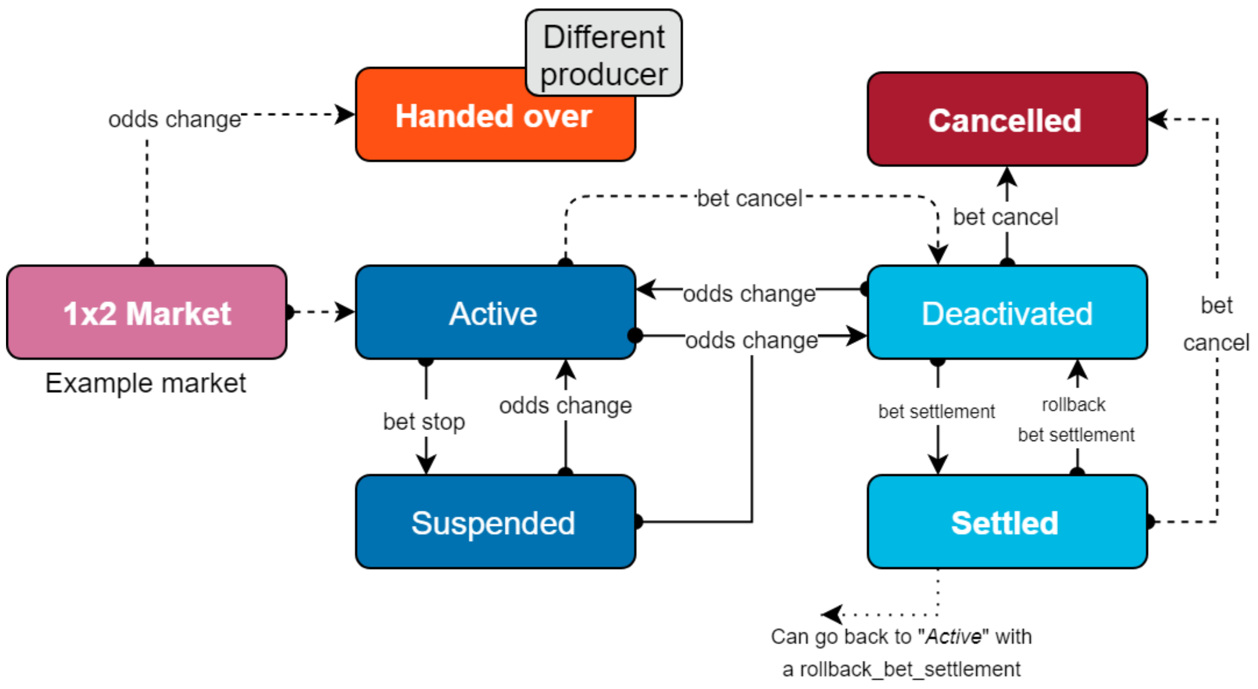


Figure 1 – Expected market behavior

Table 3 - Market behavior description

Status ID	Status text	Description
`1`	`Active`	Odds are provided and you can accept bets on the market.
`-1`	`Suspended`	Odds continue to be provided but you should not accept bets on the market for a short time (e.g. from right before a goal and until the goal has been observed/confirmed).
`0`	`Deactivated/Inactive`	Odds are no longer provided for this market. A market can go back to *Active* again i.e.: A total 3.5 market is deactivated since 0.5, 1.5 or 2.5 is the most balanced market. However, if a goal is scored, then the 3.5 market becomes the most balanced again, changing status to *active*. There are numerous other reasons for this change as well, and it happens on a regular basis.
`-3`	`Settled`	Bet Settlement messages have been sent for this market, no further odds will be provided. *However, it should be noted that*

		in rare cases (error conditions), a settled market may be moved to cancelled by a bet_cancel message.
`-4`	`Cancelled`	This market has been cancelled. No further odds will be provided for this market. This state is only seen during recovery for matches where the system has sent out a cancellation message for that particular market.
`-2`	`Handed over`	**Not a real market status**. This status is normally seen under recovery, and is a signal that the producer that sends this message is no longer sending odds for this market. Odds will come from another producer going forward (and might already have started coming from the new producer). *Handed over* is also sent by the prematch producer when the Live Odds producer takes over a market. If you have not received the live odds change yet, the market should be suspended, otherwise the message can be ignored. If the live odds change does not eventually appear, the market should likely be deactivated. **Note:** If more than 60 seconds pass without an odds change after the handed over market occurs, it should definitely be treated as an error case.

Outright markets

Outrights are ordinary markets in Unified Odds. Hence odds updates are provided with ordinary odds_change messages, and resulting is provided with normal bet_settlement messages.

Outright markets are markets that have the “outcome_type” attribute set to “pre:outcometext”. This helps to identify the outright markets. See THIS chapter for more information about market descriptions.

An outright is seen as one market for an event. The event can be a match, a race or a tournament season. This means the event_id is in one of the following formats: sr:season:1234, sr:stage:1234 or sr:simple_tournament. The types are a bit different, but they can all be looked up using sport_events/(id)/fixture.xml. This will return either a tournament_info (in the case of an sr:season, some sr:stages, and sr:simpletournament) or a fixtures_fixture (in the case of sr:match and some sr:stages).

- **sr:season:1234** refers to a particular season of a tournament/league. This is the event id type you will most often see used for long-term outrights such as “*Who is the winner of Premier League 1993 or 1994*”.

```
<tournament_seasons>
 <seasons>
       <season id="sr:season:55" name="Premier League 93/94"
   start_date="1993-08-14" end_date="1994-05-09" year="93/94"
   tournament_id="sr:tournament:17"/>

       <season id="sr:season:54" name="Premier League 94/95"
   start_date="1994-08-20" end_date="1995-05-14" year="94/95"
   tournament_id="sr:tournament:17"/>

       ... Information about all seasons ...
 </seasons>
</tournament_seasons>
```

- **sr:stage:1234** is used for top-level race events (e.g. F1-season), but also for the individual competitions (F1 GP Monaco), as well as the actual race-events (F1 Qualification Race).

```
<tournament id="sr:stage:306297" scheduled="2016-10-
13T09:00:00+02:00" scheduled_end="2017-10-02T06:00:00+02:00"
name="PGA Tour 2017">
```

- **sr:simple_tournament** does not have season information, and typically has less information like lack of translations. sr:simple_tournament is currently used for e.g. Golf tournaments.

```
<tournament id="sr:simple_tournament:40" name="Eerste Divisie">
    <tournament_length start_date="2003-07-01" end_date="2042-12-15"/>
    <sport id="sr:sport:1" name="Soccer"/>
    <category id="sr:category:35" name="Netherlands" country_code="NLD"/>
</tournament>
```

Some outrights the Betradar system provides in structured format: This means that the market has a fixed name, and that the outcomes are well-defined entities. Other outrights are free-text outrights. These outrights have a free-text name and free-text outcomes, and it is not so easy to determine if the outcome in one outright is the same as an outcome in another outright. It is Betradar’s intent to provide more and more outrights in the structured format in the future.

Outrights with free-text market name and outcome descriptions are handled by a system called variant descriptions.

Outright XML example:

```
<odds_change product="3" event_id="sr:season:41266"
timestamp="1512039661932" request_id="642330526">
  <odds>
    <market favourite="1" status="1" id="534"
specifiers="variant=pre:markettext:45810">
      <market_metadata next_betstop="1512157500000"/>
      <outcome id="pre:outcometext:5982" odds="30.0" active="1"/>
      <outcome id="pre:outcometext:6076" odds="1.1" active="1"/>
      <outcome id="pre:outcometext:6090" active="0"/>
      <outcome id="pre:outcometext:115271" odds="8.0" active="1"/>
      <outcome id="pre:outcometext:7306474" odds="7.0" active="1"/>
      <outcome id="pre:outcometext:7333812" active="0"/>
    </market>
  </odds>
</odds_change>
```

All outrights have similar format, the only exception is the *multiwinner* market. This market would have a specifier like “variant=pre:markettext:45810|winners=3”, where the additional specifier denotes the number of winners (winners=3).

Outright market ids:

Table 4 - Outright market ids

ID	Name	Sport	Example markets
559	Free text market	Formula1, cycling, all race winter sports (but not ice hockey, etc.)	• 10 km Sprint oberhof – Winner • Tour de Suisse 2017 – Stage 2 – Winner • Azerbaijan Grand Prix Race Winning Constructor • Azerbaijan Grand Prix Race – Winning Margin • Azerbaijan Grand Prix Race – Fastest Lap Winner • Tour De France 2017 – Stage 3 – Winner • Tour De France 2017 - Winner
535	Short term free text market	All sports not covered by ID: 559	• Oscar – Best Actress • BMW SA Open 2017 – 1st Round Leader

			• Nobel Peace Prize 2017 - Winner
534	Championship free text market	All sports not covered by ID: 559	• WC 2018 Qualification UEFA Group A – Winner • ATP - Australian Open 2017 – Winner • Championship 2016/17 - Top Midlands Club • Ligue 1 2016/17 - w/o Paris Saint Germain – Winner • Bundesliga 2016/17 - To Finish Bottom
536	Free text *multiwinner* market	All sports	• Premier League 2016/17 – Relegation • Ligue 1 2016/17 - Bottom 3 • Bundesliga 2017/18 - w/o FC Bayern Munich - Top 3 • 7.5 km Sprint Women Oberhof - Top 3 • Rally Sweden 2017 - Top 3 • IAAF World Championships 2017 - Javelin Throw Men - Top 3

Outrights using the API

If you are using the APIs directly, the endpoint *sport_events/(id)/summary.xml* (https://iodocs.betradar.com/unifiedfeed) will work for all types of events (including seasons, stages, tournaments, etc.) and will return different types of information for different events and for outrights. You can use this to find the name of the event as well as additional information where applicable.

Dead heat factor

Dead heat factor is a floating point number used when there is a tie in outrights. Payout should be calculated as:

payout = stake*odds*dead_heat

So in the event of a lot of a big tie in the outright, a punter can actually lose money on a winning bet.

Example:

- 2way tie for first place in an outright with one winner: dead heat would be 0.5
- 3way tie for second place in an outright with three winners: Dead heat for 1st place is null, for the 3 tied players it is 0.67

Outcome version for outright markets

For some markets the outcomes are not fixed, but they can vary over time e.g. First Goal Scorer market. There might be a new player added that wasn't there when the market was added, or an outrights winner market may receive a new competitor as an outcome, that was not present when the outright market first was offered.

By default, in Unified Odds Feed these new outcomes are just added. However, there is a configurable option to turn on "version". When this option is turned on the outcomes for a particular market is fixed, and in examples such as the ones outlined below, a new market will be offered instead. To separate these different markets when this option is turned on there is an additional specifier added to this type of markets called *version*. Each market version will have its own version id.

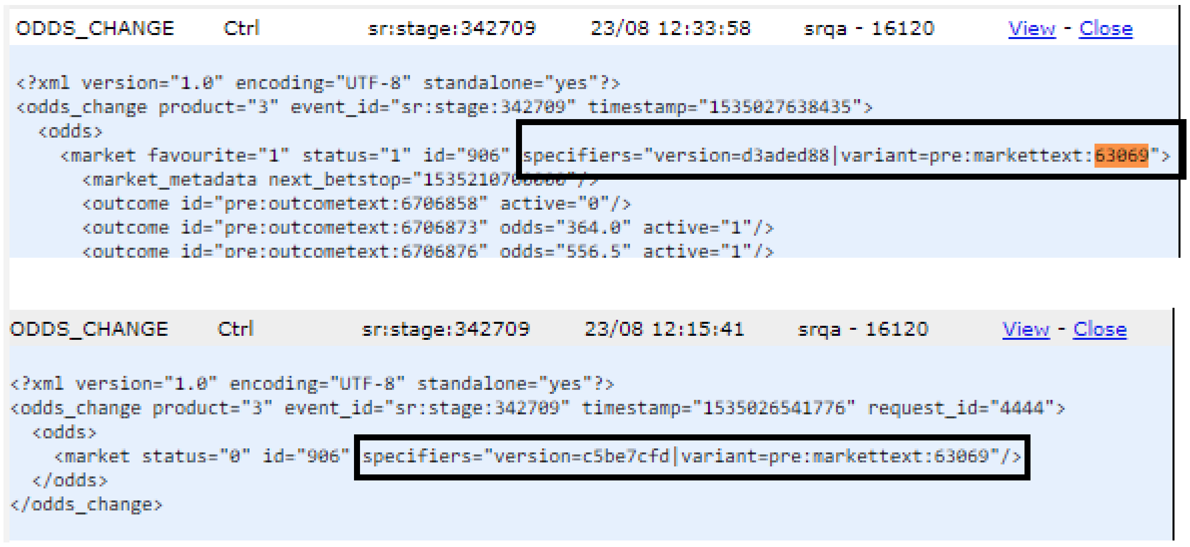
When a new outcome is detected, the odds producer will deactivate the previous version of that market and send out the new market version.

During bet settlement, the odds producer will send out bet settlement for all the different markets that it has offered odds on.

Finally, during recovery, previously sent market versions will also be included, so that you can deactivate these market versions if they are still open on the client side.

On each change of the list of outcomes (outcome added & outcome removed), the system stops the current version of the market (outright markets), and generates a new version of the market (note that market status is changed to 0).

NOTE: One important detail with the free text outrights, is that there are some free text outrights where other outcomes and used, and thereby versions don't apply. For example; where outcomes are "yes" and "no". These outrights are not affected by turning on this option. So we would only offer "other" in a situation where there is a theoretical possibility for another winner (theoretical being a f1 grand prix race where maybe a driver can get sick etc.)

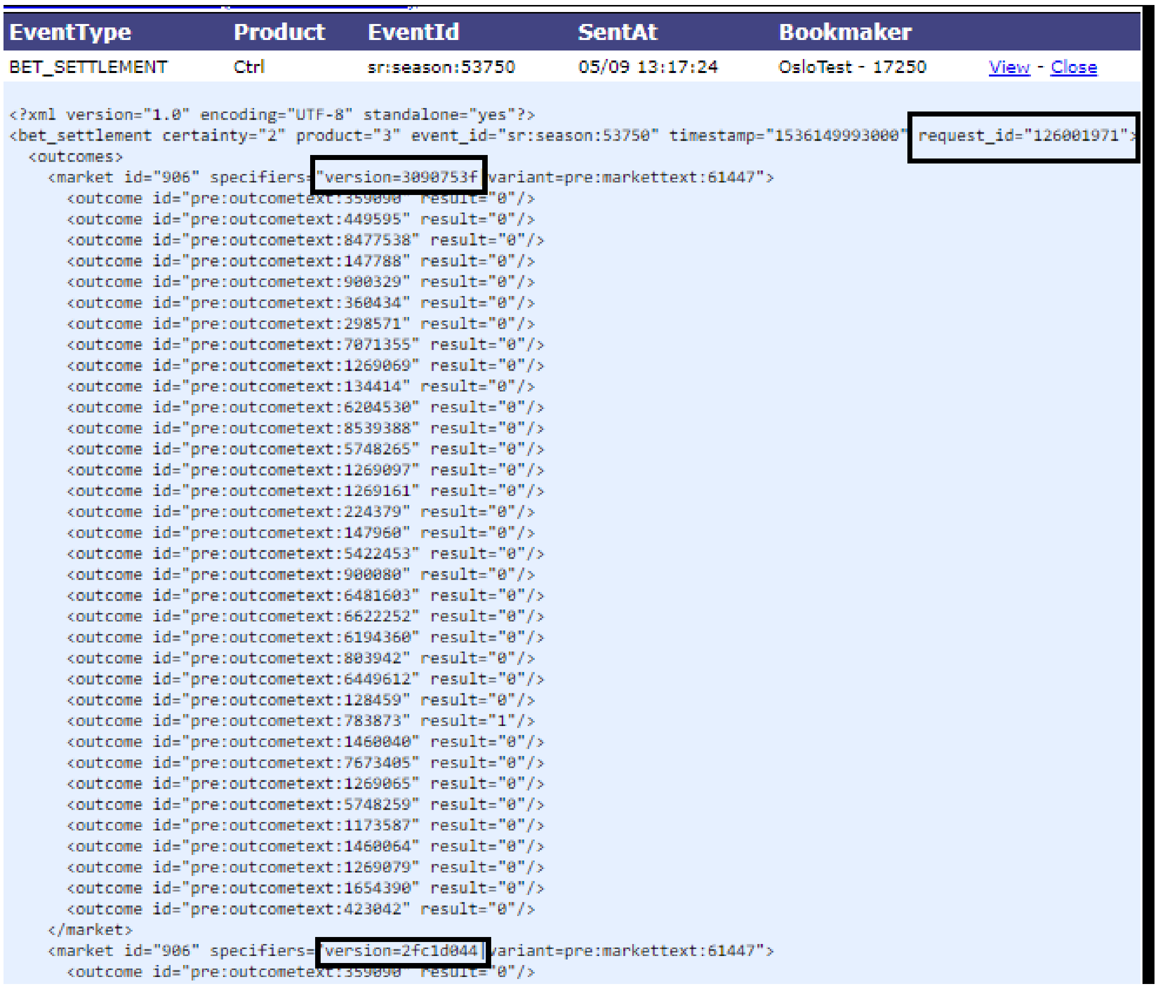


```
ODDS_CHANGE      Ctrl          sr:stage:342709      23/08 12:33:58      srqa - 16120         View - Close

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<odds_change product="3" event_id="sr:stage:342709" timestamp="1535027638435">
  <odds>
    <market favourite="1" status="1" id="906" specifiers="version=d3aded88|variant=pre:markettext:63069">
      <market_metadata next_betstop="1535210700000"/>
      <outcome id="pre:outcometext:6706858" active="0"/>
      <outcome id="pre:outcometext:6706873" odds="364.0" active="1"/>
      <outcome id="pre:outcometext:6706876" odds="556.5" active="1"/>
```

```
ODDS_CHANGE      Ctrl          sr:stage:342709      23/08 12:15:41      srqa - 16120         View - Close

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<odds_change product="3" event_id="sr:stage:342709" timestamp="1535026541776" request_id="4444">
  <odds>
    <market status="0" id="906" specifiers="version=c5be7cfd|variant=pre:markettext:63069"/>
  </odds>
</odds_change>
```

Figure 2 – Specifier identification

Settlements will still be done for all versions for one specific market, and only one version will be active at the same time:



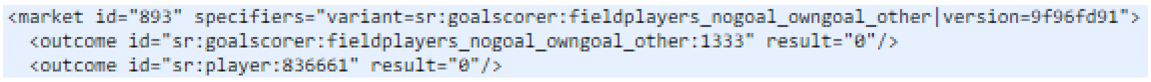
EventType	Product	EventId	SentAt	Bookmaker	
BET_SETTLEMENT	Ctrl	sr:season:53750	05/09 13:17:24	OsloTest - 17250	View - Close

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<bet_settlement certainty="2" product="3" event_id="sr:season:53750" timestamp="1536149993000" request_id="126001971">
  <outcomes>
    <market id="906" specifiers="version=3090753f|variant=pre:markettext:61447">
      <outcome id="pre:outcometext:359090" result="0"/>
      <outcome id="pre:outcometext:449595" result="0"/>
      <outcome id="pre:outcometext:8477538" result="0"/>
      <outcome id="pre:outcometext:147788" result="0"/>
      <outcome id="pre:outcometext:900329" result="0"/>
      <outcome id="pre:outcometext:360434" result="0"/>
      <outcome id="pre:outcometext:298571" result="0"/>
      <outcome id="pre:outcometext:7071355" result="0"/>
      <outcome id="pre:outcometext:1269069" result="0"/>
      <outcome id="pre:outcometext:134414" result="0"/>
      <outcome id="pre:outcometext:6204530" result="0"/>
      <outcome id="pre:outcometext:8539388" result="0"/>
      <outcome id="pre:outcometext:5748265" result="0"/>
      <outcome id="pre:outcometext:1269097" result="0"/>
      <outcome id="pre:outcometext:1269161" result="0"/>
      <outcome id="pre:outcometext:224379" result="0"/>
      <outcome id="pre:outcometext:147960" result="0"/>
      <outcome id="pre:outcometext:5422453" result="0"/>
      <outcome id="pre:outcometext:900080" result="0"/>
      <outcome id="pre:outcometext:6481603" result="0"/>
      <outcome id="pre:outcometext:6622252" result="0"/>
      <outcome id="pre:outcometext:6194360" result="0"/>
      <outcome id="pre:outcometext:803942" result="0"/>
      <outcome id="pre:outcometext:6449612" result="0"/>
      <outcome id="pre:outcometext:128459" result="0"/>
      <outcome id="pre:outcometext:783873" result="1"/>
      <outcome id="pre:outcometext:1460040" result="0"/>
      <outcome id="pre:outcometext:7673405" result="0"/>
      <outcome id="pre:outcometext:1269065" result="0"/>
      <outcome id="pre:outcometext:5748259" result="0"/>
      <outcome id="pre:outcometext:1173587" result="0"/>
      <outcome id="pre:outcometext:1460064" result="0"/>
      <outcome id="pre:outcometext:1269079" result="0"/>
      <outcome id="pre:outcometext:1654390" result="0"/>
      <outcome id="pre:outcometext:423042" result="0"/>
    </market>
    <market id="906" specifiers="version=2fc1d044|variant=pre:markettext:61447">
      <outcome id="pre:outcometext:359090" result="0"/>
```

Figure 3 - All outcomes

You can also configure to add the "*other*" outcome to all outrights where the field of participants is not complete. When participants change on such a market, we provide a version number.

It is important to note here that that different market IDs will be used when turning on the "other" outcome. An example would be a last goalscorer market having the ID of 39, will be sent as ID 893 when the option is selected (seen in the image below or table.3).



```
<market id="893" specifiers="variant=sr:goalscorer:fieldplayers_nogoal_owngoal_other|version=9f96fd91">
  <outcome id="sr:goalscorer:fieldplayers_nogoal_owngoal_other:1333" result="0"/>
  <outcome id="sr:player:836661" result="0"/>
```

Figure 4 - Version number

This helps you to distinguish the outright markets where participants have changed, and helps you identifying the most recent one. It guarantees that all versions of such an outright are settled accordingly when you use the “*other*” outcome.

All versions will also be included in a recovery request.

Market mapping for outcome versions for outright markets

The unified feed ID open market = market with version specifier:

Market name	UF ID (original ID)	UF ID open market
Championship outright	534	906
Short term outright	535	907
Multiwinner outright	536	908
First goalscorer	38	892
Last goalscorer	39	893

Table 5 - Market mapping for outright markets with outcome versions

Free-text markets versus structured markets

For now, outrights are offered as free-text markets. This will change over time, and more and more of the markets will be offered as structured markets instead. This gives more flexibility for odds-key configuration, and also give better ability to find bet constraints (i.e. two different bets on variations of the same outcome).

Free-text outright markets are market 534 (Championship), 535 (Short-term), 536 (Multi-winner), and 559. These markets have a variant specifier that describes the outright, using the normal variant specifier logic. The SDKs takes care of this automatically.

Structured outright markets was introduced for the Virtual Sports, and Sportradar will follow-up with a transition of some major outrights markets to structured form. These markets will have their own ids and will not use variants. For example, there will be a “Winner of {$event}” market where each outcome is one of the competitors competing in the competition (league, race etc).

Some structured outrights markets have multi-competitor-outcomes. Here more than one competitor is listed in the same outcome, using each competitor’s id separated by comma. This allows the client to detect related bets accurately i.e.:

"Who will win the event – Germany (sr:competitor:1234)" and "What teams will be placed top-3 – Germany(sr:competitor:1234), France, Spain" can be considered related bets as they include the same exact competitor, rather than relying on name-matching, which does not work reliably.

Some structured outrights markets will also use fixed structured outcomes such as "Yes", "No" and "Over", "Under".

Head2Head markets

Head2Head markets are ordinary markets with the two head2head competitors being the two outcomes. Each outcome's id is the competitor id for that outcome.

Head2head markets are currently associated with a broad sr:simple_tournament that is used for all head2head offers for the same sport no matter what actual race it is (just like the LCoO feed works today).

Note: This will change, and the head2head markets will be associated directly with the exact race-event.

Golf Three-balls markets

Each three-ball is 3-way market with the outcomes being the different players/competitors. Each outcome's id is the golf-player/competitor id for that outcome.

Missing Markets

Currently, some outrights available in LCoO are not available in Unified Odds. This is typically markets where Betradar has not yet associated the outright with a particular sport-event. These will be continually added over time.

1.6.5. Producers

Unified Feed gets data from several different producers. All of the different messages can be looked up in more detail in here. All producers have their own ID and name, with an added description. See the XML example below for details about some of our different producers and their scope.

XML example:

```
<producers response_code="OK">
    <producer id="1" name="LO" description="Live Odds"
api_url="https://api.betradar.com/v1/liveodds/" active="true" scope="live"
stateful_recovery_window_in_minutes="4320"/>
    <producer id="3" name="Ctrl" description="Betradar Ctrl"
api_url="https://api.betradar.com/v1/pre/" active="true" scope="prematch"
stateful_recovery_window_in_minutes="4320"/>
    <producer id="4" name="BetPal" description="BetPal"
api_url="https://api.betradar.com/v1/betpal/" active="true" scope="live"
stateful_recovery_window_in_minutes="4320"/>
    <producer id="5" name="PremiumCricket" description="Premium Cricket"
api_url="https://api.betradar.com/v1/premium_cricket/" active="false"
scope="live|prematch" stateful_recovery_window_in_minutes="4320"/>
    <producer id="6" name="VF" description="Virtual football"
api_url="https://api.betradar.com/v1/vf/" active="true" scope="virtual"
stateful_recovery_window_in_minutes="180"/>
    <producer id="7" name="WNS" description="World Number Service"
api_url="https://api.betradar.com/v1/wns/" active="false" scope="prematch"
stateful_recovery_window_in_minutes="4320"/>
    <producer id="8" name="VBL" description="Virtual Basketball League"
api_url="https://api.betradar.com/v1/vbl/" active="true" scope="virtual"
stateful_recovery_window_in_minutes="180"/>
    <producer id="9" name="VTO" description="Virtual Tennis Open"
api_url="https://api.betradar.com/v1/vto/" active="true" scope="virtual"
stateful_recovery_window_in_minutes="180"/>
    <producer id="10" name="VDR" description="Virtual Dog Racing"
api_url="https://api.betradar.com/v1/vdr/" active="true" scope="virtual"
stateful_recovery_window_in_minutes="180"/>
    <producer id="11" name="VHC" description="Virtual Horse Classics"
api_url="https://api.betradar.com/v1/vhc/" active="true" scope="virtual"
stateful_recovery_window_in_minutes="180"/>
    <producer id="12" name="VTI" description="Virtual Tennis In-Play"
api_url="https://api.betradar.com/v1/vti/" active="true" scope="virtual"
stateful_recovery_window_in_minutes="180"/>
</producers>
```

Virtual sports in Unified Odds

Each Virtual Sport is exposed as a separate producer in Unified Odds. This is a reflection of the underlying platform where each virtual sport has its own simulator/match-generator.

Virtual sports currently offer prematch odds and outrights markets. There is some work ongoing for in-game odds too, but this is currently not offered. However, everything is distributed in the normal Unified Odds format using normal Unified Odds markets.

There are however a few noteworthy differences to take into consideration:

- As stated above each virtual sport is a separated producer, which means:

    - Recovery needs to be done for each producer individually. This is handled automatically by the Sportradar SDKs.
    - Failure handling might have to consider what happens when one of the virtual sports becomes unavailable

- Virtual sports have their own unique ids (e.g. vf:match:1234 and vf:season:3456). The actual details of the matches/events aren't that important, because the results are displayed for end-users anyhow in the front-end interfaces, and the bet_settlement message includes the result. However, the season information is important and this is looked-up using the standard endpoints: tournaments/(season-id)/info.xml and tournaments/(season-id)/schedule.xml and follows the same format as real sports. The SDK automatically takes care of this so if you are an SDK user you don't have to worry.
    - Due to the nature of virtual sports, matches are played match more rapidly, and a full season don't last for more than a few hours.
    - Even if it is prematch odds, the odds update speed is more like live odds than prematch odds.
    - To ensure that the client has the schedule for the upcoming season and can cache the information about the matches that will be played, the producer is sending out a fixture_change with the season as the event when a new season starts. The recommended behavior when receiving such a fixture change is to go and fetch the season schedule using the endpoint *tournaments/(season-id)/schedule*. The SDK handles this automatically, but you may want to do something extra on your end when a new virtual season starts, if so a good time to do that is when you receive the fixture change on a virtual season.
- There are a few structured outrights markets provided. If you haven't yet implemented outrights, you may have to make some updates to handle outrights.
    - Keep in mind that an outright is seen as a market and the event is typically the season for the league/tournament or the race tournament.
    - Outrights may have competitor outcomes. Here each outcome is a competitor-id. Competitor outcomes are not seen for non-outright-markets.

Virtual Sports has some multi-competitor-outcome rights: like "*Who will be the top-3 in the league*". This is a special type of market where each outcome is a comma separated list of competitor-ids. Sportradar intends to use the same structured format for multi-competitor-outcomes for real sport outrights going forward too, but they are not offered this way currently. You may have to update your code to handle these multi-competitor-outcomes properly.

Mapping towards the legacy LCoO feed

Table 6 - Unified odds and LCoO market mapping

Unified Odds Market Id	LCoO Type	Comment
534	30	Championship
535	40	Short-term
536	50	Podium / Multiwinner
559	30,40	Free text outright

1.6.6. Bet settlement/Bet clearing

As a general rule, Betradar sends out bet-settlement information for any market line that Betradar has provided odds for. Bet settlement information can be sent either live (as soon as the important event has happened) or after the match (when the results have been confirmed and finalized). This means the client system will typically receive two bet_settlement messages for the same event. The client system can see from the bet-settlement message what certainty-level the bet-settlement is in (live-scouted early bet-settlement) or confirmed (late bet_settlement), and decide based on that information whether to act on the early or late bet-settlement information.

Note: *Live* bet settlement information is only provided for matches that Betradar covers live. Through the API using the schedule.xml or fixture.xml endpoints, you can find out whether a match is covered live or not.

The *liveodds* attribute indicates whether this match is booked by you for live coverage:

```
<sport_event liveodds="not_available"> Event is not available for you
<sport_event liveodds="booked">         Event is booked by you
<sport_event liveodds="bookable">       Event is bookable by you
<sport_event liveodds="buyable">        Event is buyable for you
```

1.6.7. Transition from Pre-match to Live

Minutes before a match starts, Betradar's live operators start covering the match (if they are scheduled to cover the match live).

The move from pre-match odds to live odds is seamless – there will just be an odds_change message for the match, including the updated odds.

First you will receive an odds_change message from the producer like this:

```
<odds_change product="1" event_id="sr:match:14011583"
timestamp="1522772129383">
  <sport_event_status status="0" match_status="0"/>
  <odds>
    <market status="1" id="16" specifiers="hcp=-1.75"
extended_specifiers="hcp_for_the_rest=-1.75">
      <outcome id="1714" odds="2.25" probabilities="0.4175194696"
active="1"/>
      <outcome id="1715" odds="1.65" probabilities="0.5824805304"
active="1"/>
    </market>

    <market favourite="1" status="1" id="546">
      <outcome id="1718" odds="2.0" probabilities="0.4652838638"
active="1"/>
      <outcome id="1719" odds="2.25" probabilities="0.4140078768"
active="1"/>
      <outcome id="1720" odds="2.35" probabilities="0.3956347016"
active="1"/>
      <outcome id="156" odds="29.0" probabilities="9.556228E-4"
active="1"/>
    </market>
```

Then after, the pre-match producer will send an odds_change message where inactive/deactivated markets are marked with status="0" (inactive/deactivated). For markets that are *active* and *handed over* to the live odds producer, these are marked with status="-2" (handed over).

```
<odds_change product="3" event_id="sr:match:14011583"
timestamp="1522772130071">
  <sport_event_status status="0" match_status="0"/>
  <odds_generation_properties expected_totals="3.21868"
expected_supremacy="1.54487"/>
  <odds>
    <market status="-2" id="546"/>
    <market status="0" id="156"/>
    <market status="0" id="542"/>
```

When Betradar is **not** covering the match live, a *betstop* message is sent out before the match is scheduled to start from the pre-match producer.

```
<bet_stop event_id="sr:match: 14011583" groups="all" product="3"
timestamp="1531465374159"/>
```

1.6.8. Fixtures

Fixture changes for a particular event are provided in the XML feed. However, you can request additional fixture information from the API or by using the SDK.

References

Unified feed uses multiple types of references. A reference in Unified Feed displays what producer reference a particular fixture is using. This is displayed inside the *reference_id* element.

Rotation numbers

Rotation numbers are issued by *Don Best*. They comprise a numbering scheme used by most sports books in the North American market. They define the participants of a contest such that bettors and purveyors can easily refer to the same contest. Typically, the away team's rotation number is an odd value. Rotation Numbers are not unique over the course of a season. They are however unique to the current schedule view.

1.7. Environments

Betradar provides three different environments for production testing & development when integrating towards the Unified Odds Feed. Production, integration and a replay server:

1.7.1. Production

This environment is only available to production customers. Available 24/7. Provides real-time messaging and an up to date information about various ongoing sport events. This is what you should use as the source in production.

- Messaging host: mq.betradar.com
- API server: api.betradar.com

1.7.2. Integration

Provides live updates just like production, but only available 24/5 (i.e. weekdays). Clients will be disconnected during weekends and will not be allowed to establish new sessions during this period (The replay environment is still available). The integration environment is available to customers for test/development purposes, both for production customers and customers under early implementation. Live Odds are slightly delayed in this environment (about 1 minute). Also includes the latest features and additions before they become available in the production environment. You can find the integration server self-service API documentation HERE.

- Messaging host: stgmq.betradar.com
- API server: stgapi.betradar.com
- Access Tokens: stgufadmin.betradar.com

The integration environment is available from *Monday* 00:00:00 UTC until *Saturday* 00:00:00 UTC

XML sent logs endpoint

It is possible for clients to fetch XML logs for events, or for a certain interval on the integration environtment. These are available through the endpoint */xmllog/events* and the */xmllog/messages* endpoints found in the self-service API HERE.

To fetch the logs for an *event*, please provide the match ID for the sport event to get the log. E.g: event_id="**sr:match:14392962**".

To fetch a *specific interval*, please provide a timestamp in milliseconds for the start and end of the desired interval. An interval can be maximum 1 hour long, and is only available for 7 days. E.g:

```
<odds_change event_id="sr:match:14392962" product="1"
timestamp="1526250693375">
<odds_change event_id="sr:match:14392962" product="1"
timestamp="1526250725288">
```

Also, a maximum of 1000 messages will be returned on a single request.

Note:

- A different access token is required to access the integration environment. Use *stgufadmin.betradar.com* to generate the required access token.
- You will need to book matches on the integration environment for live odds using the API endpoint available HERE: */liveodds/booking-calendar/events/sr:match:{id}/book*, as matches are not replicated from the production environment.

1.7.3. Replay Server

Note: Both *production* access tokens and *integration* access tokens can be used with the replay server.

The replay server allows you to replay the messages for various events exactly as if it was a normal match, and can also be sped up if needed. Includes all events older than 48 hours, and also provides special canned synthetic scenarios that allows you to easily test special boundary conditions. The replay server also allows you to replay a large number of events in parallel, which provides you with an optimum way to test how your application handles peak loads. You can configure whether to resend the messages as sent, or slightly updated (e.g. using current timestamps etc.).

- Messaging host: replaymq.betradar.com
- API server: api.betradar.com

For more information about the replay server, see our dedicated SDK and API sections for this environment.

1.8. Language support

Unified Odds supports the same languages as is supported for other Betradar products. The translations are available for sport names, category names, tournament names, team names, player names. If a particular translation is not available for a particular entity, the system defaults to *English*.

Market names are normally translated in all supported languages, and you can also specify your own translations using translation tools.

1.8.1. Country and language codes

What we call country_code is based on ISO 3166-1, meaning we use ISO 3166-1 wherever possible. However, when needed and ISO 3166-1 does not have a representation, we will provide another code. All country codes we use are provided in the tables below for language translations and country codes.

We also have the IOC-codes that we will provide in APIs wherever applicable, and as needed.

Table 7 - Language support

Language code	Language	Language code	Language
`sqi`	Albanian	`ja`	Japanese
`aa`	Arabic	`kaz`	Kazakh
`aze`	Azerbaijan	`ko`	Korean
`bs`	Bosnian	`lv`	Latvian
`br`	Brazilian Portuguese	`lt`	Lithuanian
`bg`	Bulgarian	`ml`	Macedonian
`zh`	Chinese (simplified)	`no`	Norwegian
`zht`	Chinese (traditional)	`pl`	Polish
`hr`	Croatian	`pt`	Portuguese
`cz`	Czech	`ro`	Romanian
`da`	Danish	`ru`	Russian
`nl`	Dutsch	`sr`	Serbian
`en`	English	`srl`	Serbian Latin
`et`	Estonian	`sk`	Slovak
`fi`	Finnish	`sl`	Slovenian
`fr`	French	`es`	Spanish
`ka`	Georgian	`se`	Swedish
`de`	German	`th`	Thai
`el`	Greek	`tr`	Turkish
`heb`	Hebrew	`ukr`	Ukranian
`hu`	Hungarian	`vi`	Vietnamese
`Id`	Indonesian	`it`	Italian

As explained above, country codes are based on the ISO 3166-1 and IOC-codes wherever possible, provided in the following table. For a comparison between the different codes, please reference THIS table.

Table 8 - Country codes

Country	Country A3 code
Afghanistan	AFG
Åland Islands	ALA
Albania	ALB
Algeria	DZA
American Samoa	ASM
Andorra	AND
Angola	AGO
Anguilla	AIA
Antarctica	ATA
Antigua and Barbuda	ATG
Argentina	ARG
Armenia	ARM
Aruba	ABW
Australia	AUS
Austria	AUT
Azerbaijan	AZE
The Bahamas	BHS
Bahrain	BHR
Bangladesh	BGD
Barbados	BRB

Belarus	BLR
Belgium	BEL
Belize	BLZ
Benin	BEN
Bermuda	BMU
Bhutan	BTN
Bolivia	BOL
Caribbean Netherlands	BES
Bosnia and Herzegovina	BIH
Botswana	BWA
Bouvet Island	BVT
Brazil	BRA
British Indian Ocean Territory	IOT
British Virgin Islands	VGB
Brunei	BRN
Bulgaria	BGR
Burkina Faso	BFA
Burundi	BDI
Cambodia	KHM
Cameroon	CMR
Canada	CAN
Cape Verde	CPV
Cayman Islands	CYM
Central African Republic	CAF

Chad	TCD
Chile	CHL
China, People's Republic of	CHN
Christmas Island	CXR
Cocos (Keeling) Islands	CCK
Colombia	COL
Comoros	COM
Congo, Democratic Republic of the	COD
Congo, Republic of	COG
Cook Islands	COK
Costa Rica	CRI
Côte d'Ivoire	CIV
Croatia	HRV
Cuba	CUB
Curaçao	CUW
Cyprus	CYP
Czech Republic	CZE
Denmark	DNK
Djibouti	DJI
Dominica	DMA
Dominican Republic	DOM
Ecuador	ECU
Egypt	EGY
El Salvador	SLV

England	ENG
Equatorial Guinea	GNQ
Eritrea	ERI
Estonia	EST
Ethiopia	ETH
Falkland Islands	FLK
Faroe Islands	FRO
Fiji	FJI
Finland	FIN
France	FRA
French Guiana	GUF
French Polynesia	PYF
French Southern and Antarctic Lands	ATF
Gabon	GAB
The Gambia	GMB
Georgia	GEO
Germany	DEU
Ghana	GHA
Gibraltar	GIB
Greece	GRC
Greenland	GRL
Grenada	GRD
Guadeloupe	GLP
Guam	GUM

Guatemala	GTM
Guernsey	GGY
Guinea	GIN
Guinea-Bissau	GNB
Guyana	GUY
Haiti	HTI
Heard Island and McDonald Islands	HMD
Honduras	HND
Hong Kong	HKG
Hungary	HUN
Iceland	ISL
India	IND
Indonesia	IDN
Iran	IRN
Iraq	IRQ
Ireland	IRL
Isle of Man	IMN
Israel	ISR
Italy	ITA
Jamaica	JAM
Japan	JPN
Jersey	JEY
Jordan	JOR
Kazakhstan	KAZ

Kenya	KEN
Kiribati	KIR
Korea, Democratic People's Rep. (North)	PRK
Korea, Republic of (South)	KOR
Kuwait	KWT
Kyrgyzstan	KGZ
Laos	LAO
Latvia	LVA
Lebanon	LBN
Lesotho	LSO
Liberia	LBR
Libya	LBY
Liechtenstein	LIE
Lithuania	LTU
Luxembourg	LUX
Macau	MAC
Macedonia	MKD
Madagascar	MDG
Malawi	MWI
Malaysia	MYS
Maldives	MDV
Mali	MLI
Malta	MLT
Marshall Islands	MHL

Martinique	MTQ
Mauritania	MRT
Mauritius	MUS
Mayotte	MYT
Mexico	MEX
Micronesia, Federated States of	FSM
Moldova	MDA
Monaco	MCO
Mongolia	MNG
Montenegro	MNE
Montserrat	MSR
Morocco	MAR
Mozambique	MOZ
Myanmar	MMR
Namibia	NAM
Nauru	NRU
Nepal	NPL
Netherlands	NLD
New Caledonia	NCL
New Zealand	NZL
Nicaragua	NIC
Niger	NER
Nigeria	NGA
Niue	NIU

Norfolk Island	NFK
Northern Ireland	NIR
Northern Mariana Islands	MNP
Norway	NOR
Oman	OMN
Pakistan	PAK
Palau	PLW
Palestinian Authority	PSE
Panama	PAN
Papua New Guinea	PNG
Paraguay	PRY
Peru	PER
Philippines	PHL
Pitcairn Islands	PCN
Poland	POL
Portugal	PRT
Puerto Rico	PRI
Qatar	QAT
Réunion	REU
Romania	ROU
Russia	RUS
Rwanda	RWA
Saint Barthélemy	BLM
Saint Helena	SHN

Saint Kitts and Nevis	KNA
Saint Lucia	LCA
Saint Martin (French part)	MAF
Saint Pierre and Miquelon	SPM
Saint Vincent and the Grenadines	VCT
Samoa	WSM
San Marino	SMR
São Tomé and Príncipe	STP
Saudi Arabia	SAU
Scotland	SCO
Senegal	SEN
Serbia	SRB
Seychelles	SYC
Sierra Leone	SLE
Singapore	SGP
Sint Maarten (Dutch part)	SXM
Slovakia	SVK
Slovenia	SVN
Solomon Islands	SLB
Somalia	SOM
South Africa	ZAF
South Georgia and the South Sandwich Islands	SGS
Spain	ESP
Sri Lanka	LKA

Sudan	SDN
Suriname	SUR
Svalbard and Jan Mayen	SJM
Swaziland	SWZ
Sweden	SWE
Switzerland	CHE
Syria	SYR
Republic of China (Taiwan)	TWN
Tajikistan	TJK
Tanzania	TZA
Thailand	THA
Timor-Leste	TLS
Togo	TGO
Tokelau	TKL
Tonga	TON
Trinidad and Tobago	TTO
Tunisia	TUN
Turkey	TUR
Turkmenistan	TKM
Turks and Caicos Islands	TCA
Tuvalu	TUV
Uganda	UGA
Ukraine	UKR
United Arab Emirates	ARE

United Kingdom	GBR
United States	USA
United States Minor Outlying Islands	UMI
United States Virgin Islands	VIR
Uruguay	URY
Uzbekistan	UZB
Vanuatu	VUT
Vatican City State	VAT
Venezuela	VEN
Vietnam	VNM
Wales	WAL
Wallis and Futuna	WLF
Western Sahara	ESH
Yemen	YEM
Zambia	ZMB
Zimbabwe	ZWE

2. Unified Odds - Messages

2.1. Basic configuration

AMQP Server:	mq.betradar.com
AMQP Port:	5671 (standard SSL for AMQP)
Virtual Host:	/unifiedfeed/<bookmaker_id> (where <bookmaker_id> is your bookmaker id (see `users/whoami.xml` for how to find your bookmaker_id if you are uncertain)
Exchange:	unifiedfeed
Username:	<your_security_access_token>
Password:	<blank>

Queue:	You cannot create your own queues. Instead you have to request a server-named queue (empty queue name in the request). Passive, Exclusive, Non-durable.
Version:	AMQP 0.9.1

PHP example code:

```
$connection = new AMQPSSLConnection('mq.betradar.com', 5671, ‘your-
access-token’, 'your-access-token', '/unifiedfeed/your-id-here',
array('verify_peer' => false,    'verify_peer_name' => false,
'allow_self_signed' => true));
$channel = $connection->channel();
$queue_name = $channel->queue_declare('', false, false, true)[0];
$channel->queue_bind($queue_name, 'unifiedfeed', '#');
```

2.2. Authentication

For the messaging service you must provide your betradar.com security access token as username, and leave the password blank in your AMQP client (see your AMQP client for how to provide username & password).

2.3. Messages

In this chapter you will find all the different XML messages available Unified Feed.

2.3.1. Note on timestamps in unified feed

Every message in Unified Feed has a timestamp (when the message was created). Keep this is mind if some examples in this document are missing the timestamp, they have simply been removed from the examples in order to highlight the discussion topic.

In unified feed there are multiple types of messages that are produced. In some cases, there may be messages produced at the same time (two messages have the same timestamp). It is important to view this as a parallel process, and it may happen that i.e. A fixture change message ends up having the same timestamp as an odds change message. Even the same odds producer may generate multiple odds changes on the same millisecond, but this is unlikely to happen on the same sport event.

2.3.2. Message types

Table 9 - All available messages in the XML feed

Name	Priority	Description
`alive`	Low	If not received, something is broken/not working.
`bet_cancel`	Low	Cancel a market due to an error.
`bet_settlement`	Low	Settle/clear bets for the listed markets and outcomes for a particular event.
`bet_stop`	High	Specified group of markets for a specified sport event should all change their market status to either suspended or deactivated.
`fixture_change`	High	Sent when an important fixture change has happened (typically some near-term change, such as a new event or a change to an event (e.g. delayed match)).
`odds_change`	High	Lists odds changes for some or all markets for a match and signal bet stop for the whole match, individual markets, or outcomes.
`rollback_bet_settlement`	Low	Undo a previously sent bet settlement that was sent in error.
`rollback_bet_cancel`	Low	Undo a bet cancel that was sent by mistake.
`snapshot_complete`	None	Sent after all odds updates from a recovery API request has been sent

2.3.3. Message: Odds change

The *odds_change* messages are sent whenever Betradar has new odds for some markets for a match. An *odds_change* can include a subset of all markets; if so, markets not reported remain unchanged. All outcomes possible within a market are reported.

Table 10 – odds_change attributes

Name	Description
`event_id`	The ID of the event (match/race/outright) the odds information is about.
`timestamp`	When the message was created in the odds producing system (milliseconds since Epoch UTC).
`product`	Specifies which producer generated these odds. At any given point in time there should only be one product generating odds for a particular event. This tag can later be used when a producer

	is detected to be down. (**1** = LiveOdds producer, **3** = Betradar Ctrl producer, **4** = BetPal producer, **5** = Premium Cricket producer)
odds_change_reason	(Optional) can be set to "riskadjustment_update" if this message is caused by a manual odds change.
sport_event_status	Reports various status information about the event that is relevant (see the special section on sport_event_status below for further description).
odds	Describes all odds updates for this event per market.
odds.betstop_reason	Sent after a betstop while under betstop to signal the cause of the betstop.
odds.betting_status	When set signals that markets have been opened again after a betstop, but it is still early after betstop. Risk sensitive customers may decide to continue to keep the markets closed until betting_status is no longer present. The value of betting_status signals the cause for the previous betstop. (See the API endpoint descriptions/betting_status.xml for the description of the individual betting status values)
market	Describes the odds updates for a particular market.
market.id	The ID of the market (see markets.xml for a detailed description of various markets).
market.favourite	If present, this is set to 1, which states that this is the most balanced or recommended market line. This setting makes most sense for markets where multiple lines are provided (e.g. the Totals market). If this is NOT set to 1, it will not show up in the feed.
market.specifiers	A | separated list of key=value-pairs. All keys are specified in the Betting API in the market descriptions. These are further specifiers for the market.
market.status	Active/suspended/deactivated (active (**1**) = display odds & accept tickets, suspended (**-1**) = display odds & don't accept tickets, deactivated (**0**) = stop displaying odds, don't accept tickets). The default value is active if status is not present. During recovery the following additional status may also be sent: handed_over, cancelled and settled (**-3**). **See note below this table for additional information.**
m.market_metadata	Additional information about a specific market (see more below)
m.mm.next_betstop	Timestamp in UTC when to betstop this market. Typically used for outrights and typically is the start-time of the event the market refers to.
m.extended_specifiers	Additional info about this market that does not logically changes the market but is interesting for display purposes. Best example is Asian Handicap markets. Where the extended_specifers is to to hcp_for_the_rest_of_the_match

`odds_generation_properties`	Provided by the prematch odds producer only, and contains a few key-parameters that can be used in a client's own special odds model, or even offer spread betting bets based on it.
`ogp.expected_totals`	Parameter that can be used by client's own special odds model. Provided by prematch odds producer only. "*How many goals are expected in total*"?
`ogp.expected_supremacy`	Parameter that can be used by client's own special odds model. Provided by prematch odds producer only. "*How big is the expected goal supremacy"* (home goals minus away goals)?

Special note on market.status: If you do a recovery from a producer after it has handed over a match, then all markets will appear with the *handed_over* status, so unless you see these markets from another producer, you should treat them as closed.

```
<odds_change event_id="sr:match:1234" timestamp="1234" product="2">
<sport_event_status status="1" reporting="1" match_status="1"
home_score="2" away_score="0">
     <clock match_time="10:00" remaining_time="50:00" stopped=true"/>
   </sport_event_status>
  <odds>
    <market id="47" specifiers="score=41.5" favourite="1" status="1">
      <outcome id="1" odds="1.12" active="1"/>
      <outcome id="2" odds="1.92" active="1"/>
    </market>
    <market id="48" specifiers="score=42.5">
      <outcome id="1" odds="1.12" active="1"/>
      <outcome id="2" odds="1.92" active="1"/>
    </market>
    <market id="49" status="deactivated"/>
    <market id="123" specifiers="set=2|game=3|point=1" status="-1">
      <outcome id="1" odds="1.3" active="1"/>
      <outcome id="2" odds="1.7" active="1"/>
    </market>
    <market id="40">
      <outcome id="sr:player:1234" odds="1.4" active="1"/>
      <outcome id="sr:player:4322" odds="1.5" active="0"/>
      <outcome id="sr:player:71111" odds="1.87" active="1"/>
      <outcome id="sr:player:9919" odds="1.9" active="1"/>
      <outcome id="sr:player:1119" odds="2.1" active="0"/>
    </market>
  </odds>
</odds_change>
```

You can configure the *odds_change* message to receive probabilities too. These probabilities are then sent on each individual market outcome.

Note: The probability attribute will not be included on an outcome with a probability lower than *1e-10*.

```
<odds>
  <market id="47" specifiers="score=41.5" favourite="1" status="1">
    <outcome id="1" odds="1.12" probabilities="0.6836763″ active="1"/>
    <outcome id="2" odds="1.92" probabilities="0.8836763″ active="1"/>
  </market>
  <market id="48" specifiers="score=42.5">
    <outcome id="1" odds="1.12" probabilities="0.1836763″ active="1"/>
    <outcome id="2" odds="1.92" probabilities="0.3336763″ active="1"/>
  </market>
  <market id="49" status="deactivated"/>
  <market id="123" specifiers="set=2|game=3|point=1" status="-1">
    <outcome id="1" odds="1.3" probabilities="0.77346763″ active="1"/>
    <outcome id="2" odds="1.7" probabilities="0.32946763″ active="1"/>
  </market>
</odds>
```

2.3.4. Message: Bet stop

The *bet_stop* message is an optimized signal to indicate that all, or a set of markets should be instantly suspended (continue to display odds, but don't accept tickets). The same effect can be achieved by sending an odds_changes message that lists all the affected markets and moves them to status="-1" (suspended).

It is important to keep in mind that only active markets should be set to suspended, and not markets that are already *deactivated*, *settled* or *cancelled*. This is also the case for the attribute *market_status* (explained in Table 9 below). If it is not present, the market should be moved to suspended. However, if the market is already *deactivated*, *settled* or *cancelled* this is not a good practice. Only move ACTIVE markets to suspended.

The *bet_stop* is sent very rapidly, as soon as a Betradar operator detects an issue. At the time the bet_stop is sent, the cause for the betstop is not always available (typically not for live matches). The cause of the bet_stop is provided in a subsequent *odds_change* message (see THIS chapter).

Table 11 – bet_stop message attributes

Attribute	Description
`timestamp`	The timestamp (milliseconds since epoch UTC) for this message.
`product`	The Betradar producer that is sending this message **1**=LiveOdds **2**=MTS **3**=BetradarCtrl **4**=BetPal

	5=Premium Cricket See the endpoint descriptions/producers.xml for a listing of current producers.
`event_id`	What sport event this message refers to.
`groups`	A description of which set of markets should be suspended – the value should be a group-name as can be seen in the market-descriptions ('all' is a special keyword that means all markets for this event).
`market_status`	If not present, the markets specified should be moved to suspended. If present, they should be either suspended or deactivated based on the value of this field.
`betstop_reason`	If present, describes the reason for the bet stop. **NOTE**: Not all producers use this attribute (i.e: The *Ctrl* producer).

```
<bet_stop timestamp="12345" product="3" event_id="sr:match:471123"
groups="all"/>
```

Note: Currently there is a difference in how the Premium Cricket producer and Ctrl producer send bet_stop messages (pre-match only). In Ctrl you will receive a *bet_stop* at match kick off, but in Premium Cricket you will receive a normal *odds_change* message with the attribute status="0".

2.3.5. Message: Bet settlement

If something happens in a match that defines the outcome of a specific odds type, you will receive a *bet_settlement* message that contains information about the outcome of bets. And, in some cases, whether the bet was voided or not. If the bet is (partly) voided, the voided part should be refunded to the customer.

The following list includes all possible combinations of *outcome (result)* and *void_factor*:

- result="0" and no void_factor: Lose entire bet
- result="1" and no void_factor: Win entire bet
- result="0" and void_factor="1": Refund entire bet
- result="1" and void_factor="0.5": Refund half bet and win other half
- result="0" and void_factor="0.5": Refund half bet and lose other half.

Note! The evaluation of bet outcomes is the responsibility of the Client system.

The client system will often receive two bet_settements for the same outcome – one immediately after the match ends caused by the live scout, and a second message when the results have been officially confirmed. The two messages have different certainty-levels to indicate the difference. In almost all cases the outcome results will be the same. In extra-ordinary cases the results may differ, and the Client system will have to decide how to handle this. (One type of client system may always wait for the official confirmed results; another type may primarily use the live results).

Table 12 – bet_settlement attributes

Name	Description
`event_id`	The ID of the event this bet settlement refers to.
`product`	The producer that generated this bet settlement information **1**=LiveOdds **2**=MTS **3**=BetradarCtrl **4**=BetPal **5**=PremiumCricket Etc…
`timestamp`	When this message was generated.
`certainty`	Is this bet-settlement sent as a consequence of scouts reporting the results live (1) or is this bet-settlement sent post-match when the official results have been confirmed (2)
`market.void_reason`	Describes the reason for voiding certain outcomes for a particular market. Only set if at least one of the outcomes have a void_factor. (See the API-endpoint void_reasons for descriptions of the various void_reasons)
`outcome.id`	A unique identifier illustrated by a number or number/text value.
`outcome.result`	Set to 1 if this outcome was the winning outcome, set to 0 if it is a losing outcome.
`outcome.void_factor`	If the bet on an outcome should be refunded completely void-factor is set to 1.0. If half of the bet on an outcome should be refunded void_factor is set to 0.5.
`outcome.dead_heat_factor`	A dead-heat factor may be returned for markets where a bet has be placed on a particular team/player to place and this particular player has placed but the place is shared with multiple players, reducing the payout.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<bet_settlement certainty="2" product="3" event_id="sr:match:16807109"
timestamp="1547538073717">
  <outcomes>
    <market id="193">
      <outcome id="74" result="0"/>
      <outcome id="76" result="1"/>
    </market>
    <market id="204" specifiers="setnr=1|total=9.5">
      <outcome id="12" result="0"/>
      <outcome id="13" result="1"/>
    </market>
    <market id="201">
      <outcome id="863" result="0"/>
      <outcome id="864" result="0"/>
      <outcome id="861" result="1"/>
      <outcome id="862" result="0"/>
    </market>
    </outcomes>
</bet_settlement>
```

bet_settlement and results

You should NOT use bet_settlement messages as a signal that all results for a match are available. If the certainty is confirmed, you are guaranteed that the bet_settlements are confirmed as stated. However, you are **not** guaranteed that all results are available just because you receive a bet_settlement with certainty=”2” (confirmed). For example: there might be additional result information that may come in later, but we may have enough results to send out bet_settlement messages for all the markets we have offered.

2.3.6. Message: Rollback bet settlement

“Rollback_bet_settlement” is sent when a previously sent ‘bet_settlement’ was sent in error and needs to be rolled back/undone.

```
<rollback_bet_settlement event_id="sr:match:299321" timestamp="1236”
product="1">
   <market id="47"/>
</rollback_bet_settlement>
```

2.3.7. Message: Bet cancel

A *bet_cancel* message is sent when a bet made on a particular market needs to be cancelled and refunded due to an error (which is different to a bet-settlement/refund).

A bet_cancel can be sent for a market that has already been settled. This however only happens in rare circumstances when a market that should have been cancelled has been settled in error in a previous message.

Table 13 – bet_cancel attributes

Name	Description
event_id	The ID of the event this cancel bet is for.
timestamp	When this event was generated (seconds since Epoch UTC).
product	The producer that generated this message (1=LiveOdds, 2=MTS, 3=BetradarCtrl, 4=BetPal, 5=premium cricket).
start_time end_time	If start and end time are specified, they designate a range in time for which bets made should be cancelled. If there is an end_time but no start_time, this means cancel all bets placed before the specified time. If there is a start_time but no end_time this means, cancel all bets placed after the specified start_time.
superceded_by	Used in rare cases for outright odds, if the original tournament season id was wrong, the old odds_changes are bet_cancelled and the new tournament season id is included in the superceded_by attribute. A client seeing this bet_cancel has at least two choices: **1)** Just ignore the superceded_by attribute completely and process the bet_cancel as any betcancel (i.e. void bets placed if any) **2)** If possible move placed bets to the new tournament season id.

```
<bet_cancel event_id="sr:match:4711" start_time="1212000"
end_time="1223000" timestamp="1234000">
      <market id="48" specifiers="score=41.5"/>
</bet_cancel>
```

2.3.8. Message: Rollback bet cancel

A *Rollback_bet_cancel* message is sent when a previous bet cancel should be undone (if possible). This may happen, for example, if a Betradar operator mistakenly cancels the wrong market (resulting in a *bet_cancel* being sent) during the game; before realizing the mistake.

```
<rollback_bet_cancel event_id="sr:match:4444" timestamp="1239">
   <market id="48" specifiers="score=41.5"/>
</rollback_bet_cancel>
```

2.3.9. Message: Fixture change

A *fixture_change* message is sent when a Betradar system has made a fixture change it deems as important. These are typically changes that affect events in the near-term (e.g. a match was added that starts in the next few minutes, a match was delayed and starts in a couple of minutes, etc.). The message is short and includes a bare minimum of relevant details about the addition/change. The recommended practice is to always to do a follow-up API call to lookup the updated fixture information.

We might cancel the coverage of a match before it starts, or in the middle of the match. This might cause the sport_event_status status=”3” (match is ended) to not occur. However, a fixture_change message will be sent instead if this happens (see the dedicated Sport Event Status section for more information about this element).

For virtual sports a fixture change with a season id is sent out at the start of a new season. This is an indication that the new season schedule is available and could be read and cached, before the virtual sports odds_changes messages start arriving. The SDKs will automatically handle this and read in the next season. However, the client system may also want to do something particular when a new season starts.

Table 14 - fixture_change attributes

Name	Description
`event_id`	The match/race/tournament this fixture change is for.
`change_type`	(Optional) if specified, declares what type of change it is (new, start time, coverage). For a start time change or coverage change the details are in the message. Otherwise, the new fixture has to be requested from the API. **1**=NEW, **2**=DATETIME, **3**=CANCELLED, **4**=FORMAT, **5**=COVERAGE
`product`	The producer that generated this message (**1**=LiveOdds, **2**=MTS, **3**=BetradarCtrl, **4**=BetPal, **5**=Premium Cricket).

```
<!-- Unspecified type of change -->
<fixture_change event_id="sr:match:1234" product="3"/>

<!-- New -->
<fixture_change event_id="sr:match:1234" change_type="1" product="1"/>

<!-- Coverage change -->
<fixture_change event_id="sr:match:1234" change_type="5" product="3"/>

<!-- Start time change -->
<fixture_change event_id="sr:match:1234 " change_type="2" product="1"/>
```

The change_type attribute (if present), describes what type of change that caused the message to be sent. In general, best practices are to always re-fetch the updated fixture from the API and not solely rely on the change_type and the message content. This is because multiple different changes could have been made. Here is a listing of possible change_types:

Table 15 - Change types

Change	Id	Description
`NEW`	1	This is a new match/event that has been just added.
`DATE_TIME`	2	Start-time update
`CANCELLED`	3	This sport event will not take place. It has been cancelled.
`FORMAT`	4	The format of the sport-event has been updated (e.g. the number of sets to play has been updated or the length of the match etc.)
`COVERAGE`	5	Coverage update. Sent for example when liveodds coverage for some reason cannot be offered for a match.

2.3.10. Message: Alive

Alive messages are sent by each producer every 10 seconds. This is indicating that the given producer is operating normally and you are able to receive messages from it. You can also see from the message if you are subscribed to that producer or not.

If it should happen that you stop receiving alive messages at the expected intervals, it is indicating something might be wrong on the connection between our service and yours. What is important to look at is the timestamp inside the alive messages, and the time interval you are receiving them. If the timestamp inside the messages is consistent at 10 seconds, but you receive the messages at a slower pace or irregular interval, this might indicate a bad network connection between your servers and our service, or slow processing on your side. If the timestamp inside the alive messages is not at an approximate 10 seconds interval, this indicates something is wrong on our side. Both these situations are denoted as not receiving an alive message in the below.

- The recommended action is described below, and you can only treat the connection as up again once a recovery has been completed (confirmed with a snapshot_complete message)
- If you do not receive an alive message for more than 15 seconds you must wait until you start receiving alive messages again, if needed reconnecting to the message broker, and then issue a recovery from the time you last received a message.

- **Be aware** that if you start issuing recovery requests before you start receiving alive messages, the requests will not be successful, and you might hit our rate limiting on the recovery endpoint.

If you have not received an alive message, and successfully completed a recovery from a given producer for a certain amount of time, you should act according to the event state (pre match or in play). Here are some recommendations, but you are free to use other thresholds for the actions:

- If the event is pre-match, and kicking off in more than 5 minutes, and you have not received an alive message in 5 minutes, then you should suspend all markets on that event. *We recommend to suspend all markets. However, to avoid losing business, it might be good to keep accepting bets in a restricted way (e.g. increased odds key, reduced limits, limited market offers, etc.). **Note** events which are scheduled to start in the next 5 minutes have to be suspended until the system is fully recovered.*
- If the event is in play or kicking off in less than 5 minutes, and you have not received an alive message in 15 seconds, you should suspend all markets.
- Events from the Virtual producers should be treated as if the event is in play.
- Events from the Numbers Betting producer should be treated as if it is pre event.

The alive message is also sent to all customers with the subscribed attribute set to 0 when a product is back after downtime. A client that sees one of these messages must instantly initiate the recovery sequence for the particular product (as outlined below) to get the current state and re-subscribe to messages.

For all producers you should only reopen bets after you have successfully completed a recovery. Please contact your Betradar representative if you believe this has occurred and no warning was provided beforehand.

Table 16 - Example showing how to act for an in play event

Time/seconds	10	24	30	40	45	50	56	60	70
Alive message from live producer	OK	OK	OK				OK	OK	OK
In play event status	Active	Active	Active	Active	Suspended	Suspended	Suspended	Active	Active

Connection handling	-	-	-	-	-	-	Issue recovery	Snapshot complete	-

Table 17 - Example showing how to act for a pre-match event

Time/minutes:seconds	1:00	1:10	1:20…	2:00	2:10…	7:00	10:00…	11:00…	12:00…
Alive message from pre match producer	OK			OK			OK	OK	OK
Pre match event status	Active	Active	Active	Active	Active	Suspended	Suspended	Active	Active
Connection handling	-	-	-	-	-	-	Issue recovery	Snapshot complete	-

XML example of an alive message

```
<alive timestamp="1234579" product="2" subscribed="1"/>
```

Table 18 - alive attributes

Name	Description
`timestamp`	Timestamp in milliseconds since epoch when this message was generated according to generating system's clock.
`product`	The producer that sent this alive message.
`subscribed`	If set to 0 this means the product is up again after downtime, and the receiving client will have to issue recovery messages against the API to start receiving any additional messages and get the current state.

*All available producers for the **product** attribute in the above table*:

```
<producers response_code="OK">
    <producer id="1" name="LO" description="Live Odds"
    <producer id="3" name="Ctrl" description="Betradar Ctrl"
    <producer id="4" name="BetPal" description="BetPal"
    <producer id="5" name="PremiumCricket" description="Premium Cricket"
    <producer id="6" name="VF" description="Virtual football"
    <producer id="7" name="WNS" description="Numbers Betting"
    <producer id="8" name="VBL" description="Virtual Basketball League"
    <producer id="9" name="VTO" description="Virtual Tennis Open"
</producers>
```

2.3.11. Message: Snapshot complete

This message indicates that all messages relating to an initiate request for odds from the RESTful API have been processed. The request_id parameter returned is the request_id that was specified in the initiate_request POST to the API. If no request_id parameter was specified in the original request, no request_id parameter is present. It is highly recommended to set a request_id to some number.

```
<snapshot_complete request_id="1234" timestamp="1234578" product="3"/>
```

2.4. Sport event status element

The element "*sport_event_status*" is provided in the odds_change message. *Status* is the only required attribute for this element, and this attribute describes the current status of the sport-event itself (not started, live, ended, closed). Additional attributes are live-only attributes, and only provided while the match is live; additionally, which attributes are provided depends on the sport. The following table lists the various attributes and their meaning:

Table 19 - sport_event_status element

Attribute name	Description	Format/Values	Sports
`Status` **(Required)**	High-level generic status of the match.	0 (not started), 1 (live), 3 (ended), 4 (closed) - only one of these 4 is possible in the odds_change message – However please note that other "stopped" states are available in the API, but never in the odds_change message	All
`reporting`	Does Betradar have a scout watching the game.	Active(1)/suspended(-1)/not available(0) - only present when status = live or suspended	All
`home_score`	Current score for the home team.	Number	All match sports
`away_score`	Current score for the away team.	Number	All match sports
`match_status`	Sports-specific integer code the represents the live match status (first period, 2nd break, etc.).	See specific section below.	All

`match_time`	The playing minute of the match. Or minute:seconds if available	44 29:20	Soccer, Ice Hockey
`current_server`	The player who has the serve at that moment.	1 or 2 (1 for team/player 1 (home), 2 for team/player 2 (away)	Tennis, Table-Tennis, Badminton, Squash
`home_gamescore`	The point score of the “home” player. The score will be 50 if the “home” player has advantage. This attribute is also used for the tiebreak score when the game is in a tiebreak.	Number (15 30 40 50)	Tennis
`away_gamescore`	The point score of the “away” player. The score will be 50 if the “away” player has advantage. This attribute is also used for the tiebreak score when the game is in a tiebreak.	Number	Tennis
`tiebreak`	Whether a match is in a tiebreak.	true (only present if tie-break)	Tennis
`expedite_mode`	Whether the expedite system is in operation or not.	true or false	Table Tennis
`home_suspend`	The amount of suspensions for the home team.	Int	Ice Hockey, Handball, Futsal
`away_suspend`	The amount of suspension for the away team	Int	Ice Hockey,

			Handball, Futsal
`strikes`	Number of strikes for the current batter.	Number 0..2	Baseball
`balls`	Number of balls for the current batter.	Number 0..3	Baseball
`outs`	Number of outs for the current batter.	Number 0..2	Baseball
`bases`	Which bases are loaded.	Number	Baseball
`home_batter`	Current batter for the home team.	Int	Baseball
`away_batter`	Current batter for the away team.	Int	Baseball
`possession`	The team that has the ball.	1 (home) or 2 (away)	American Football
`try`	Try number, it's the current "down" until reaching the 10 yards if defined.	Int	American Football
`yards`	Yards until first down. How many yards down in this drive (a series of offensive plays).	Int	American Football
`visit`	Which player is currently visiting.	1 (home) or 2 (away) (absence undefined)	Snooker
`remaining_reds`	Number of remaining red balls.	number	Snooker
`home_legscore`	The home player score at any given leg whether one game or during a match.	number	Darts
`away_legscore`	The away player score at any given leg	number	Darts

	whether one game or during a match		
`throw`	The player who had the first throw for the current leg.	1 (home) or 2 (away)	Darts
`visit`	The player visiting the board.	1 (home) or 2 (away)	Darts
`delivery`	The player who will deliver the next bowl.	1 (home) or 2 (away)	Bowls
`home_remaining_bowls`	The remaining number of bowls for the home team.	number >= 0	Bowls
`away_remaining_bowls`	The remaining number of bowls for the away team.	number >= 0	Bowls
`current_end`	Current end in set	Int	Bowls
`home_dismissal`	The number of dismissals during the current inning for the home team.	Int	Cricket
`away_dismissal`	The number of dismissals during the current inning for the away team.	Int	Cricket
`home_penalty_runs`	The number of penalty runs awarded to the home team during an over.	Int	Cricket
`away_penalty_runs`	The number of penalty runs awarded to the away team during an over.	Int	Cricket
`innings`	What innings it is.	number > 0 Note that the second innings can also be superover.	Cricket

`over`	What over in the inning it is.	number > 0 The first over will have value 1, please note this is different from the Cricket convention which would start counting at zero.	Cricket
`delivery`	Which ball in the over it is.	number 1..6	Cricket
`current_ct_team`	Which team is playing counter terrorists (1 = home, 2=away)	Number 1..2	CS:GO
`home_penalty_score`	Home team penalty score	Number > 0 In the event of a game being decided by a penalty shootout, then the goal will be added to the winning team's score (and game total) for settlement purposes.	Ice Hockey
`away_penalty_score`	Away team penalty score	Number > 0 In the event of a game being decided by a penalty shootout, then the goal will be added to the winning team's score (and game total) for settlement purposes.	Ice Hockey

```
<sport_event_status match_status_name="1st quarter" away_score="0"
home_score="0" match_status="13" reporting="1" status="1">
</sport_event_status>
```

2.4.1. Match status element

The XML element <*match_status*> is a sports-specific element provided during the sport event that describes what period or break the match is currently in. An example would be before a match, the match status has ID="0", meaning the match is "*not_started*". During the match under normal circumstances, *match-status* is represented as an integer code as the game/match is played. Below you can see an example of how the different integer values (ID) are mapped to a *match_status* (the string explaining the context of the match):

```
<match_status_descriptions response_code="OK">

<match_status id="0" description="Not started"/>
<match_status id="1" description="1st period" period_number="1"/>
<match_status id="301" description="First break"/>
<match_status id="2" description="2nd period" period_number="2"/>
<match_status id="302" description="Second break"/>
<match_status id="80" description="Interrupted"/>
<match_status id="81" description="Suspended"/>
<match_status id="90" description="Abandoned"/>

</match_status_descriptions>
```

For a complete list of all available match statuses, use our Unified feed live documentation at: https://iodocs.betradar.com/unifiedfeed, at the endpoint labelled *descriptions/(lang)/match_status.xml*.

2.4.2. Clock element in sport_event_status

The sport_event_status may contain a clock element. This clock element includes various clock/time attributes that are sports specific. The following table lists these attributes.

Table 20 - clock element

Element Name	Description	Format/Example	Sports
`match_time`	The playing minute of the match (or minute:second if available)	`44 29:20`	Soccer
`stoppage_time`	How far into stoppage time is the match in minutes		Soccer
`stoppage_time_announced`	Set to what the announce stoppage time is		Soccer
`remaining_time`	How many minutes remains of the match	`mm:ss`	Ice Hockey, Basket, American Football
`remaining_time_in_period`	How much time remains in the current period	`mm:ss`	Ice Hockey, Basket,

			American Football
stopped	`true` if the match clock is stopped otherwise false		Ice Hockey, Basket, Futsal

XML example of the clock element

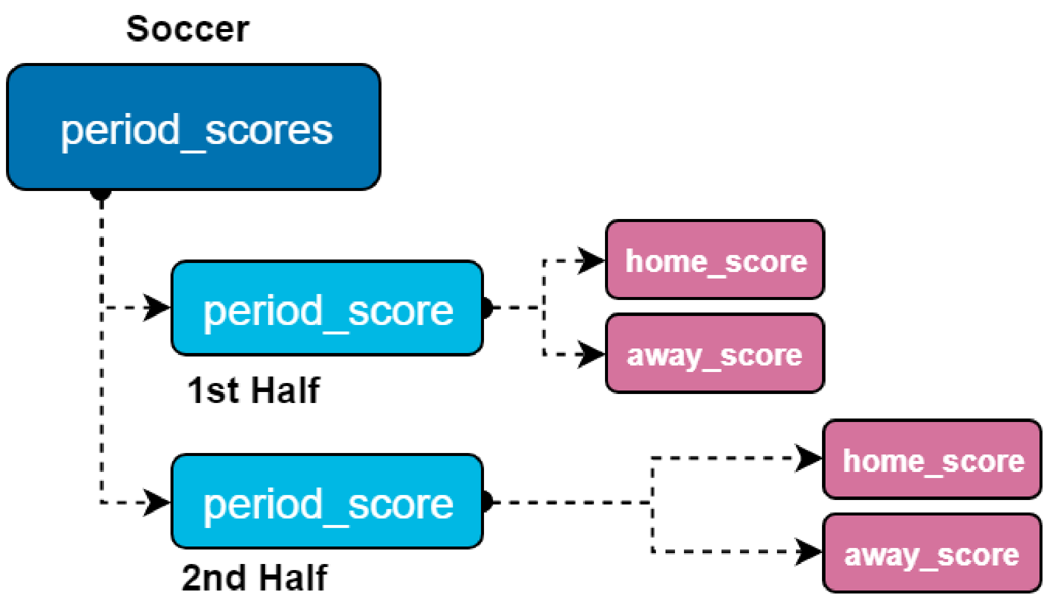
```
<sport_event_status _match_status_name="1st half" away_score="0"
away_suspend="0" home_score="6" home_suspend="0" match_status="6"
reporting="1" status="1">

        <clock match_time="3:56" remaining_time="56:04"
remaining_time_in_period="26:04" stopped="true"/>

        <period_scores>
            <period_score away_score="0" home_score="6"
match_status_code="6" number="1"/>
        </period_scores>
</sport_event_status>
```

2.4.3. Period scores in sport_event_status

In the *sport_event_status* element you can find the *period_scores* element that lists the individual period scores for a match. "Period" is a generic name for the sport-specific equivalent, so in soccer the 1st half, 2nd half, overtime and penalties are periods. In basketball the periods are the basketball quarters and in tennis it is the tennis sets, etc.

The period_scores element has a variable number of period_score sub-elements depending on how many periods this particular match had/currently has.

Figure 5 - Period_score illustration



Each period_score has the following attributes:

Table 21 - Attributes in the period_score element

Attribute	Description
`home_score`	The number of points/goals/games the competitor designated as “home” has scored for this period
`hway_score`	The number of points/goals/games the competitor designated as “away” has scored for this period
`number`	Present if the type is a regular period, and indicates what regular period this is
`match_status_code`	If present, provides a match_status_code that can be used to find a descriptive/display friendly name for this period (see match_status for more details).

Period_scores are available both in the Sports API where available, and in odds_change messages for live odds (games that are live).

Whether there is a period_scores attribute depends on the coverage level of the match, and if we don’t have period level coverage, no period_scores will be available.

```
<sport_event_status status="closed" home_score="2" away_score="2"
status_code="4" match_status_code="100">
    <period_scores>
        <period_score home_score="2" away_score="2" type="regular_period"
number="1"/>
        <period_score home_score="0" away_score="0" type="regular_period"
number="2"/>
    </period_scores>
</sport_event_status>
```

2.4.4. Statistics element in sport_event_status for soccer

While a soccer match is live and Betradar has live coverage of the match, there is a statistics element within the sport_event_status that list the most common statistics as counters for both home and away team:

```
<statistics>
  <yellow_cards home="1" away="0"/>
  <red_cards home="0" away="0"/>
  <yellow_red_cards home="0" away="0"/>
  <corners home="2" away="3"/>
</statistics>
```

Table 22 - statistics element

Element Name	Description
corners	Number of corners for the team
red_cards	Number of red cards for the team
yellow_cards	Number of yellow cards for the team
yellow_red_cards	Number of red cards as a result of a previous yellow card for the team

2.5. Bet stop hierarchy

Note that bets are stopped in a hierarchy. To see if bets are open on an outcome for a market and an event, the rules below all need to passed:

1. The Betradar system is available (messages have been received in the last 20 seconds).
2. The product handling the event is not flagged as down.
3. The market status is active (not suspended or deactivated).
4. The outcome is active (not active="false").

Conversely, if any of the above conditions become false, all corresponding bets should be stopped.

2.6. The minutes before an event starts

A few minutes before an event starts (in which Betradar offers Live Odds), Betradar has scouts and operators in place monitoring the game. At this time, the live markets are opened. Up until the match scheduled start (or actual start), Betradar provides many markets for both live odds and pre-match odds. Betradar, by default, distributes the "live" odds as the odds for the market. You also have the option to use the pre-match odds instead. When the match starts (status="live") a pre-match only system must act on this message and close the markets.

If Betradar does not offer live coverage for a match, Betradar systems will send out a *bet stop* that closes the markets for this sport event once the scheduled start time of the event is exceeded. In addition, for matches without live coverage, Betradar will not update the status field to live during the match, the status field will change from *not_started* to *closed* when the results are entered.

2.6.1. Special market statuses during handover

During the handover from the prematch producer to the Live Odds producer, the live producer will inform the prematch producer what markets it will provide, and then start sending odds for these markets to the client system. The prematch producer will send a final *odds_change message*. There are no odds updates in this message, only market status updates, the statuses will either be *deactivated* or *handed_over*. The deactivated markets should be deactivated by the client system, as these are markets that the live producer will no longer send. The markets that are marked as *handed_over* should be handled the following way:

- If you have already received odds for this market from the live odds producer, it should ignore this market update completely
- If it has not yet received odds for this market it should suspend that market

This last step is to ensure consistency on the client side in the unlikely event that the live odds producer fails after it has told the prematch producer to stop sending odds, but before it has been able to send its first odds update for this sport event. In such a case: The markets will be suspended as they should, and as soon as the Live Odds producer is available again it will start sending odds for this market.

2.7. Recovery – General information

Recovery is mostly done through the API. See the API or SDK section for more detailed information about recovery. In this chapter you will find some other notable cases to look out for while performing a recovery.

Note that the system has an *at-least-once-guarantee*. If something goes down, there is some risk that the same message is sent multiple times; your system should be built to handle this (for odds changes this is typically not a problem).

2.7.1. Handling special states

Forced betstop

In case of the following scenario:

- If the scheduled time for a match has started and the prematch odds producer provided the last odds.
- You have not received any odds from another producer, and no betstop from the prematch producer has been received.

In this case, the client system should automatically betstop all markets for the match. This is an extra precaution if the prematch system is down and you have not yet noticed this happening.

Handed_over market state during recovery

During recovery a producer may send a message saying that some markets have been handed_over. This is not a market state, but an indicator that since the disconnect happened this producer has handed over odds production for this market to another odds producer. If you already have received odds for this market from another odds producer, you can ignore this message, otherwise you should now mark this market as *suspended*. You should shortly receive odds from the other producer after it is handed over. For more information about the life cycle of market, see this chapter.

Settled and cancelled market state during recovery

During recovery some markets can be sent with the market state *settled*. This means that bet_settlements already have been sent for this market. This can for example happen during the 2nd half in soccer, where the 1st half markets have already been *settled*. There is no more activity expected for this market (the only way the system can change this is to send a rollback_bet_settlement). Similarly, during recovery a market can be sent with the market state *cancelled*, which should be treated in the same manner as above - no more activity is expected, and if you haven't already received a bet_cancel when recovery ends, a bet_cancel message has been missed somewhere.

General recommendations for market state handling when an Odds Producer is unavailable

If a match is live and the client system loses connection to the odds producer providing odds for that market, the client system must set all markets to *suspended* and stop accepting tickets.

If a match is not yet *live*, you have a choice: Either *suspend* all markets directly and stop accepting tickets. Alternatively, sport-events that are more than e.g. 2 hours from scheduled start time can be kept open as it is unlikely that there will be significant odds changes, and you are likely to get connected again shortly. **However**, we still recommend that if you haven't been able to reconnect to the odds producer in ~10 minutes, stop all markets for all sport_events for this odds producer as soon as possible.

2.8. AMQP topic filtering

Messages are sent on various topics that are intended to provide easy and flexible routing/filtering. The topic key is divided into 8 sections:

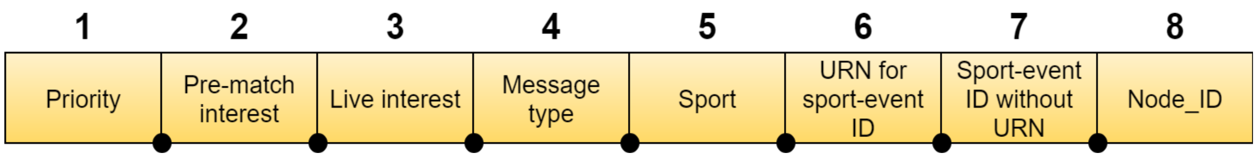


Figure 6 - AMQP topic filtering

1. **Priority (hi/lo/-)**: Is this a timing sensitive message (bet_settlement messages are considered not timing sensitive)?
2. **Pre-match interest (pre/virt/-)**: Is this message interesting for a pre-match only system? *For real sport-events prematch odds are sent with "pre" as the keyword. If this word is "virt", it means this is prematch odds for one of Betradar's virtual sports*.

3. **Live interest (live/-)**: Is this message interesting for live odds only system?
4. **Message type(s)**: odds_change, bet_settlement, bet_stop, bet_cancel, rollback_bet_settlement, rollback_bet_cancel, fixture_change, alive.
5. **Sport (Sport ID/-)**: 1 = *Soccer*, 2 = *Basketball*, 3 = *Baseball*, etc.
6. **URN for sport-event id**
7. **Sport-event id without URN**
8. **Node_id** *can be used for recovery to ensure that a session is not receiving recovery messages for another session for the same customer*

Table 23 - Typical topic keys

A pre-match odds update:	`hi.pre.-.odds_change.1.sr:match.1234.-`
A live odds update:	`hi.-.live.odds_change.1.sr:match.1234.-`
Odds update 5 min pre-game:	`hi.pre.live.odds_change.1.sr:match.1234.-`
A live bet-_settlement:	`lo.pre.live.bet_settlement.1.sr:match.1234.-`
Post-match bet-settlement	`lo.pre.-.bet_settlement.1.sr:match.1234.-`
A virtual sport odds update	`hi.virt.-.odds_change.1.vs:match.1234.-`
An alive message	`-.-.-.alive.-.-.-.-`
A snapshot complete message	`-.-.-.snapshot_complete.-.-.-`

Note: Unless you are binding to all messages ("#"), you will typically bind to at least two routing key patterns (e.g. "*.*.live.#" and "-.-.-.#") because you are typically always interested in receiving the system messages that will come with a routing key starting with -.-.-

Betradar strongly recommends to have all binding patterns always end with .# to ensure backward compatibility if additions are made to the routing key.

2.9. Message examples

The following snippets illustrates a few different message examples you might encounter in the unified feed:

***A** soccer goal*:

```
<odds_change event_id="sr:match:13656221" product="1"
timestamp="1525590430719">
    <sport_event_status _match_status_name="1st half" away_score="1"
home_score="0" match_status="6" reporting="1" status="1">
        <clock match_time="3:27"/>
        <period_scores>
            <period_score away_score="1" home_score="0"
match_status_code="6" number="1"/>
        </period_scores>
        <statistics>
            <yellow_cards away="0" home="0"/>
            <red_cards away="0" home="0"/>
            <yellow_red_cards away="0" home="0"/>
            <corners away="0" home="0"/>
        </statistics>
    </sport_event_status>
</odds_change>
```

***S**tart of match (if Betradar **has** live coverage of the event)*:

```
<odds_change event_id="sr:match:1234">
   <sport_event_status status="0" reporting="1"/>
</odds_change>   <--- scout has arrived pre-match (reporting="1")

<odds_change event_id="sr:match:12345">
  <sport_event_status status="1" reporting="1">
</odds_change>
```

***S**tart of match (if Betradar **does NOT** have live coverage of the event)*:

Right before scheduled start time:

```
<bet_stop event_id="sr:match:4711" group="all"/> <!-- <id> match started,
no live reporting -->
```

2.9.1. Use of rollback_bet_cancel when extending a temporary bet_cancel to a complete bet_cancel

Sometimes a bet_cancel is first sent out for a specified timeframe (i.e. from a start_time to an end_time), then later it is decided that the bet_cancel should be all bets not only for the specified timeframe. In such cases, the system will send out a rollback_betcancel with the start and endtime specified, and a new bet_cancel with no start and end_time.

```
<bet_cancel end_time="1535435606000" event_id="sr:match:15471064" prod-
uct="1" start_time="1535432789478" timestamp="1535435612134">
```

*Then some time later (**Note** the order of these two messages can be switched):*

```
<bet_cancel event_id="sr:match:15471064" timestamp="1535435612134"/>
<rollback_betcancel event_id="sr:match:15471064"
timestamp="1535435612134"/>
```

2.10.Special message cases

Bet settlements for another outcome for correct score markets

Some correct score markets have limited outcomes. If the actual correct score is something different than the outcomes that were offered, there is a choice:

- The bets placed can be considered lost
- The bets placed can be refunded (as the winning outcome wasn’t offered).

Betradar intends this to be a configuration option going forward. As it is currently: You will receive a bet_settlement first from our live-odds producer, where all outcomes are reported as losses, and a second bet_settlement from the prematch producer, where all outcomes are reported as voided (refunded).

3. Unified Odds - SDK

The Unified Odds SDK provides a simple and efficient way for bookmakers to access Betradar's odds and sport information. It combines subscription of messages and API calls into a unified Java or .NET interface that hides most of the complexity, including recovery.

Our SDK has a dedicated documentation section both for the Java and .NET versions. For more detailed information about using and getting started with the SDK, please visit our dedication SDK sections:

Java: http://sdk.sportradar.com/content/unifiedfeedsdk/java2/javadoc/

.NET: http://sdk.sportradar.com/content/unifiedfeedsdk/net/doc/html/15adaea0-5ed9-0831-79a5-f2702175b2d9.htm

3.1. SDK benefits over protocol

- The SDK hides the separation between messages and API-lookups. The client system just receives the message objects where all information is prefilled by the SDK caches, and in some cases looked up in the background when the client system requests some more rarely requested information.
- The SDK takes care of translations transparently. The client system just needs to define what languages it needs.
- The SDK takes care of dynamic text markets and outright markets automatically, which requires some extra logic and lookups for someone not using the SDK.
- The SDK handles initial connect and state, as well as recovery in case of a temporary disconnect. This needs to be handled manually by someone not using the SDK.
- The SDK provides an up to date cache of each sport-events current status that is updated automatically.

3.2. Implementer’s notes

Remember that your OddsFeedListener call-backs are handled by a thread; if you do too much processing within your call-back, subsequent odds messages are delayed as they will not be processed until you finish the call-back processing.

The recommended way to handle longer processing times is to put the received messages on a queue, and let them be processed by a different thread.

The first time you try to access information about a sport event, the SDK looks up its details using an API-call. This information is cached for subsequent calls. For that reason, the first call for a new sport event may take a bit longer. It is also during this first time call that the various languages you have requested are looked up for the event – so if you have specified multiple languages the first call may take even longer.

If you can avoid accessing any non-basic properties, you can avoid longer delays most of the time, as the basic information is normally cached for almost all messages by the SDK. (Basic information is sport, category, tournament, match competitors, scheduled start time).

4. Unified Odds - API

This section of the documentation will highlight the most important information when working with the unified odds API.

4.1. Self-service documentation

All endpoints are available for testing purposes at http://iodocs.betradar.com. From there you can see each of the endpoints, execute the requests, and see sample responses as long as you have an access-token (see below).

4.2. Authentication

To access the API you must visit Betradar.com and generate an application key associated with your Betradar.com user details. This API key has to be sent as an HTTP header on every request towards the service on the following format:

```
x-access-token: <access token>
```

4.3. Endpoints

The following table lists all available endpoints found on: https://iodocs.betradar.com/unifiedfeed

Table 24 - API endpoints

HTTP	Endpoint-path from https://api.betradar.com/v1/	Description
GET	`descriptions/(lang)/markets.xml`	Describes all currently available markets.
GET	`descriptions/(lang)/match_status.xml`	Describes all sports specific match status codes used during live matches in the odds_changes message. Translated to available languages.
GET	`descriptions/betstop-reasons.xml`	Describes all bet stop reasons.
GET	`descriptions/betting_status.xml`	Describes all betting statuses (in odds_changes.odds)
GET	`descriptions/producers.xml`	Describes all currently available producers and their ids.
GET	`descriptions/void_reasons.xml`	Describes all possible void reasons (in bet_settlement.market)
POST	`liveodds/booking-calendar/events/(id)/book`	POST on this endpoint to book a match in the booking-calendar

GET	`probabilities/(id)`	Retrieve the probabilities for all available markets for a sport-event. Typically used for cashout purposes.
GET	`probabilities/(id)/(market-id)`	Retrieve all the probabilities for all the market lines for the specified sport-event and market-id. Typically used for cashout purposes.
GET	`probabilities/(id)/(market-id)/(specifiers)`	Retrieve the probabilities for the specified market line.
POST	`(product)/stateful_messages/events/(id)/initiate_request`	Used request all stateful messages (bet_settlement, bet_cancel, etc.) sent for a particular sport event.
POST	`(product)/recovery/initate_request.xml?after=(timestamp)[&request_id=(x)][&node_id=(y)]`	Sends all current odds and historical stateful messages (bet_settlement, rollback_bet_settlement, bet_cancel, rollback_betcancell) from the specified timestamp and onwards. An error is returned if the specified timestamp is too far in the past.
POST	`(product)/odds/events/(id)/initiate_request`	All current odds for all the markets for one single match/race. The event must be either not_started or live.
GET	`sports/(lang)/competitors/(id)/profile.xml`	Name and details about a competitor (team, race driver, tennis player etc).
GET	`sports/(lang)/fixtures/changes.xml`	Lists the ids of sport-events that have fixture changes the last 24 hours.
GET	`sports/(lang)/sport_events/(id)/fixture.xml`	Lists the fixture for a specified event (event = match, game, race, outright).
GET	`sports/(lang)/sport_events/(id)/timeline.xml`	Information about a sport event that is ongoing or completed. The actual endpoint is event-specific and the actual attributes varies slightly between sports.
GET	`sports/(lang)/sport_events/(id)/summary.xml`	Returns the results for a sport event. The <id> must use an urn-scheme. sr:match:<id> for match-ids. sr:stage:<id> for race results. sr:tournament:<id> for tournament results, etc.
GET	`sports/(lang)/players/(id)/profile.xml`	Name and details about a player. A player here is a player in a team. (A tennis player is a competitor)
GET	`sports/(lang)/schedules/(date)/schedule.xml`	Lists all sport events scheduled to start at a specified date (UTC). <date> is an ISODate (e.g. 2016-02-10).
GET	`sports/(lang)/schedules/live/schedule.xml`	Lists all currently live sport events (id, what teams, start time, etc.).

GET	`sports/(lang)/schedules/pre/schedule.xml?start=(x)&limit=(y)`	Lists almost all events we are offering prematch odds for. This endpoint can be used during early start-up to obtain almost all fixtures. This endpoint is one of the few that uses pagination.
GET	`sports/(lang)/sports.xml`	A list of all available sports
GET	`sports/(lang)/sports/(id)/categories.xml`	This lists available categories for a sport. Category is generic classification term BetRadar uses to at the highest level subclassify a particular sport (e.g. for Tennis the categories can be ATP-Tour, WTA-Tour, David Cup etc., for soccer the categories are the various countries)
GET	`sports/(lang)/sports/(id)/tournaments.xml`	A list of all available tournaments for the specified sport
GET	`sports/(lang)/tournaments.xml`	A list of all available tournaments
GET	`sports/(lang)/tournaments/(id)/info.xml`	Provides basic information about a tournament (such as the dates of the tournament, type of tournament and competitors in the tournament)
GET	`sports/(lang)/tournaments/(id)/schedule.xml`	Provides all scheduled matches in the specified tournament.
GET	`sports/(lang)/venues/(id)/profile.xml`	Details about a venue.
GET	`users/whoami.xml`	Lists the callers bookmaker id, AMQP access point and when the provided access token will expire.

- (lang) Is the requested language (note that the XML elements and attributes themselves are not translated, only some of the attribute values; e.g. team names, player names etc.).
- (producer) Can be live odds (Live Odds) or pre (normal prematch odds), BetPal, premium_cricket, vfl (virtual football), vto (virtual tennis), vbl (virtual basketball), wns (number betting). Additional producers will be added over time: (See the endpoint: https://api.betradar.com/v1/descriptions/producers.xml for a listing of currently available producers and their ids).

Note: When using the *sports/(lang)/schedules/pre/schedule.xml?start=(x)&limit=(y)* endpoint in the table above, keep in mind that this is intended for a mostly stateless start-up of the client.

Start: Is the starting record (this is an index, not time).

Limit: Is how many records (sport_events) to return. Max for limit is 1000, so you can do; *start = 0&limit=1000*, then *start=1000&limit=1000* and so on, until you get no data.

The primary goal of this endpoint is to reduce the number of individual fixtures you have to read during a recovery.

4.3.1. Response codes

The general http status codes are defined here https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

Here is how Status Codes are used in this API:

Table 25 - HTTP response codes

Code	Name	Description
`200`	Ok	When everything is ok and we return data directly in the http body
`201`	Created	For booking calendar when client successfully books a match
`202`	Accepted	The system accepted your request for odds recovery/stateful messages and will soon/immediately send it out on the feed
`403`	Forbidden	Token is missing or invalid.
`404`	Not found	Resource (event id or sport id etc) was not found, the system will not fulfill your request.
`409`	Conflict	Bookmaker has another odds recovery already in progress
`503`	Service unavailable	Returned when the underlying odds producers are temporarily down. Retry again soon.

4.3.2. Special note on the tournaments.xml endpoint

This endpoint lists all available tournaments where Sportradar provides coverage.

You can still receive information about a match that is associated with a tournament not on the returned list from the endpoints. This could for example happen with lower division soccer league matches where Sportradar does not provide full league coverage, but may do so only upon request from one or more bookmakers. Such tournaments are typically not listed, and Sportradar does not provide a tournament schedule either for these tournaments.

Therefor it is important to not assume that these endpoints will return every single tournament of interest, but rather all available tournaments where Sportradar provide coverage and/or customers buy coverage for a particular match where Sportradar has plans for full coverage.

4.4. Fixtures - API

Fixtures describe static or semi-static information about matches and races. There are multiple endpoints for fixtures, all found at the fixture.xml or schedule.xml endpoints:

Table 26 - API for fixtures

HTTP	Endpoint	Description
GET	`sports/(lang)/fixtures/changes.xml`	Lists the ids of sport-events where changes were made the last 24 hours.
GET	`sports/(lang)/sport_events/(id)/fixture.xml`	Lists the fixtures for a specified event (event = match, game, race, outright).
GET	`sports/(lang)/schedules/(date)/schedule.xml`	Lists all events for a specific day.
GET	`sports/(lang)/schedules/live/schedule.xml`	Lists all currently live events.
GET	`sports/(lang)/scheduleds/pre/schedule.xml?start=(x)&limit=(y)`	Lists almost all events we are offering prematch odds for. This endpoint can be used during early startup to obtain almost all fixtures. This endpoint is one of the few that uses pagination.
GET	`sports/(lang)/fixtures/changes.xml`	Lists all fixtures that have changed during the last 24 hours. This can be used to only update the sport events that were updated since the last time you checked (to avoid polling and reprocessing the complete list).
GET	`sports/(lang)/tournaments/(id)/schedule.xml`	Lists all events for a particular season of a league/competition where we have coverage.

4.4.1. Attributes

All relevant elements and attributes found inside the *fixture.xml* endpoint in the API.

Table 27 - Fixture elements and attributes

Name	Description
`id`	Unique identifier of this event (match, game, race, outright).
`start_time`	When the specific match will start.

`start_time_confirmed`	If true, the start time is scheduled and confirmed (the event can still become delayed) otherwise, it could be a best guess.
`start_time_tbd`	Possible values: True or false This is only set if the starting time or date is not confirmed. If we know the date but not the starting time, this attribute is set to “True”.
`next_live_time`	When live odds should be offered next time (i.e. start of match, or when a match will start again after suspension). At this point you will typically receive an Odds Change message with <sport_event_status reporting="1">.
`liveodds`	Indicates whether this match is booked by you for livecoverage (“booked”) bookable (“bookable”), is buyable (“buyable”) or not available (“not_available”).
`status`	Indicates whether this match is not started yet, live, or finished ("not_started", "live", "ended", "closed", “cancelled", “postponed”, “suspended”, “interrupted”, “delayed”). When it is ended, the results may not have been fully confirmed yet. When it is closed Betradar has confirmed the results.
`competitors`	A listing of the competitors in the event. For team-based matches the competitors are two teams. For a singles match in Tennis it is two tennis players etc. Get additional info about the competitor in a separate request to /sports/<lang>/competitors/<id>/profile.xml
`venue`	Where the event is taking place. Also see /sports/<lang>/venues/<id>/summary.xml
`venue.map_coordinates`	The GPS coordinates of the venue (comma separated)
`tv_channels`	A list of TV channels.
`delayed_info`	Time and reason for delays to this event.
`extra_info`	Within this element, sport-specific key-value pairs are added to provide sport-specific details about the event.
`coverage_info`	Information about how much information Betradar has, or will collect, for this particular sport event.
`product_info`	In this element, all data that is Betradar specific is kept.
`product_info.streaming`	Details about Betradar streaming offering.
`product_info.info`	A set of key-value elements describing various Betradar properties (see table below).
`product_info.links`	Within this element Betradar has links to various pages within Betradar’s hosted solution offering for this particular event.
`reference_id`	Additional ids for competitors. The reference_id name can be “betradar” and the id is then the primary id used in the LCOO Feeds. If the name is “betfair” the id is the id used in Betfair. If the name is BetradarCtrl the id is the id as seen in BetradarCtrl

`tournament_round.betradar_id`	Id used in the previously existing LCoO Feeds for this tournament. (Note that different tournament rounds could have different LCoO Feed Ids)
`category.country_code`	If category is a country type category there will be a country code which is the three-letter ISO country-code wherever possible. If there is no ISO country-code available, we will provide another code and document this.

XML Fixture Example:

```
<fixtures_fixture>
<fixture id="sr:match:8696826" scheduled="2016-10-31T18:00:00+00:00"
start_time_tbd="false" next_live_time="2016-10-31T18:00:00+00:00"
start_time="2016-10-31T18:00:00+00:00" start_time_confirmed="true">
    <tournament_round betradar_id="4301" type="group" number="25"/>
    <season id="sr:season:12346" name="Div 1, Sodra 2016" start_date="2016-
04-16" end_date="2016-11-05" year="2016" tournament_id="sr:tournament:68"/>
    <tournament id="sr:tournament:68" name="Div 1 Sodra">
        <sport id="sr:sport:1" name="Soccer"/>
        <category id="sr:category:9" name="Sweden" country_code="SWE"/>
    </tournament>
    <competitors>
        <competitor id="sr:competitor:1860" name="IK Oddevold"
country="Sweden" country_code="SWE" abbreviation="ODD" qualifier="home">
            <reference_ids>
                <reference_id name="betradar" value="1449434"/>
            </reference_ids>
        </competitor>
        <competitor id="sr:competitor:22356" name="Tvaakers IF"
country="Sweden" country_code="SWE" abbreviation="TVA" qualifier="away">
            <reference_ids>
                <reference_id name="betradar" value="7281378"/>
            </reference_ids>
        </competitor>
    </competitors>
    <extra_info>
        <info key="neutral_ground" value="false"/>
        <info key="period_length" value="45"/>
        <info key="overtime_length" value="15"/>
    </extra_info>
    <coverage_info level="bronze" live_coverage="false">
        <coverage includes="basic_score"/>
    </coverage_info>
    <product_info>
        <is_in_hosted_statistics/>
    </product_info>
    <reference_ids>
        <reference_id name="BetradarCtrl" value="11428313"/>
    </reference_ids>
</fixture>
</fixtures_fixture>
```

4.4.2. Extra info values

Information and values found inside of the extra_info attribute in *fixtures.xml* in the API.

Table 28 - Extra info for fixtures

Name	Type	Description
neutral_ground	boolean	Set to true if one of the competitors is playing on their home venue, otherwise set to false.
auto_traded	boolean	Set to true if the match is auto-traded. For auto-traded matches odds are solely generated by a mathematical odds model and live scoring. Any market monitoring and trader interaction is not considered.
draw_possible	boolean	Set if a draw is possible
no_advantage_scoring	boolean	Only present and set to true if no advantage scoring is used. In this tennis rule mode, the first player to get 4 points wins the game.
best_of	integer	The number of sets (or equivalent) that the match maximum will have. Typically the winner needs best_of/2 + 1 sets to win.
set_limit	integer	Number of points in a set (max).
period_length	integer	Number of minutes in a period.
overtime_length	integer	Number of minutes for each overtime period.
super_tie_break	integer	Number of points required to win match deciding tiebreak.
coverage_source	enumeration (venue, tv)	How are the live results obtained (directly from venue, from TV).
extended_live_markets_offered	boolean	What live markets will be offered; either the normal ones or an extended offering, depending on coverage type (basic or deeper coverage).
surface	enumeration	**Possible values**: hard_court, grass, sand, red_clay, green_clay, hardcourt_outdoor, carpet_indoor, synthetic_indoor,

		synthetic_outdoor, synthetic_grass, hardcourt_indoor, red clay indoor.
best_of_legs	integer	Number of legs. In darts a single game is called a leg. You win a match by winning a number of legs. In some competitions a match consists of sets where each set consists of legs (in such cases both best_of and best_of_legs are typically available).

```
    <extra_info>
        <info key="neutral_ground" value="false"/>
        <info key="period_length" value="45"/>
        <info key="overtime_length" value="15"/>
        <info key="best_of" value="3"/>
    </extra_info>
```

4.4.3. Reference ids

Displays what producer reference this fixture is using, displayed inside the *reference_id* element.

Table 29 - Fixture references

Name	Description
betradar	The ID used in the Betradar LCoO Feed
betradarctrl	The ID used in BetradarCtrl
betfair	The ID used in Betfair
rotation_number	If the reference id name is a rotation_number, the id is a rotation number. See the concept section for details on rotation numbers.
aams	AAMS (Amministrazione Autonoma dei Monopoli di Stato) – Italian betting regulator.

```
    <reference_ids>
        <reference_id name="BetradarCtrl" value="11428313"/>
    </reference_ids>
```

4.4.4. Betradar product info

Additional information about the fixture found inside the *product_info* attribute in *fixtures.xml*.

Table 30 - Fixture product information

Name	Description
is_in_live_score	If Betradar has Live Score for this event, this key is present and set to true.

`is_in_hosted_statistics`	If Betradar has hosted statistics for this event, this key is present.
`is_in_live_center_soccer`	If this event is a Soccer event and Betradar offers it within the Live Sports Centre Soccer, this key is present and set to true.
`is_auto_traded`	If this event is **not** covered by Live Odds operations, this key is present and set to true.

```
    <product_info>
        <is_in_hosted_statistics/>
    </product_info>
```

4.4.5. Competitors

Competitors in the fixture have an id attribute and a name attribute. Competitor IDs that are on the format “sr:competitor:XXX” can typically be looked up individually in the API (competitors/(id)/profile.xml), and may have translations to multiple languages.

Competitors with an id of type “sr:simpleteam:xxx” do not have translations, and are not unique. These are teams that Betradar have created for a one-off match that we cover, that are not part of some tournament/league Betradar normally covers, and for these teams we do not provide translations. The name that is in the competitor element can preferably be used to name the team. If the same team competes in another tournament (sr:simpleteam:xxx), the team will be using a different id. Betradar tries to keep the number of competitors of this type to a minimum.

```
<competitors>
      <sport id="sr:sport:1" name="Soccer"/>
      <category id="sr:category:17" name="Tournaments"/>
      <competitor id="sr:competitor:12345" name="Team1"/>
      <competitor id="sr:competitor:11123" name="Team2"/>
</competitors>
```

4.4.6. Sport event context

Matches and races happen within some context. They are of some sport and some Betradar category. They are part of some league, competition, tournament or season. They may be part of a particular round in a group-stage, or a play-off stage. They may be part of a cup at some level of the cup-tree. It may be a practice or qualification race etc.

Our Sports API offers functionality to understand and display this context information in the client system.

```
<sport_event id="sr:match:11830662" scheduled="2017-11-19T16:00:00+00:00"
start_time_tbd="true">
<tournament_round type="group" number="12"/>
  <season id="sr:season:40942" name="Premier League 17/18"
start_date="2017-08-11" end_date="2018-05-14" year="17/18"
tournament_id="sr:tournament:17"/>
   <tournament id="sr:tournament:17" name="Premier League">
   <sport id="sr:sport:1" name="Soccer"/>
   <category id="sr:category:1" name="England" country_code="ENG"/>
</tournament>
```

Sport

Betradar has a set of defined Sports that is used as the highest level of categorization.

Category

The second highest level of categorization is called *Category* in Betradar, this is Betradar's own classification system. The categories available are different for different sports. For example, for soccer the categories are typically countries, but for tennis the categories are the different tours: ATP, WTA, ITF, and so on.

Season

A match typically happens within a season of a recurring tournament/competition (e.g. Bundesliga 2016/17 or Wimbledon, Men Singles, 2017).

Tournament

Tournament is the recurring competition. This is best explained by example: "Bundesliga" is the **tournament**. "Bundesliga 16/17" is a season of "Bundesliga". The tournament can be used to get easily get the information for most current season of the tournament, or list available seasons of the tournament.

Groups

If a competition/league has multiple groups, the different groups are listed in the tournament/(season-id)/info.xml endpoint, which also lists what teams are included in what group. In cases where a season has a qualification stage and a group stage (and maybe a playoff-stage), there will be an unnamed group that lists all competitors that were part of the qualification stage. Only named groups should typically be displayed when listing groups on the client side. The unnamed qualification group will typically list some of the same competitors that are also found in the named groups as well as some competitors that are not at all found in the named groups (the teams that did not manage to qualify).

Round

In a match to describe what cup stage or round the match is, part the API offers the concept of a round.

Tournament round has three types: *group*, *cup* or *qualification*. If the type is "group", this is a group/pool/table match, and there is an additional attribute that displays the round number. If the type is *cup* or *qualification*, there is an additional attribute *cup_level* which describes the “level” of the cup. It can be any string but there some standard values = "round_of_128" |"round_of_64" | "round_of_32" | "round_of_16" | "quarterfinal" | "semifinal" | "final" | ”3rd_place_final”.

4.4.7 Getting prematch fixtures at start-up

There is a special endpoint to get almost all fixtures before initiating recovery. This endpoint is designed to significantly reduce the number of API calls required during recovery. The endpoint is also special in that it is one of the few endpoints that uses pagination (i.e. you have to specify a start record and a limit for the number of records to retrieve). The maximum number of records you can retrieve in one call is 1000. The records returned are all sport_events just like the records returned in the daily schedule endpoint. This can be done using the */schedule.xml* and */tournament.xml* endpoints, and specifying a language ID and date. An example of a sport_event message included `scheduled="2016-03-31`:

From */schedule.xml:*

```xml
<sport_event id="sr:match:9199415" scheduled="2016-03-31T09:45:00+00:00"
start_time_tbd="false" status="closed" next_live_time="2016-03-
31T09:45:00+00:00">
    <tournament_round type="cup" name="round_of_16"/>
    <season id="sr:season:12801" name="India Open (SS) 2016"
start_date="2016-03-28" end_date="2016-04-04" year="2016"
tournament_id="sr:tournament:1277"/>
    <tournament id="sr:tournament:1277" name="India Open (SS)">
        <sport id="sr:sport:31" name="Badminton"/>
        <category id="sr:category:259" name="International"/>
    </tournament>
    <competitors>
        <competitor id="sr:competitor:87528" name="Matsutomo M / Takahashi
A" abbreviation="M/T" qualifier="home">
            <reference_ids>
                <reference_id name="betradar" value="5261571"/>
            </reference_ids>
        </competitor>
        <competitor id="sr:competitor:101271" name="Stoeva G / Stoeva S"
abbreviation="S/S" qualifier="away">
            <reference_ids>
                <reference_id name="betradar" value="6170039"/>
            </reference_ids>
        </competitor>
    </competitors>
</sport_event>
```

From */tournament.xml:*

```
<tournaments>
    <tournament id="sr:tournament:109" name="MLB">
        <sport id="sr:sport:3" name="Baseball"/>
        <category id="sr:category:16" name="USA" country_code="USA"/>
    </tournament>
    <tournament id="sr:tournament:1046" name="Italian Baseball League">
        <sport id="sr:sport:3" name="Baseball"/>
        <category id="sr:category:510" name="Italy" country_code="ITA"/>
    </tournament>
    <tournament id="sr:tournament:2456" name="MLB Spring Training">
        <sport id="sr:sport:3" name="Baseball"/>
        <category id="sr:category:16" name="USA" country_code="USA"/>
    </tournament>
    <tournament id="sr:tournament:2541" name="KBO League ">
        <sport id="sr:sport:3" name="Baseball"/>
        <category id="sr:category:384" name="Republic of Korea"
country_code="KOR"/>
    </tournament>
    <tournament id="sr:tournament:14602" name="ABL">
        <sport id="sr:sport:3" name="Baseball"/>
        <category id="sr:category:775" name="Australia"
country_code="AUS"/>
    </tournament>
    <tournament id="sr:tournament:24087" name="MILB">
        <sport id="sr:sport:3" name="Baseball"/>
        <category id="sr:category:16" name="USA" country_code="USA"/>
    </tournament>
</tournaments>
```

4.4.6. Fixture change history

We provide a date and fixture change history for all matches, available in the *fixtures/changes.xml* endpoint in the self-service documentation HERE.

```
<fixture>
<scheduled_start_time_changes>
<scheduled_start_time_change old_time="2018-03-23T11:00:00+00:00"
new_time="2018-09-07T10:00:00+00:00" changed_at="2018-03-
14T00:07:23+00:00"/>
</scheduled_start_time_changes>
</fixture>
```

4.5. Sport event live and resulting information

This section details information about our “Live Information” (when a match is live), and “Resulting information” (when the match is finished). These can be accessed with the */timeline.xml* and */summary.xml* endpoints.

4.5.1. Live information

Table 31 - Live information endpoints

HTTP	Endpoint	Description
GET	`sports/(lang)/sport_events/(id)/summary.xml`	Information summary about a particular event (pre-match, live or post-match). Pre-match information is very brief. Post-match includes the results.
GET	`sports/(lang)/sport_events/(id)/timeline.xml`	Information about the timeline of a particular event (pre-match, live or post-match). Pre-match information is very brief. Post-match includes the results.

/timeline.xml example:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<match_timeline xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
generated_at="2018-10-30T11:15:06+00:00"
xmlns="http://schemas.sportradar.com/sportsapi/v1/unified"
xsi:schemaLocation="http://schemas.sportradar.com/sportsapi/v1/unified
http://schemas.sportradar.com/bsa/unified/v1/xml/endpoints/unified/match_ti
meline.xsd">
<sport_event id="sr:match:14704749" scheduled="2018-08-20T16:30:00+00:00"
start_time_tbd="false">
    <tournament_round type="cup" name="cupround"/>
    <season id="sr:season:54377" name="DFB Pokal 18/19" start_date="2018-
08-17" end_date="2019-05-26" year="18/19"
tournament_id="sr:tournament:217"/>
    <tournament id="sr:tournament:217" name="DFB Pokal">
        <sport id="sr:sport:1" name="Soccer"/>
        <category id="sr:category:30" name="Germany" country_code="DEU"/>
    </tournament>
    <competitors>
        <competitor id="sr:competitor:2532" name="Energie Cottbus"
country="Germany" country_code="DEU" abbreviation="COT" qualifier="home">
            <reference_ids>
                <reference_id name="betradar" value="11135"/>
            </reference_ids>
        </competitor>
        <competitor id="sr:competitor:2538" name="SC Freiburg"
country="Germany" country_code="DEU" abbreviation="SCF" qualifier="away">
            <reference_ids>
                <reference_id name="betradar" value="11095"/>
            </reference_ids>
        </competitor>
    </competitors>
    <venue id="sr:venue:13" name="Stadion der Freundschaft"
capacity="22528" city_name="Cottbus" country_name="Germany"
map_coordinates="51.751263,14.344669" country_code="DEU"/>
```

```
</sport_event>
<sport_event_conditions>
    <referee id="sr:referee:70951" name="Osmers, Harm"
nationality="Germany"/>
    <venue id="sr:venue:13" name="Stadion der Freundschaft"
capacity="22528" city_name="Cottbus" country_name="Germany"
map_coordinates="51.751263,14.344669" country_code="DEU"/>
    <weather_info pitch="good" weather_conditions="medium"/>
</sport_event_conditions>
<sport_event_status status="closed" match_status="ap" home_score="5"
away_score="7" winner_id="sr:competitor:2538" status_code="4"
match_status_code="120">
    <period_scores>
        <period_score home_score="0" away_score="0" type="regular_period"
number="1" match_status_code="6"/>
        <period_score home_score="1" away_score="1" type="regular_period"
number="2" match_status_code="7"/>
        <period_score home_score="1" away_score="1" type="overtime"
match_status_code="40"/>
        <period_score home_score="3" away_score="5" type="penalties"
match_status_code="50"/>
    </period_scores>
</sport_event_status>
<coverage_info level="gold" live_coverage="true" covered_from="tv">
    <coverage includes="basic_score"/>
    <coverage includes="key_events"/>
    <coverage includes="detailed_events"/>
    <coverage includes="lineups"/>
    <coverage includes="commentary"/>
</coverage_info>
<timeline>
    <event id="451022004" type="match_started" time="2018-08-
20T16:31:40+00:00"/>
    <event id="451022002" type="period_start" time="2018-08-
20T16:31:40+00:00" period="1" period_name="1st half"
match_status_code="6"/>
    <event id="451022864" type="corner_kick" time="2018-08-
20T16:35:19+00:00" match_time="4" match_clock="3:38" team="home" x="95"
y="5"/>
    <event id="451023900" type="corner_kick" time="2018-08-
20T16:39:52+00:00" match_time="9" match_clock="8:12" team="away" x="5"
y="90">
        <player id="sr:player:319861" name="Waldschmidt, Luca"/>
    </event>
    <event id="451026456" type="corner_kick" time="2018-08-
20T16:51:02+00:00" match_time="20" match_clock="19:21" team="away" x="5"
y="90">
        <player id="sr:player:319861" name="Waldschmidt, Luca"/>
    </event>
```

```
<event id="451030590" type="yellow_card" time="2018-08-20T17:09:36+00:00"
match_time="38" match_clock="37:55" team="away">
        <player id="sr:player:319861" name="Waldschmidt, Luca"/>
    </event>
    <event id="451031914" type="yellow_card" time="2018-08-
20T17:14:03+00:00" match_time="42" team="home">
        <player id="sr:player:169183" name="Startsev, Andrej"/>
    </event>
    <event id="451032760" type="period_score" time="2018-08-
20T17:16:43+00:00" period="1" home_score="0" away_score="0"
match_status_code="6"/>
    <event id="451032764" type="break_start" time="2018-08-
20T17:16:43+00:00" period_name="Pause" match_status_code="31"/>
    <event id="451036854" type="period_start" time="2018-08-
20T17:34:49+00:00" period="2" period_name="2nd half"
match_status_code="7"/>
    <event id="451037196" type="score_change" time="2018-08-
20T17:36:16+00:00" match_time="47" match_clock="46:27" team="home" x="78"
y="53" home_score="1" away_score="0">
        <goal_scorer id="sr:player:770433" name="Freitas, Marcelo"/>
        <assist id="sr:player:132155" type="primary" name="Zimmer,
Maximilian"/>
    </event>
    <event id="451037692" type="corner_kick" time="2018-08-
20T17:38:29+00:00" match_time="49" match_clock="48:40" team="away" x="5"
y="90">
        <player id="sr:player:319861" name="Waldschmidt, Luca"/>
    </event>
    <event id="451040540" type="corner_kick" time="2018-08-
20T17:50:52+00:00" match_time="62" match_clock="61:03" team="away" x="5"
y="90">
        <player id="sr:player:319861" name="Waldschmidt, Luca"/>
    </event>
    <event id="451041302" type="corner_kick" time="2018-08-
20T17:54:32+00:00" match_time="65" match_clock="64:44" team="away" x="5"
y="90">
        <player id="sr:player:142231" name="Gunter, Christian"/>
    </event>
    <event id="451044014" type="corner_kick" time="2018-08-
20T18:07:34+00:00" match_time="78" match_clock="77:44" team="away" x="5"
y="90">
        <player id="sr:player:38502" name="Gondorf, Jerome"/>
    </event>
    <event id="451044596" type="yellow_card" time="2018-08-
20T18:09:56+00:00" match_time="80" team="home">
        <player id="sr:player:132589" name="Scheidhauer, Kevin"/>
    </event>
    <event id="451046632" type="yellow_card" time="2018-08-
20T18:18:05+00:00" match_time="89" match_clock="88:15" team="away">
        <player id="sr:player:49639" name="Hofler, Nicolas"/>
    </event>
    <event id="451047070" type="score_change" time="2018-08-
20T18:20:19+00:00" match_time="89" team="away" x="19" y="55" home_score="1"
away_score="1">
        <goal_scorer id="sr:player:22566" name="Frantz, Mike"/>
        <assist id="sr:player:228981" type="primary" name="Kleindienst,
Tim"/>
    </event>
```

```
<event id="451047800" type="corner_kick" time="2018-08-20T18:24:04+00:00"
match_time="90" match_clock="90:00" stoppage_time="5" team="away" x="5"
y="90"/>
    <event id="451048024" type="corner_kick" time="2018-08-
20T18:25:15+00:00" match_time="90" match_clock="90:00" stoppage_time="6"
team="home" x="95" y="90"/>
    <event id="451048066" type="break_start" time="2018-08-
20T18:25:27+00:00" period_name="Awaiting extra" match_status_code="32"/>
    <event id="451049228" type="period_start" time="2018-08-
20T18:31:00+00:00" period="11" period_name="1st extra"
match_status_code="41"/>
    <event id="451050964" type="score_change" time="2018-08-
20T18:39:29+00:00" match_time="98" team="away" x="10" y="47" home_score="1"
away_score="2">
        <goal_scorer id="sr:player:20826" name="Petersen, Nils"/>
    </event>
    <event id="451051512" type="yellow_card" time="2018-08-
20T18:41:56+00:00" match_time="101" match_clock="100:56" team="away">
        <player id="sr:player:20826" name="Petersen, Nils"/>
    </event>
    <event id="451052010" type="score_change" time="2018-08-
20T18:44:12+00:00" match_time="103" team="home" x="87" y="48"
home_score="2" away_score="2">
        <goal_scorer id="sr:player:142768" name="Viteritti, Fabio"/>
    </event>
    <event id="451052626" type="break_start" time="2018-08-
20T18:47:01+00:00" period_name="Extra time halftime"
match_status_code="33"/>
    <event id="451053180" type="period_start" time="2018-08-
20T18:49:37+00:00" period="12" period_name="2nd extra"
match_status_code="42"/>
    <event id="451054398" type="corner_kick" time="2018-08-
20T18:56:38+00:00" match_time="113" match_clock="112:00" team="home" x="95"
y="90"/>
    <event id="451054490" type="corner_kick" time="2018-08-
20T18:57:03+00:00" match_time="113" match_clock="112:25" team="home" x="95"
y="90"/>
    <event id="451054740" type="yellow_card" time="2018-08-
20T18:59:05+00:00" match_time="114" team="away">
        <player id="sr:player:280009" name="Holer, Lucas"/>
    </event>
    <event id="451055870" type="break_start" time="2018-08-
20T19:06:01+00:00" period_name="Awaiting penalties"
match_status_code="34"/>
    <event id="451056680" type="period_start" time="2018-08-
20T19:11:00+00:00" period="20" period_name="Penalties"
match_status_code="50"/>
    <event id="451056766" type="score_change" time="2018-08-
20T19:11:27+00:00" team="away" x="10" y="48" home_score="2"
away_score="3"/>
    <event id="451056896" type="score_change" time="2018-08-
20T19:12:05+00:00" team="home" x="88" y="48" home_score="3"
away_score="3"/>
    <event id="451057018" type="score_change" time="2018-08-
20T19:12:40+00:00" team="away" x="10" y="48" home_score="3"
away_score="4"/>
    <event id="451057128" type="score_change" time="2018-08-
20T19:13:19+00:00" team="home" x="88" y="48" home_score="4"
away_score="4"/>
```

```
    <event id="451057224" type="score_change" time="2018-08-
20T19:13:56+00:00" team="away" x="10" y="48" home_score="4"
away_score="5"/>
    <event id="451057354" type="score_change" time="2018-08-
20T19:14:41+00:00" team="home" x="88" y="48" home_score="5"
away_score="5"/>
    <event id="451057474" type="score_change" time="2018-08-
20T19:15:16+00:00" team="away" x="10" y="48" home_score="5"
away_score="6"/>
    <event id="451057794" type="score_change" time="2018-08-
20T19:16:32+00:00" team="away" x="10" y="48" home_score="5"
away_score="7"/>
    <event id="451057816" type="match_ended" time="2018-08-
20T19:16:36+00:00"/>
    <event id="451255508" type="yellow_card" time="2018-08-
21T12:57:08+00:00" match_time="104" team="home">
        <player id="sr:player:132155" name="Zimmer, Maximilian"/>
    </event>
</timeline>
</match_timeline>
```

/summary.xml example:

```
<?xml version="1.0" encoding="UTF-8"?>
<match_summary xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
generated_at="2018-10-30T11:16:57+00:00"
xmlns="http://schemas.sportradar.com/sportsapi/v1/unified"
xsi:schemaLocation="http://schemas.sportradar.com/sportsapi/v1/unified
http://schemas.sportradar.com/bsa/unified/v1/xml/endpoints/unified/match_su
mmary.xsd">
<sport_event id="sr:match:14704749" scheduled="2018-08-20T16:30:00+00:00"
start_time_tbd="false">
    <tournament_round type="cup" name="cupround"/>
    <season id="sr:season:54377" name="DFB Pokal 18/19" start_date="2018-
08-17" end_date="2019-05-26" year="18/19"
tournament_id="sr:tournament:217"/>
    <tournament id="sr:tournament:217" name="DFB Pokal">
        <sport id="sr:sport:1" name="Soccer"/>
        <category id="sr:category:30" name="Germany" country_code="DEU"/>
    </tournament>
    <competitors>
        <competitor id="sr:competitor:2532" name="Energie Cottbus"
country="Germany" country_code="DEU" abbreviation="COT" qualifier="home">
```

```
<reference_ids>
                <reference_id name="betradar" value="11135"/>
            </reference_ids>
        </competitor>
        <competitor id="sr:competitor:2538" name="SC Freiburg"
country="Germany" country_code="DEU" abbreviation="SCF" qualifier="away">
            <reference_ids>
                <reference_id name="betradar" value="11095"/>
            </reference_ids>
        </competitor>
    </competitors>
    <venue id="sr:venue:13" name="Stadion der Freundschaft"
capacity="22528" city_name="Cottbus" country_name="Germany"
map_coordinates="51.751263,14.344669" country_code="DEU"/>
</sport_event>
<sport_event_conditions>
    <referee id="sr:referee:70951" name="Osmers, Harm"
nationality="Germany"/>
    <venue id="sr:venue:13" name="Stadion der Freundschaft"
capacity="22528" city_name="Cottbus" country_name="Germany"
map_coordinates="51.751263,14.344669" country_code="DEU"/>
    <weather_info pitch="good" weather_conditions="medium"/>
</sport_event_conditions>
<sport_event_status status="closed" match_status="ap" home_score="5"
away_score="7" winner_id="sr:competitor:2538" status_code="4"
match_status_code="120">
    <period_scores>
        <period_score home_score="0" away_score="0" type="regular_period"
number="1" match_status_code="6"/>
        <period_score home_score="1" away_score="1" type="regular_period"
number="2" match_status_code="7"/>
        <period_score home_score="1" away_score="1" type="overtime"
match_status_code="40"/>
        <period_score home_score="3" away_score="5" type="penalties"
match_status_code="50"/>
    </period_scores>
</sport_event_status>
<coverage_info level="gold" live_coverage="true" covered_from="tv">
    <coverage includes="basic_score"/>
    <coverage includes="key_events"/>
    <coverage includes="detailed_events"/>
    <coverage includes="lineups"/>
    <coverage includes="commentary"/>
</coverage_info>
<statistics>
<totals>
        <teams>
            <team id="sr:competitor:2532" name="Energie Cottbus">
                <statistics cards="3" corner_kicks="4" yellow_cards="3"/>
            </team>
            <team id="sr:competitor:2538" name="SC Freiburg">
                <statistics cards="4" corner_kicks="7" yellow_cards="4"/>
            </team>
        </teams>
```

```
</totals>
    <periods>
        <period name="1st half">
            <teams>
                <team id="sr:competitor:2532" name="Energie Cottbus">
                    <statistics cards="1" corner_kicks="1"
yellow_cards="1"/>
                </team>
                <team id="sr:competitor:2538" name="SC Freiburg">
                    <statistics cards="1" corner_kicks="2"
yellow_cards="1"/>
                </team>
            </teams>
        </period>
        <period name="2nd half">
            <teams>
                <team id="sr:competitor:2532" name="Energie Cottbus">
                    <statistics cards="1" corner_kicks="1"
yellow_cards="1"/>
                </team>
                <team id="sr:competitor:2538" name="SC Freiburg">
                    <statistics cards="1" corner_kicks="5"
yellow_cards="1"/>
                </team>
            </teams>
        </period>
        <period name="1st extra">
            <teams>
                <team id="sr:competitor:2532" name="Energie Cottbus">
                    <statistics cards="0" corner_kicks="0"
yellow_cards="0"/>
                </team>
                <team id="sr:competitor:2538" name="SC Freiburg">
                    <statistics cards="1" corner_kicks="0"
yellow_cards="1"/>
                </team>
            </teams>
        </period>
        <period name="2nd extra">
            <teams>
                <team id="sr:competitor:2532" name="Energie Cottbus">
                    <statistics cards="1" corner_kicks="2"
yellow_cards="1"/>
                </team>
                <team id="sr:competitor:2538" name="SC Freiburg">
                    <statistics cards="1" corner_kicks="0"
yellow_cards="1"/>
                </team>
            </teams>
        </period>
    </periods>
</statistics>
</match_summary>
```

Live match information reports the current information about a particular ongoing match (only matches with status live, suspended, or ended should be here – status in closed could be temporarily present here).

Table 32 - Live information metadata

Name	Description
`id`	The ID of the event (match, race).
`competitors`	The sub elements list the competitors of this sport event. Typically, team-based match the competitors will be two teams.
`venue`	Where the sport event takes place.
`sport_event_status`	Contains current status of the match, and basic score information'.
`status`	Reflects the high-level status of the match itself. See section 4.5.4 for more information.
`reporting`	active/suspended/stopped – only present when status = live or suspended. Describes if Betradar has someone reporting directly from the game (either in venue, or using TV). If the status is suspended, Betradar has temporarily lost contact with the scout.
`match_status`	*Sports specific* - the current status/time/period of the game.
`home_score`	*Sports specific* - the current score for the home-team in a team-based match.
`away_score`	Sports specific - the current score for the away-team in a team-based match.
`match_time`	*Sports specific* - tells how long the event has been ongoing according to the match clock (which depending on the sport may stop or not).
`remaining_time`	*Sports specific* - describes how much time is left in the match.
`clock_stopped`	*Sports specific* - only applicable if the sport has a match clock – true if the match clock is stopped.
`winner_id`	Set when there is a winner and set to the ID of the winning competitor.
`coverage_info.level`	Can be bronze/silver/gold and for soccer also platinum. Describes at a high-level how much datapoints that will be collected for a particular event. Bronze is the lowest level. Platinum is the highest level. Exactly what is covered is different for different sports. Bronze typically means just the overall score and possibly the period-scores. Gold is normally the most detailed level Betradar offers and Silver is in between.
`coverage_info.live_coverage`	True if Betradar plans to have live coverage of the event. (Note this does not necessarily mean Betradar will offer LiveOdds for the events only that we will have live sports data for the match –

	see liveodds booking status for whether the match will be covered by LiveOdds)

Events in the Timeline Endpoint

The following events are available in the timeline endpoint, if applicable to the sport and if Betradar is covering the match live.

Table 33 - Events available in the timeline endpoint

Event Type	Description
`match_started`	When the match starts
`period_start`	When each period/quarter/set/frame starts
`break_start`	When a period/quarter ends and a break between periods start
`match_ended`	When the match ended (note that scores and statistics may still be updated after this event. This is a signal that the match ended, not that the data-entry for the match is complete – what the sport_event_status status getting set to closed for the latter)
`period_score`	For set-based sports (Tennis, Volleybal etc.) period_score reflects the current set score. For non-set-based sports (soccer, basketball etc.), the period_score is reported once at the end of periods – listing the number of points/goals made by each team during that period.
`score_change`	Any time the top-level score changes. For non-set-based sports (soccer, basketball, icehockey etc.) this means every time someone scores (i.e. goal / point) For set-based sports, a score change event is sent after the end of each period reflecting who won the past set.

In addition, for soccer the following event types are used:

Table 34 - Extra event types

Event Type	Description
`corners`	A corner kick happened

`yellow_card`	The referee has officially cautioned a player
`red_card`	The referee is sending off the player
`yellow_red_card`	The referee has cautioned a player a second time causing a send-off

Sportradar offers more detailed events for many of the sports in the Media Sports APIs offered separately. There are timeline endpoints in the individual Media Sports API packages too that uses the exact same format, but with a lot more details. Clients that wish to have access to these details can obtain them from the other Sports API packages for a fee. Please contact your Betradar representative to discuss the details.

4.5.2. Results information

Results information provides the same information as for live information, but only when the status of the match is closed (ID 4): `<sport_event_status status="4”`.

```xml
<match_summary xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://bsa.betradar.com/v1" generated_at="2016-02-
10T10:04:17+00:00"xsi:schemaLocation="http://bsa.betradar.com/v1
/schemas/v1/soccer/match_summary.xsd">
  <sport_event id="sr:match:8761304" scheduled="2016-02-05T00:00:00+00:00">
    <tournament_round type="table" number="2"/>
    <tournament id="sr:tournament:231" name="Primera Division">
      <sport id="sr:sport:1" name="Soccer"/>
      <category id="sr:category:281" name="Venezuela"/>
    </tournament>
    <competitors>
      <competitor id="sr:competitor:6231" name="DVO Tachira"
qualifier="home"/>
      <competitor id="sr:competitor:6237" name="Zamora FC"
qualifier="away"/>
    </competitors>
  </sport_event>
  <venue id="sr:venue:20" name="Pueblo Nuevo" capacity="38755"
city_name="San Cristobal" country_name="Venezuela"
map_coordinates="7.786641,-72.197746"/>
  <sport_event_status status="4" home_score="1" away_score="2"
winner_id="sr:competitor:6237">
    <period_scores>
      <period_score home_score="1" away_score="1" type="period"
number="1"/>
      <period_score home_score="0" away_score="1" type="period"
number="2"/>
    </period_scores>
  </sport_event_status>
</match_summary>
```

4.5.3. Resulting coverage per tournament

The level of details Betradar provides per match depends on the coverage-level for a particular tournament/league/competition.

To find out exactly what we guarantee per tournament/league/completion, you can use the Betradar Coverage document:

https://www.betradar.com/wp-content/uploads/sites/4/2014/10/Betradar_Coverage.pdf

First look up the Sport you are interested in, then find the tournament/league/competition you are looking for. Look at the level number in the resulting-column. Now scroll-down to the resulting table, there you will find the Sport and the level number. Now you see what resulting information we guarantee to provide for that tournament.

4.5.4. Sport event status in the API

Sport Event Status is an element provided in the summary and timeline endpoints in the API. *Status* is the only required attribute, this attribute describes the current status of the sport-event itself. This element contains the high-level status of the match including status, score etc.

Note: Sometimes live-only matches will never get the *status* element set to status="closed" in the API, they will stay in status="ended". This is because matches are put in "closed" when production has prematch-resulted these matches. So therefor, if they are only live covered (see scope of the producers), they will never be "closed", only "ended".

The status attribute

The status attribute is an enumeration in the API and has a finite set of states as outlined in the state diagram below:

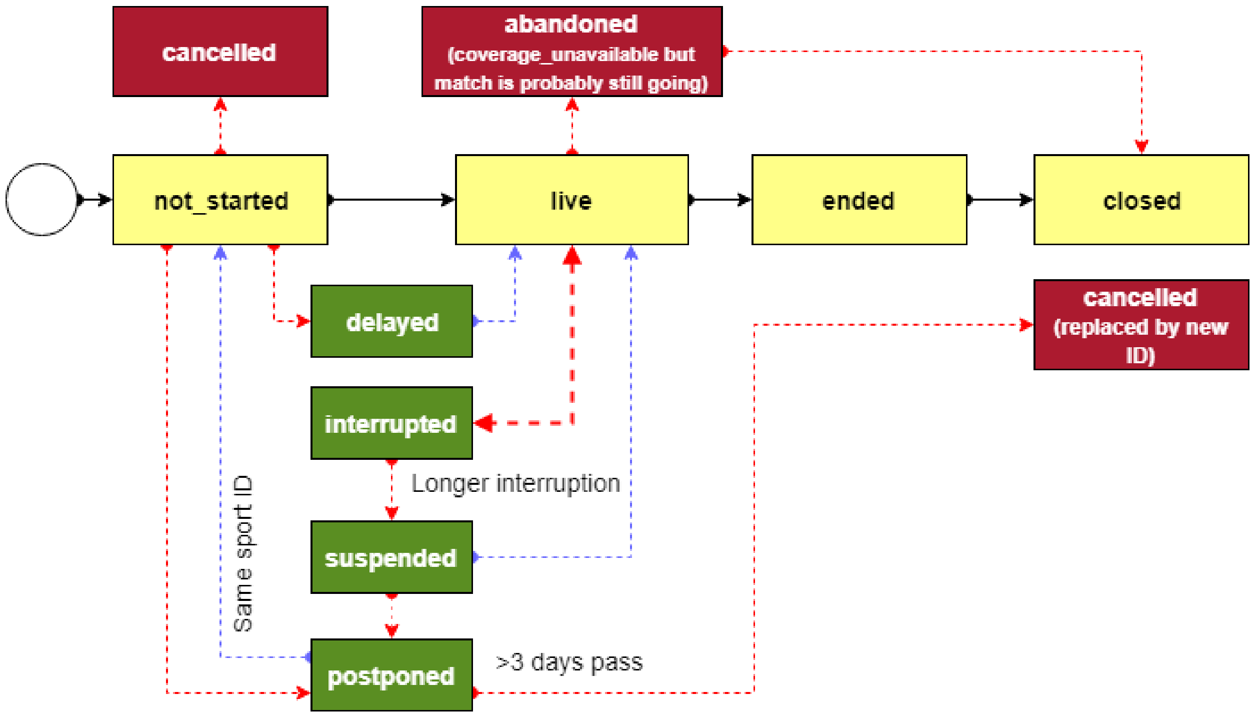


Figure 7 – Sport event status attribute illustration. Yellow blocks are statuses that are expected from the odds change messages.

Table 35 - sport event status attributes

State	Description
`not_started`	The match has not started yet. (Alternatively, Betradar has no live coverage of the event, the match has started but we do not know this. The match will then move to closed when Betradar enters the match results)

`live`	The match is live and covered by Betradar.
`ended`	The match has finished, but results have not been confirmed yet.
`closed`	The match is finished, results confirmed, and no more changes are expected to the results (only for events covered by pre-match producer).
`cancelled`	The sport event (either the actual match, or this Betradar representation of the match) has been cancelled
`delayed`	The sport event start has been delayed from scheduled start (most often seen for tennis).
`interrupted`	The sport event has been temporarily interrupted. Interruption is expected to be just a few minutes. Longer interruptions may lead to a match being suspended, or possibly postponed.
`suspended`	The sport event looks to be interrupted for a longer period than a few minutes
`postponed`	The sport event has been postponed and will be played at a later date. Typically, if the later date is more than 3 days away. This sport event id will be cancelled and replaced by a new id. If the match is postponed to just one or two days from now, the same sport-id will change its state just before match start.
`abandoned`	Used to indicate that Betradar has no live coverage or has lost live coverage but match is still likely ongoing.

Note: The 4 states **not_started**, **live**, **ended**, and **closed** are the only event statuses that are being sent out in the odds_change messages. The rest of the states are only available through the API. Because the API can be slightly delayed compared to the odds change messages, it is therefore recommended that clients prioritize the states in the odds change message, not the API.

Relationship to odds_change sport_event_status

When providing live odds for a live match, the odds_change message normally contains the sport_event_status element to provide an accurate representation of what the status was of the match when the odds were generated.

Due to caching and timing differences, the API and the odds_change message may temporarily report different values. For live matches the status as reported in the odds_change should always be used if available. The API is more likely to lag behind the odds_change message, but it could occasionally be ahead (The difference should always be less than a handful of seconds).

The odds_change message represents statuses using integer codes due to performance reasons, whereas in the API string representations of the statuses are provided for clarity. However, the meaning of the states is the same. In particular, for the status attribute the odds_change message uses only a few of the possible states available in the API, so the states in the API is a superset of those used in the odds_change message. The states not used in odds_change are typically stopped states where no live odds are provided, and hence no odds_change messages sent (states such as delayed, interrupted, postponed, cancelled, etc.).

AggregateHomeScore/Away event

The *aggregate_home_score* and *aggregate_away_score* is present for two-legged ties (i.e. soccer). *aggregate_*_score* represents the total score for each team, summing up the results of the first and second match. Please note that *aggregate_*_score* is only present in the **second** match of the two-legged-tie-series.

4.6. Recovery – API

This section includes instructions and details relevant to performing a recovery in Unified feed using the API. Please make sure to read the provided description and information regarding the different endpoints and if available, notes on performing recovery in edge cases.

Table 36 - Recovery endpoints

HTTP	Endpoint	Description
POST	`(product)/recovery/initiate_request?after=(timestamp)[&request_id=(x)][&node_id=(y)]`	All odds changes for sport events handled by this product. The timestamp must be no longer than 72 hours in the past
POST	`(product)/odds/events/(id)/initiate_request`	All current odds for all the markets for one single match/race. The event must be either not_started or live.
POST	`(product)/stateful_messages/events/(id)/initiate_request`	All stateful messages for the specified match/race. The event can be up to 30 days in the past.

All of these endpoints return either success or failure. If success, the actual messages come as one or more message per match over AMQP. The recovery sequence ends with a snapshot_complete message being sent:

```
<snapshot_complete request_id="1234" timestamp="1234578" product="3"/>
```

The (product) argument in the API (above table) can be one of: liveodds, pre, BetPal, mts or virtuals:

```
liveodds/recovery/initiate_request?after=(timestamp)[&request_id=(x)][&node_i
d=(y)]
```

All requests can have an additional parameter request_id=X, Where X is an integer of your own choice. The concept is that if this request ID is present, it will be included in all stateful messages resulting from this call as an extra attribute, and it will be included in the snapshot_complete message once all messages have been sent. You set this request_id if you want to track the results of individual requests.

All requests can also have an additional parameter node_id=Y, where Y is an integer of your choice. If you have multiple sessions, Betradar's suggestion is that you use different numbers for different sessions. During the recovery the node_id you passed in will be sent in as the 8th word in the routing key. This way you can add a binding key that ensures that you only receive recovery messages for your particular session, and not for your other sessions.

Note: The SDKs automatically takes care of recovery, and ensures that different instances of the SDK do not receive each other's recovery messages.

There is also the "after" parameter, which is specified in milliseconds since Epoch UTC**.** This is for example: https://currentmillis.com/

The after parameter should normally be set to the timestamp of the last message received. This should be the timestamp in the message itself, not something the client system computed when it read the message. If the timestamp is not too far in the past, the client system is guaranteed to have current odds for any active matches, and also all bet_settlement, bet_cancel and rollbacks that have happened in-between the timestamp and now.

Note: The client system receives current odds for active sport_events. This does not mean all odds changes. If there have been multiple updates to a market, the client system will only receive the most recent one. If the complete odds change history is desired, this has to be requested for an individual match separately. This odds history is not available through the API, but can be downloaded through *Ctrl* or the *Live Booking Calendar* if needed.

Similarly, if there was a bet_settlement, followed by a rollback_bet_settlement, followed by a new correct bet_settlement, you will only receive the new correct bet_settlement and not the previous incorrect bet_settlement and rollback.

Please be aware that multiple bet_settlements can be merged into one single bet_settlment message during recovery as well. For example: "*If a bet_settlment was sent out at half-time for half-time markets, and another bet_settlement was sent after the match, the recovery bet_settlement for the same match will than include both the half-time and post-match markets in one message*".

The client system may also receive messages sent before the timestamp provided. The client system should be built robust enough to handle this. The closer the after timestamp is to now, the better it is. If the last message you received was more than 72 hours in the past, this is too far in the past.

If the client system does not specify the after parameter, this means: "*I am new, just send me current active odds".* The response to this is that all current odds for live matches (where odds are active) are sent. The Live Odds producer handles all the live matches, but the prematch producers sends all matches where we have generated odds, and the match_status description="not_started".

If the after parameter is too far in the past or in the future, it will result in the request getting rejected and if the client system was not subscribed previously, the client system will remain unsubscribed.

4.6.1. Rate limit for recovery endpoint

While a producer is processing a recovery request, it may reject further recovery requests. If a different node_id is specified in the additional recovery request. They will not be rejected unless the rate limit for recoveries has been hit.

For more information about rate limits and access restrictions, see THIS section.

4.6.2. Recovering all bet settlements for a specific event

Under some circumstances you may have missed/lost some bet_settlements in a match. Or you may want to recover the state of a match the results for a match that ended more than three days ago. In such cases, you can call a specific end-point to recover all bet_settlements, bet_cancels and rollbacks done for a particular event.

4.6.3. Recovery sequence

Proper recovery requires the following steps:

- Before starting the recovery: Call the daily schedule for the next 3 days and cache the fixtures for all matches received. This will reduce the number of individual API calls later on
- Ensure you have an established AMQP session
- Call the recovery endpoint
- Until you receive a snapshot_complete for the previous recovery call, process like this:
    - If they are odds changes with no recovery id, process them as normal. If they are odds_change with recovery id and you have no recent update on the specified sport event, process them as normal odds change updates. If they are odds_changes with recovery_id and you have already received a recent update, ignore this message (you have already received a more recent update)
    - If they are stateful messages (bet settlement, rollback bet settlement, bet cancel, rollback bet cancel) - store them in a queue for later processing
- When snapshot complete for the recovery request is received
    - Process all stateful messages in the queue received in the previous step
- After the snapshot complete: you can process all stateful messages directly without queueing them up.
- As soon as you have received an odds_change for a sport_event even under the recovery, you open up the markets for that sport event. A conservative solution can wait until snapshot_complete has been received and open up all markets for all received sport-events then, but this is not strictly necessary

Recovery sequence when an <alive product=”x” subscribed=”0”/> is received

Whenever an alive message is received and the subscribed attribute in the message is set to 0, This means that a particular odds producer has been down (due to maintenance or some system error), and your system needs to initiate a partial recovery. This is done following the same steps as above, but in step two, only an API-call for the product that reported it was down is called.

i.e. you receive `<alive product="1" subscribed="0"/>`

you call pre/recovery/initiate_request and process messages as outlined above until *snapshot_complete* is received.

4.7. Probabilities and cashout endpoints

The probability endpoints can be found near the bottom of our self-service documentation.

- For the production environment: https://iodocs.betradar.com/unifiedfeed
- For the integration environment: https://iodocs.betradar.com/ufstaging

Currently, the cashout service is available for matches that fit the following criteria:

- The match is for a cashout supported sport: Soccer, Basketball, Tennis, Table tennis, Badminton, Beach volleyball, Volleyball, Ice hockey, Handball and Squash
- The match is active in Live Odds (and you have Live Odds access to the match): Live Odds is offered for this match; odds are enabled 15 minutes before match start

Matches that do not fit the above criteria will not be available in the cashout service.

The purpose of the probabilities is to give Betradar clients a way to offer their customers the option to cash out on bets that have already been placed, but are not settled yet. This is done in order to increase turnover. To cash out a bet, the client needs to know the current probability of the outcomes, so that they can offer a reasonable return for a customer that wants to cash out a bet before the bet can be definitively settled. To do this, our probabilities service offers probabilities for all active markets and all markets that have previously been offered for the match.

Currently we offer probabilities down to 1e-10 (0.0000000001), and markets with a lower probability than this will be ignored. The feed will produce a message that has a valid odds value, but the probability value will be missing from this market outcome.

```
<market favorite="1" status="1" id="435" specifier="framenr=4">
    <outcome id="1" odds="1.7" probabilities="0.54865 active="1"/>
    <outcome id="2" odds="3.8" probabilities="0.22432 active="1"/>
    <outcome id="3" odds="20.0" active="1"/> …Probability lower than 1e-10…
</market>
```

Note: Customers are strongly recommended to cache and use probabilities where available for speed and efficiency reasons. The probabilities API is primarily intended for market lines with low likelihood that are currently not offered in the odds_change message (e.g. a totals line with a total far from the current most likely total).

In order to offer cashout probabilities, we provide a cashout API where the clients can request the cashout probabilities for all markets for all active matches.

Table 37 - Probabilities endpoints

Endpoint	Path Parameters	Description
`/probabilities/{urn_type}:{id}`	Match id	Sport event probabilities – Get probabilities for a sport event
`/probabilities/{urn_type}:{id}/{market_id}`	Match id, market id, market	Market probabilities – Get probabilities for a sport event's specific market.
`/probabilities/{urn_type}:{id}/{market_id}/{specifier}`	Match id, market id, market specifiers	Market probabilities with specifiers – Get probabilities for a sport event's specific market with specifiers

4.7.1.XML content

The following table has an overview over what elements are present in the Cashout XML response.

Table 38 - Elements in the cashout response

Element	Description	Attribute	Description	Possible values
`cashout`	Top level element.	product	What product the probabilities come from	Currently only 1 for Live Odds
		event_id	Match id	The match id in urn format, sr:match:{matchid}

		timestamp	When the message was created	The number of milliseconds since January 1, 1970 UTC
`sport_event_status`	Subelement of cashout. Has attributes for the sport event's current status. See more in THIS section for additional information.	status	The current betstatus	0 (match not started) 1 (betstart, also early betstart if the bookmaker has enabled early betstart in the xml config) 2 (betstop) 3 (match is ended) 4 (match is ended and all markets are cleared)
		match_status	The current match status	**Soccer** 0 (not started) 6 (first half) 7 (second half) 31 (halftime) 32 (awaiting ot) 33 (ot halftime) 34 (awaiting penalties) 41 (first half ot) 42 (second half ot) 50 (penalty shooting) 80 (interrupted) 90 (abandoned) 100 (ended) 110 (after ot) 120 (after penalties)
`odds`	Subelement of cashout. This element contains all the active markets. The odds element with market probabilities are available also when the match is on betstop.			
`market`	Subelement of odds. Contains market information. The markets are only added to the odds element if the match is on betstart.	id	Unified feed market id	Integer
		specifiers	Unified feed market specifiers	String
		status	Market status	0 (inactive) 1 (active) -1 (suspended) -2 (handed_over) -3 (cleared)

				-4 (cancelled)
		cashout_status	The cashout status of the market	1 AVAILABLE (available for cashout) -1 UNAVAILABLE (temporarily unavailable for cashout) -2 CLOSED (permanently unavailable for cashout)
`outcome`	Subelement of market.	id	Unified feed outcome id	Integer
		probabilities	Outcome probability	Raw model output. Not rounded.
		active	If the outcome is active	1 (active) 0 (inactive/deactivated)

4.7.2. Cashout status

Each market element will have a cashout_status attribute determining if the market is available for cashout or not. The value of the cashout_status attribute will be determined by the following rules, applied in prioritized order:

Table 39 - Available cashout statuses for market elements

Id	Name	Description
-2	Closed	Market is settled or canceled.
1	Available	Market is active and match is on betstart or early betstart if the bookmaker has early betstarts enabled in his configuration
-1	Unavailable	Default status. A market will be unavailable in cashout unless any of the rules above apply.

A market will only contain cashout probabilities if the cashout status is Available (cashout_status=”1”).

Example XML

```xml
<cashout product="1" event_id="sr:match:10383187"
timestamp="1488376222936">
    <sport_event_status status="1" match_status="6"/>
    <odds>
        <market status="1" cashout_status="1" id="20"
specifiers="total=1.5">
            <outcome id="425" probabilities="0.6163236925215048"
active="0"/>
            <outcome id="426" probabilities="0.38367630747849524"
active="0"/>
        </market>
        <market status="1" cashout_status="1" id="29">
            <outcome id="41" probabilities="0.5761726108027806"
active="0"/>
            <outcome id="42" probabilities="0.4238273891358624"
active="0"/>
        </market>
        <market status="1" cashout_status="1" id="18" specifiers="total=3">
            <outcome id="114" probabilities="0.6366925169830813"
active="0"/>
            <outcome id="115" probabilities="0.36330748301691873"
active="0"/>
        </market>
    </odds>
</cashout>
```

4.7.3. Error messages

This section covers what error messages the server will reply with if something went wrong or some of the provided information is incorrect, etc.

Table 40 - Server error messages

Symptom	Reason
`XML Message with response code Forbidden`	No authentication token
`XML Message with response code Forbidden`	Invalid authentication token
`500 Internal server error`	Error processing the request
`404 Not Found`	Trying to access resource that does not exist
`403 Forbidden`	Bookmaker is blocked from accessing that resource

4.8. Descriptive endpoints

The endpoints described in this chapter are exclusive to the different producers that deliver odds through Unified Feed.

4.8.1. Producer description

This endpoint describes currently available producers, whether they are activated for the calling client, what recovery endpoint they use and what their ids are.

Table 41 - Producer endpoint

Element/Attribute	Description
`producer.id`	Producer Id as sent in messages
`producer.active`	Set to true if this producer is activated for the calling client otherwise false.
`producer.api_url`	The API starting-url that can be used to request recovery etc for this producer.
`producer.name`	A short name for this producer
`producer.description`	A longer name describing this producer
`producer.scope`	live / prematch

Example XML

```
<producers response_code="OK">
  <producer active="true" api_url="https://api.betradar.com/v1/liveodds/"
description="Live Odds" name="LO" id="1"/>
  <producer active="true" api_url="https://api.betradar.com/v1/pre/"
description="Betradar Ctrl" name="Ctrl" id="3"/>
  <producer active="true" api_url=”https://api.betradar.com/v1/betpal/”
description="BetPal" name=”BetPal" id="4"/>
  <producer active="false"
api_url=https://api.betradar.com/v1/premium_cricket/ description= "Premium
Cricket" name="PremiumCricket" id="5"/>
  <producer active="true" api_url="https://api.betradar.com/v1/vf/"
description = "Virtual football" name="VF" id="6"/>
</producers>
```

4.8.2. Entity descriptions

There are entity endpoints that provide more details about related entities described in fixtures and event information. By prefixing any of the entity paths with a language, the localized version of that entity is provided. The following table shows the main entities that can be accessed; Players, competitors and venues.

Table 42 - Entity endpoint

HTTP	Endpoint	Description
GET	`(lang)/players/(id)/profile.xml`	Name and details about a team player in the requested language (en for English).
GET	`(lang)/competitors/(id)/profile.xml`	Name and details about a competitor (this could be a team, a player (e.g. tennis), a race driver (e.g. F1) in the requested language (en for English).
GET	`(lang)/venue/(id)/profile.xml`	Name and details about a venue in the requested language (en for English).

Player profile XML example:

```
<player_profile>
<player id="sr:player:947" name="Barnard, Lee" full_name="Lee Barnard"
type="forward" date_of_birth="1984-07-18" nationality="England"
country_code="ENG" height="178" weight="68" gender="male"/>
</player_profile>
```

Competitor profile XML example:

```
<competitor_profile>
<competitor id="sr:competitor:245739" name="WUNDER, MARKUS"
country="Germany" country_code="DEU" abbreviation="WUN">
    <sport id="sr:sport:5" name="Tennis"/>
    <category id="sr:category:785" name="ITF Men"/>
</competitor>
</competitor_profile>
```

Venue summary XML example:

```
<venue_summary>
<venue id="sr:venue:3" name="Krohnsminde Idrettspark" city_name="Bergen"
country_name="Norway" country_code="NOR"/>
</venue_summary>
```

4.8.3. Betting related descriptions

There are multiple endpoints that provide static data and describe IDs referred to odds information messages. These endpoints can be translated to a requested language.

Table 43 - Endpoints for odds information messages

HTTP	Endpoint	Description
GET	`descriptions/betstop-reasons.xml`	Describes all bet stop reasons.

GET	`descriptions/betting_status.xml`	Describes all the possible reasons for a previous betstop (as provided in the message odds_change in the odds element).
GET	`descriptions/void_reasons.xml`	Describes all void-reasons (as provided in the message bet_settlement in market void_reason)
GET	`descriptions/(lang)/markets.xml[?include_mappings=true]`	Describes all currently available markets translated to the requested language (en for English). If include_mappings=true is added as a parameter, the response will include a section for each market on how this market is mapped to equivalent markets in other Betradar products.
GET	`descriptions/(lang)/match_status.xml`	Describes all sports specific match status codes used during live matches in the odds_changes message. Translated to available languages.

Market Descriptions

Table 44 - Market description

Element/Attribute	Description
`market.id`	Market Id as used in messages
`market.description`	Human readable name of the market (translatable). Has some markers that need to be replaced (See section below for more information)
`market.groups`	Pipe "|" separated list of groups that this market belongs to. A group is just a name/ string. These groups can be used in a bet_stop message to stop select markets.
`m.outcome_type`	Describes the type of outcome – allowed values are: • **competitor:** Competitor type outcomes means that the ids will be competitor ids (and possibly also some fixed outcomes such as "No Winner" etc. that is only using an integer number id and not an URN type id) - the competitors can only be ids of competitors listed to be part of this particular sport-event (otherwise use free text) • **player:** This outcome means that the ids will be players that play for either of the competitors in the specified sport-event. line up (preferred when available) or team-profile endpoints should be available to retrieve available players • **free_text:** This outcome means that the ids will be variant ids, and will have to be looked up (the market will have a variant and this variant will list the market text as well as the outcome texts, and the ids in this message will refer to the outcome text).

`outcome.id`	Outcome id as used in odds and settlement messages
`outcome.name`	Human readable name of the outcome (translatable).
`specifiers`	Lists the additional specifiers that will always be sent for this market
`specifier.name`	The name of the specifier as sent in the messages
`specifier.type`	What type the values for this specifier will have. (integer, decimal, variable_text)
`attributes.attribute.name`	Lists special attributes that apply to particular markets. See table market description attributes below for descriptions of market attribute names.

Market XML example

```
<market id="234" name="Highest scoring quarter" groups="all">
        <outcomes>
            <outcome id="920" name="1st quarter"/>
            <outcome id="921" name="2nd quarter"/>
            <outcome id="922" name="3rd quarter"/>
            <outcome id="923" name="4th quarter"/>
            <outcome id="924" name="equal"/>
        </outcomes>
    </market>
    <market id="176" name="1st half - corner handicap" groups="all">
        <outcomes>
            <outcome id="1714" name="{$competitor1} ({+hcp})"/>
            <outcome id="1715" name="{$competitor2} ({-hcp})"/>
        </outcomes>
        <specifiers>
            <specifier name="hcp" type="decimal"/>
        </specifiers>
</market>
```

Market Attribute Name Descriptions

Table 45 - Market attribute names

Attribute Name	Description
`is_flex_score`	A flex score market. See Flex Score Markets below for a description.
`deprecated`	This market is no longer in use, but could still be seen for historical events (e.g. in replay or when requesting recovery)

XML example

```
<markets>
  <market id="1" description="Three-way" groups="all">
    <outcomes>
      <outcome id="1" name="{$competitor1}"/>
      <outcome id="2" name="Draw"/>
      <outcome id="3" name="{$competitor2}"/>
    </outcomes>
  </market>
 <market id="216" name="{!setnr} set game {gamenr} - race to {pointnr}
points" groups="all">
   <outcomes>
      <outcome id="4" name="{$competitor1}"/>
      <outcome id="5" name="{$competitor2}"/>
   </outcomes>
   <specifiers>
     <specifier name="setnr" type="integer"/>
     <specifier name="gamenr" type="integer"/>
     <specifier name="pointnr" type="integer"/>
   </specifiers>
  </market>

<markets>
  <market id="1" description="Trevägs">
    <outcomes>
      <outcome id="1" name="Hemmalaget vinnare">
      <outcome id="2" name="Lika">
      <outcome id="3" name="Bortalaget vinnare">
    </outcomes>
  </market>
</markets>
```

Betstop Descriptions

```
<betstop_reasons>
  <betstop_reason id="12" description="Match ended"/>
  <betstop_reason id="1" description="Possible goal"/>
  <betstop_reason id="0" description="Unspecified"/>
  <betstop_reason id="-1" description="Loss of TV coverage"/>
  <betstop_reason id="-2" description="Loss of connection to scout"/>
</betstop_reasons>
```

Betting Status Descriptions

Note: The betting status description is only sent when (if) the Live Odds producer is in early betstart.

```
<betting_status_descriptions response_code="OK">
    <betting_status id="0" description="UNKNOWN"/>
    <betting_status id="1" description="GOAL"/>
    <betting_status id="2" description="DANGEROUS_FREE_KICK"/>
    <betting_status id="3" description="DANGEROUS_GOAL_POSITION"/>
    <betting_status id="4" description="POSSIBLE_BOUNDARY"/>
    <betting_status id="5" description="POSSIBLE_CHECKOUT"/>
    <betting_status id="6" description="INGAME_PENALTY"/>
</betting_status_descriptions>
```

Void Reasons

The following table lists all available void reasons for a market. These void reasons will show up on individual markets in bet_settlement and bet_cancel messages/events.

Table 46 - Void reasons

ID	Name	Examples of how it is used
0	Other	Technical issues, other unforeseen reasons
1	No goalscorer	No longer in use (Deprecated)
2	Correct score missing	No longer in use (Deprecated)
3	Results Unverifiable	In very rare cases, Betradar may not be able to verify the results. For example: if the Scout loses connection to the venue and is unable to recreate the necessary sequence of events afterwards.
4	Format change	This void reason should indicate that some markets cannot be settled at all due to the wrong match format, but others were just offered with incorrect odds.
5	Cancelled Event	Match will not be played at all due to: officially cancelled, change of draw, incorrect teams, swapped fixtures
6	Missing goalscorer	No longer in use (Deprecated)
7	Match ended in Walkover	Match ended in walk-over.
8	Dead heat	Used if the dead heat rule is applied for a market, and there is no winner.
9	Retired or Defaulted	One of the competitors retired or defaulted and according to the settlement rules the market should be voided.
10	Abandoned Event	Match started but will not be finished due to weather, riots etc.
11	Postponed Event	Match will be played but 48hours or more later. Typically due to: bad weather, administrative reasons(security missing, ambulance missing, floodlight issues, etc.), fan riots etc.
12	Incorrect Odds	Trader mistakes (Betradar side issues when setting odds)

13	Incorrect Statistics	Incorrect statistics due to data entry issues either by operator or from data feed (incorrect scores, card stats,
14	No Result Assigned	The actual result was not offered as an outcome. Example: who will score the xth point (and point x is not played).
15	Client side settlement needed	Used for very special markets where Betradar cannot offer all outcomes, but basic calculations can be used on the client side to compute the result. (Currently used by World-Number-Service straight hit and premium cricket spread markets). Most vendors can choose to ignore implementing client side settlement.

XML example

```
<void_reasons_descriptions response_code="OK">
    <void_reason id="0" description="OTHER"/>
    <void_reason id="1" description="NO_GOALSCORER"/>
    <void_reason id="2" description="CORRECT_SCORE_MISSING"/>
    <void_reason id="3" description="RESULT_UNVERFIABLE "/>
    <void_reason id="4" description="MATCH_WENT_TO_SHOOTOUT"/>
    <void_reason id="5" description="CANCELLED_EVENT"/>
    <void_reason id="6" description="MISSING_GOALSCORER"/>
    <void_reason id="7" description="MATCH_ENDED_IN_WALKOVER"/>
    <void_reason id="8" description="DEAD_HEAT_NO_WINNER"/>
    <void_reason id="9" description="RETIRED_OR_DEFAULTED"/>
    <void_reason id="10" description="EVENT_ABANDONED"/>
    <void_reason id="11" description="EVENT_POSTPONED"/>
    <void_reason id="12" description="INCORRECT_ODDS"/>
    <void_reason id="13" description="INCORRECT_STATISTICS"/>
    <void_reason id="14" description="NO_RESULT_ASSIGNED"/>
    <void_reason id="15" description="CLIENT_SIDE_SETTLEMENT_NEEDED"/>
</void_reasons_descriptions>
```

4.9. Markets and outcomes

This chapter covers the most relevant information in regards to markets and outcomes in Unified feed. If you are coming over from the legacy Live Odds feed, there are some notable changes to how markets work.

4.9.1. Market mappings to market ids to the other products of Betradar

If the market descriptions are requested with the additional parameter *include_mappings=true*, the market descriptions will contain market mappings that say what Live Odds and LCoO market id that is the same as this unified odds market (or closest).

The below example shows how the market “Total bookings” is described. The mapping section specifies first that for Live Odds (Soccer), this market is known as type=8 and subtype=153 product_id=”1” for Live Odds). Then it says that for LCoO for Soccer (product_id=”3”), that this market is known as 236.

```
<market id="139" name="Total bookings">
  <outcomes>
    <outcome id="13" name="under {total}"/>
    <outcome id="12" name="over {total}"/>
  </outcomes>
  <specifiers>
    <specifier name="total" type="decimal"/>
  </specifiers>
  <mappings>
    <mapping product_id="1" sport_id="sr:sport:1" market_id="8:153"
sov_template=”{total}>
       <mapping_outcome outcome_id=”12” product_outcome_id=”573”
product_outcome_name=”over”/>
       <mapping_outcome outcome_id=”13” product_outcome_id=”571”
product_outcome_name=”under”/>
    </mapping>
    <mapping product_id="3" sport_id="sr:sport:1" market_id="236"
sov_template=”{total}”>
       <mapping_outcome outcome_id=”12” product_outcome_id=”6”
product_outcome_name=”Over”/>
       <mapping_outcome outcome_id=”13” product_outcome_id=”7”
product_outcome_name=”Under”/>
    </mapping>
  </mappings>
</market>
```

The following excerpt for the market “Exact Games” unified odds id: 241. This says that this market is mapped to Live Odds type 8 and subtype 25 for the three sports: Table Tennis (sr:sport:20), Badminton (sr:sport:31) & Squash (sr:sport:37).

```
<mapping product_id="1" sport_id="sr:sport:20" market_id="8:25"
valid_for=”variant=sr:exact_goals:bestof:5”/>
<mapping product_id="1" sport_id="sr:sport:31" market_id="8:25"
valid_for=”variant=sr:exact_goals:bestof:5”/>
<mapping product_id="1" sport_id="sr:sport:37" market_id="8:25"
valid_for=”variant=sr:exact_goals:bestof:5”/>
```

There are some noteworthy changes to keep in mind: In Unified Odds, two markets in two different sports that have the same meaning, have often been mapped to the same unified market. The following excerpt says that the market “Correct Score best of 3” (id 199) is mapped to id 233 for LCoO for Tennis, but to id 350 for LCoO for BeachVolley.

```
<mapping product_id="3" sport_id="sr:sport:5" market_id="233"/>
<mapping product_id="3" sport_id="sr:sport:34" market_id="350"/>
```

Because of the new way unified odds uses specifiers, in some cases where we previously used multiple markets, Unified Odds now use one market with different specifiers. For example, in “Xth quarter handicap” – which used to be one market for each quarter in Live Odds, is now the same market with different specifiers in Unified Odds (Unified Odds market 303):

```
<mapping product_id="1" sport_id="sr:sport:2" market_id="7:44"
valid_for="quarternr=1"/>
<mapping product_id="1" sport_id="sr:sport:2" market_id="7:48"
valid_for="quarternr=2"/>
<mapping product_id="1" sport_id="sr:sport:2" market_id="7:51"
valid_for="quarternr=3"/>
<mapping product_id="1" sport_id="sr:sport:2" market_id="7:54"
valid_for="quarternr=4"/>
```

A special *valid_for* pattern is used for totals markets where certain multiple (*.5) are mapped to the Live Odds normal total market, and other multipliers (*.0, *.25, *.75) are mapped to the Live Odds Asian Total. The below example shows the pattern with the meaning for soccer, for Live Odds this market is called 7:21 if the total specifier ends with .5:

```
<mapping product_id="1" sport_id="sr:sport:1" market_id="7:21"
valid_for="total~*.5">
```

NOTE: In some cases, the attribute “product_ids” will be displayed in addition to the normal “product_id” attribute. This is an indication that some mappings are available for multiple odds producers. This is usually the case when the mapping includes both the Live Odds producer and the BetPal producers. In this case, the attribute will display: product_ids=”1|4” (1 is for Live Odds, 4 is for BetPal).

We are moving away from using the “product_id” attribute, and it will be replaced with “product_ids” in the future.

```
<mappings>
    <mapping product_id="1" product_ids="1|4" sport_id="sr:sport:6"
market_id="8:232" sov_template="{total}">
      <mapping_outcome outcome_id="13" product_outcome_id="2528"
product_outcome_name="under"/>
      <mapping_outcome outcome_id="12" product_outcome_id="2530"
product_outcome_name="over"/>
</mapping>
```

4.9.2. Market descriptions

Market descriptions and outcome descriptions have a template language that can be used to describe the actual market or outcome. This is easiest described with an example:

```
<market id="300" name="Race to {pointnr} points">
```

Anything between curly braces {} can be replaced by the actual market information in an odds_change message. Normally what is in between the curly braces is a name of a specifier and should be replaced by that particular specifier value. Consequently, if you receive an odds_change message like this:

```
<market id="300" specifiers="pointnr=3">
```

You should print this market name as “Race to 3 points”.

If the first character after the opening curly brace is an !, followed by a specifier name, then you should replace the curly-braces expression with the specifier as an ordinal number, illustrated below.

```
<market id="446" name="{!periodnr} period - total">
```

And you receive an odds_change message with the following market info:

```
<market id="446" specifiers="periodnr=2">
```

You should display the market name as: “**Second** period – total”.

If the first character after the opening curly brace is a + followed by a specifier name, then the specifier must be of number type, and you should replace the curly-braces expression with the specifier and add the right +/- sign in front of it.

If the first character after the opening curly brace is a - followed by a specifier name, then the specifier must be of number type, and you should replace the curly-braces expression with the negated specifier and add the right +/- sign in front of it.

There are a few special keywords that can also show up within the curly-braces that have a special meaning:

- {$competitor1} means to replace the expression with the name of the first competitor in the sport event – note this is often occurring in the outcome descriptions.

- {$competitor2} means to replace the expression with the name of the second competitor in the sport even – note this is often occurring in the outcome descriptions
- {$event} means to replace the expression with the name of the event. This is typically used for outrights

NOTE: The SDK takes care of these market description automatically for you, so you don't have to manage any of the above yourself when using the SDK.

Table 47 - Market descriptions

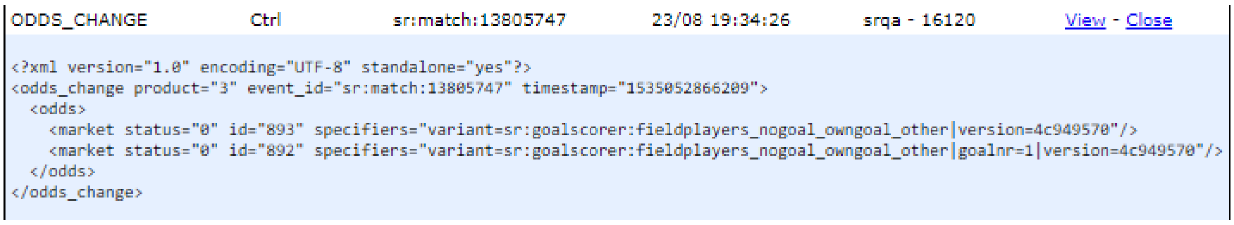
Pattern	Action	Example market description template:	Example specifiers in a message:	Example market description
`{X}`	Replace {X} with the value of the specifier X.	"Race to {pointnr} points"	… specifiers="pointnr=3"	"Race to 3 points"
`{!X}`	Replace {X} with the ordinal value of the specifier X			
`{+X}`	Replace {X} with the value of the specifier X with a +/- sign in front.			
`{-X}`	Replace {X} with the negated value of the specifier with a +/- sign in front			
`{(X+/-c)}`	Replace the expression with the value of the specifier X + or - the number c.	"{!(inningnr+1)}	inningnr=2	"3rd inning"
`{$competitorN}`	Replace with the Nth			

	competitor in the event or if you prefer “TeamN” or “PlayerN” as appropriate			
`{$event}`	Replace with the name of the event.	“Winner of {$event}”	Id=”sr:tournament:1”	“Winner of Euro2016”
`{%player}`	Replace with the name of the specifier (which should be an id typically to a player or competitor)	“{%player} total dismissals"	player=sr:player:1234	“John Rodriquez total dismissals”

Goalscorer markets

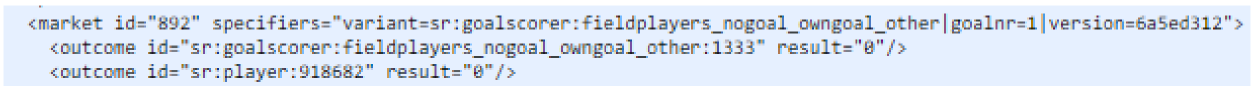
Market versions for first and last *goalscorer* markets are also available. These work the same way as outcome versions for the outright markets mentioned HERE.

When a new version is generated, the old version will be invalidated

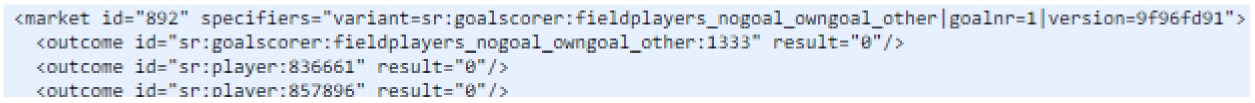


```
ODDS_CHANGE          Ctrl          sr:match:13805747          23/08 19:34:26          srqa - 16120          View - Close

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<odds_change product="3" event_id="sr:match:13805747" timestamp="1535052866209">
  <odds>
    <market status="0" id="893" specifiers="variant=sr:goalscorer:fieldplayers_nogoal_owngoal_other|version=4c949570"/>
    <market status="0" id="892" specifiers="variant=sr:goalscorer:fieldplayers_nogoal_owngoal_other|goalnr=1|version=4c949570"/>
  </odds>
</odds_change>
```

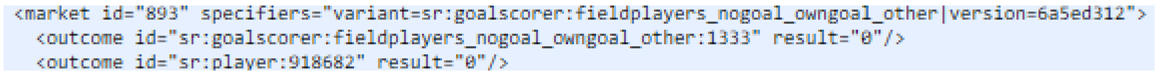
All versions will be included in the settlement for first/last goalscorer markets.



```
  <market id="892" specifiers="variant=sr:goalscorer:fieldplayers_nogoal_owngoal_other|goalnr=1|version=6a5ed312">
    <outcome id="sr:goalscorer:fieldplayers_nogoal_owngoal_other:1333" result="0"/>
    <outcome id="sr:player:918682" result="0"/>
```



```
 <market id="892" specifiers="variant=sr:goalscorer:fieldplayers_nogoal_owngoal_other|goalnr=1|version=9f96fd91">
   <outcome id="sr:goalscorer:fieldplayers_nogoal_owngoal_other:1333" result="0"/>
   <outcome id="sr:player:836661" result="0"/>
   <outcome id="sr:player:857896" result="0"/>
```



```
 <market id="893" specifiers="variant=sr:goalscorer:fieldplayers_nogoal_owngoal_other|version=6a5ed312">
   <outcome id="sr:goalscorer:fieldplayers_nogoal_owngoal_other:1333" result="0"/>
   <outcome id="sr:player:918682" result="0"/>
```

```
<market id="893" specifiers="variant=sr:goalscorer:fieldplayers_nogoal_owngoal_other|version=9f96fd91">
  <outcome id="sr:goalscorer:fieldplayers_nogoal_owngoal_other:1333" result="0"/>
  <outcome id="sr:player:836661" result="0"/>
```

```
<market id="893" specifiers="variant=sr:goalscorer:fieldplayers_nogoal_owngoal_other|version=9f96fd91">
  <outcome id="sr:goalscorer:fieldplayers_nogoal_owngoal_other:1333" result="0"/>
  <outcome id="sr:player:836661" result="0"/>
```

All versions (first/last goalscorer) will be sent in a UF recovery.

Mapping for goalscorer markets

The following list shows the mapping for first and last goalscorer markets based on the different selection of feed options:

1. "only players – no goal" -> map to 38/39; with no goal outcome always void and deactivated
2. "players and no goal" -> map to 892/893 with version, no goal outcome that support settlement
3. "players, no goal, and other" -> map to 892/893 with version, no goal outcome, other outcome
4. "players, no goal, other, and own goal" -> map to 892/893 with version, no goal outcome, other outcome, own goal outcome

4.9.3. Market & outcome description variant text

Some markets and outcomes have variable descriptions. This is for example the case for many outright markets and outcomes, as well as some dynamic markets – most importantly the correct score market, but also some cricket markets. These markets have a special specifier *variant* and this specifier is set in the odds_change message to a urn that looks like this: pre:markettext:1234.

To find the actual market name you need to do an additional API-call to descriptions/en/markets/<market>/variant/<urn>. The returned document will return the market name and where applicable, outcome names. Note that the same market can have different market descriptions at different times if the variant-specifier changes.

```
<market_descriptions response_code="OK">
    <market id="241" name="Exact games" variant="sr:exact_games:bestof:5">
        <outcomes>
            <outcome id="sr:exact_games:bestof:5:39" name="3"/>
            <outcome id="sr:exact_games:bestof:5:40" name="4"/>
            <outcome id="sr:exact_games:bestof:5:41" name="5"/>
        </outcomes>
    </market>
</market_descriptions>
```

Two markets with the same market id but with different variant descriptions should be treated as different market lines. This is the same way it works otherwise (i.e. market-id + specifiers uniquely identify a market line). Consequently, the Betradar system will handle and settle these market lines independently of each other.

4.9.4. Flex score markets

There is a special type of markets called flex score markets. These are only offered live and are an alternative to correct score markets. The correct score markets have a fixed set of outcomes. If the score goes above the fixed set of outcomes the ordinary correct score market is deactivated. The flex score markets offer a set of outcomes based on the current score of the match, hence they are always available regardless of how many goals that are scored.

These markets are displayed in the market description endpoint by having an extra <attribute>-element with the name *is_flex_score*. For these markets the proper outcome names are created by taking the current score and "adding" the listed outcome name to that score. For example, if the current score is 5:3 and the outcome name is 1:1. Betting on that outcome means betting on a final outcome of 6:4.

4.10. Match bookings

Note that in the daily schedule, all sport-events are listed with a "live-odds" attribute. This attribute lists the match-booking state and can be in the following states: ***booked*** (already booked), ***bookable*** (can be booked), ***buyable*** (need to be bought before it can be booked, contact your sales-representative), and **not*_ailable*** (currently not available at all for live booking).

This is only for live odds where a customer can request booking of a particular match.

Table 48 - Match booking endpoint for Live Odds

HTTP	Endpoint	Description
POST	`liveodds/booking-calendar/events/(id)/book`	POST on this Endpoint to put a match in the booking calendar.

Example response:

```
<response response_code="OK">
    <action>Request for booking an event : sr:match:12345678 from
bookmaker: 1234 received</action>
    <message>OK.</message>
</response>
```

4.11.Administrative

The following is an administrative message that can be used to find out what the callers bookmaker id is (which is needed to find the name of the AMQP vhost to connect to), and when the callers access token will expire.

Table 49 - User information endpoint

HTTP	Endpoint	Description
GET	`users/whoami.xml`	Lists the callers bookmaker id, AMQP access point and when the provided access token will expire.

Example response

```
<bookmaker_details response_code="OK" expire_at="2016-07-08T11:58:53Z"
bookmaker_id="15879" virtual_host="/unifiedfeed/15879"/>
```

Result codes for users/whoami.xml:

Table 50 - Results codes from the user information endpoint

Token	HTTP Response Code	Description
Valid token provided	**200**	OK
No token provided	**400**	Bad Request
Unknown Token	**404**	Not found

Expired token	**403**	Forbidden: "Error: Token is not valid, perhaps it has expired?"
UF Package Missing	**403**	Forbidden: "Error: Bookmaker <id> does not have access to unified feed"

4.12.Endpoint update frequency

To avoid unnecessary re-computation of endpoints, the following endpoints are cached and will not be recomputed until the cache time expires:

- **S**tatic: 1 day - endpoint almost never changes. Example: List of sports
- **L**ong: 1 hour - endpoint very rarely changes. Example: Schedules.
- **M**edium: 10 minutes - Endpoint affected by played matches, but not time critical: stats
- **S**hort: 1 minutes - Endpoint affected by played matches, that needs quick update
- **L**ive: 1 seconds - Endpoint needs real-time updates.

Note: This cashing is done on the server side and does not need to be performed on the client side.

Table 51 - Endpoint types and Cache times

Endpoint	Type	Cache TTL
sports.xml	Static	1 day
venues/{id}/summary.xml	Static	1 day
sports/{id}/tournaments.xml	Long	1 hour
sports/(id)/categories.xml	Long	1 hour
players/(id)/profile.xml	Long	1 hour
schedules/(date)/schedule.xml	Long	1 hour
tournaments.xml	Long	1 hour
tournaments/(id)/info.xml	Long	1 hour
tournaments/(id)/schedule.xml	Long	1 hour
competitors/(id)/profile.xml	Medium	10 min
schedules/(date)/results.xml	Short	1 min
schedules/live/schedule.xml	Live	1 sec
sport_events/(id)/fixture.xml	Live	1 sec

`sport_events/(id)/summary.xml`	Live	1 sec
`sport_events/{id}/timeline.xml`	Live	1 sec

Cashing remarks for implementation

On the client side, it is possible to cache fixtures. If you do this, you have to invalidate a particular fixture when you receive a fixture_change message, for that fixture. 24 hours is a reasonable caching time.

Variant descriptions can be cached for 24 hours or more. The only reason a particular variant description would change is in case of a data entry error.

5. Unified Odds – Replay server

The Betradar System provides a replay server. Using the replay server, you can replay all messages sent for a particular sport event. The sport event has to be older than 48 hours. The matches are re-playable forever. Betradar reserves the right to remove events that are older than 2 years.

Note: Both *production* access tokens and *integration* access tokens can be used with the replay server.

Server address: *replaymq.betradar.com*

The replay server is a message queue server. On this server you will only receive messages for matches you have in your replay list that you are currently replaying. The replay server is controlled through a separate RESTful API. Here you first setup what matches you want to have replayed, then you start the replay and the messages will start arriving.

You can control the speed of the replay with a parameter when you start replaying. This allows you to for example replay the messages 10x faster than they were recorded (This is also the default replay speed).

Furthermore, you can also control the maximum delay between messages. This means that if the delay between two messages would be more than this much, the delay will be shortened to the maximum delay. The default value here is 10 seconds. This guarantees that no two messages ever arrive more than 10 seconds apart. Typically, this is very valuable for prematch odds when there can be a long time between updates.

Note: The messages replayed are the messages sent to a special replay-user, so they may not 100% reflect the odds-values you receive (in particular not your odds-key). This is also means that you may have the ability to replay matches that you did not have access to yourself.

Alive message: When a replay is being started (either a scenario or a match), the replay server will send out alive messages for product 1 and product 3, every 10 seconds in the current time.

Alive message XML example:

```
<alive product="1" subscribed="1" timestamp="1521709923892"/>
<alive product="3" subscribed="1" timestamp="1521709923893"/>
```

5.1. Example replays

The following table lists events for different sports, products and properties. Being able to handle these is a good start when building a Unified Odds client.

Table 52 - Available replays

	Event Id	Sport	Tags	Description
1	`sr:match:11538563`	American football		NFL 2018
2	`sr:match:13552497`	American football	Playerprops	American Football Game with PlayerProps
3	`sr:season:40175`	American football	Outrights	Season outrights / Long-term outrights (NFL 2017/18)
4	`sr:match:12587650`	Aussie Rules		Aussie Rules (AFL 2017 Final)
5	`sr.match:13600687`	Badminton		Badminton
6	`sr:match:12906380`	Baseball		Baseball - MLB Final 2017
7	`sr:match:11733773`	Basketball		Basketball - NBA Final 2017
8	`sr:match:12953638`	Basketball	Voided_market	Basketball match with voided DrawNoBet (2nd half draw)
9	`sr:match:12233896`	Basketball	Playerprops	Basketball with player props
10	`sr:match:13682571`	Beach Volley		Beach volleyball
11	`sr:match:13530237`	Bowls		Bowls
12	`sr:match:11836360`	Cricket	Premium_cricket	Cricket - The Ashes 2017
13	`sr:match:13073610`	Cricket	Premium_cricket rain_affected	Cricket – rain affected – ODI
14	`sr:match:13497893`	CS:GO		Counter strike (CS:GO) – ESL Pro League 2018
15	`test:match:15112430`	CS:GO		Counter strike (CS:GO) – bet_cancel, rollback_bet_cancel, bet_settlement, rollback_bet_settlement, bet_settlement (different way compared to the other bet_settlement message).
16	`sr:stage:329361`	Cycling	Outrights	Race Outrights(/Short-term outrights(Tour Down Under 2018)
17	`sr:match:13451765`	Darts		Darts – PDC WC 17/18 – Final
18	`sr:match:12209528`	Dota2		Dota2 – The International 2017 – Final
19	`sr:match:12363102`	Futsal		Futsal
20	`sr:simple_tournament:66820`	Golf	Outrights 3balls winner_event dead_heat_factor	Golf – Winner Events – 3balls – Dead Heat Factor (South African Open)
21	`sr:match:12362564`	Handball		Handball Match (DHB Pokal 17/18)
22	`sr:match:11784628`	Ice hockey		Ice hockey - NHL Final 2017 (6th match)
23	`sr:match:11878140`	Ice hockey	Rollback_betcancel	Icehockey with rollback betcancel
24	`sr:match:11878386`	Ice hockey	Overtime rollback_bet_cancel aet	Icehockey with overtime + rollback_bet_cancel + match_status "aet" (after extra time)
25	`sr:match:13516251`	LoL		League of Legends (LCK Spring 2018)
26	`sr:match:12979908`	Rugby League		Rugby League
27	`sr:match:12420636`	Rugby Union		Rugby Union
28	`sr:match:13673067`	Rugby Union 7s		Rugby Union 7s
29	`sr:match:12927314`	Snooker		Snooker – Int Championship 2017 (Final best-of-19 frames)
30	`sr:match:11830662`	Soccer		Soccer English Premier League 2017 (Watfords vs Westham)
31	`sr:match:12865222`	Soccer	Overtime	Soccer Match with Overtime – Primavera Cup
32	`sr:match:12873164`	Soccer	Overtime penalty_shootout	Soccer Overtime & Penalty Shootout – KNVB beker 17/18
33	`sr:match:11958226`	Soccer	Rollback_bet_settlement	Soccer with Rollback Betsettlement from Prematch Producer

34	`sr:match:11971876`	Soccer	Aborted	Soccer match aborted mid-game – new match played later (first match considered cancelled according to betting rules)
35	`sr:match:12055466`	Soccer	Playerprops	Soccer Match w Player Props (prematch odds only)
36	`sr:match:12841530`	Squash		Squash (Quatar Classic)
37	`sr:match:12820410`	Table Tennis		Table Tennis (WC 2017 Final)
38	`sr:match:12927908`	Tennis		Tennis Match – ATP Paris Final 2017
39	`sr:match:16218476`	Tennis	Player_retires	Tennis Match where one of the players retire
40	`sr:match:13616027`	Tennis	Rollback_betcancel	Tennis Match with bet_cancel adjustments using rollback_bet_cancel
41	`sr:match:13600533`	Squash	Coverage_loss voided_market	Squash with voided markets due to temporary loss of coverage – no ability to verify results
42	`sr:season:48787`	Tennis	Outrights	Long-term(winner) outrights
43	`sr:match:12716714`	Volleyball	Bet_cancel	Volleyball
44	`sr:match:13582831`	Volleyball	Coverage_loss	Volleyball where Betradar loses coverage mid-match and has no ability to verify results
45	`wns:draw:299000867`	World Number Service	number_betting	World Number Service / Number Betting
46	`wns:draw:300736016`	World Number Service	number_betting	World Number Service – Cancelled after draw
47	`wns:draw:299000768`	World Number Service	number_betting	World Number Service – Cancelled before draw
48	`test:match:14531321`	Soccer	Overtime – correct score	Soccer with overtime – correct score, next scoring type, next goal scorer markets home and away team, halftime/fulltime, player to score 2+ and 3+, exact number of goals (AAMS)
49	`sr:match:11877526`	Ice hockey	Penalties	Ice hockey match with penalties
50	`sr:match:13625345`	Baseball	Overtime	Baseball match including overtime
51	`sr:match:13633505`	Baseball	Interrupted	Interrupted baseball match
52	`sr:match:14387858`	American football	Touchdown scorer	Touchdown scorer (including overtime market)
53	`sr:match:15408384`	Futsal		
54	`sr:match:14723771`	Ice hockey	Penalty shootout	
55	`sr:stage:364909`	Alpine Skiing	Version outrights	Top 3 and Winner outrights which include the version specifier and "other" outcome
56	`sr:match:15121494`	Field Hockey	4x15 min format	4x15 minutes per quarter Field Hockey match. Ended after regular time with a result of 4:2, including several suspensions.

5.1.1. List scenarios

Scenarios are sets of sport-events/matches that are prepared by Betradar. Instead of adding individual matches you can add a complete scenario. The sport-events in scenarios are better cached, and it is recommended that you use the scenarios for i.e. stress testing. The complete list is also available HERE, in the /scenario endpoint.

List scenarios – Lists available scenarios. Currently we provide three scenarios, two stress testing scenarios and one scenario with a mix of all different kinds of events (see THIS table for more info about individual events). The current available scenarios are:

Scenario 1) – 200 parallel matches

```
<replay_scenarios>
<replay_scenario id="1" description="200 matches parallel"
run_parallel="true">

... Match List ...

</replay_scenario>
```

Scenario 2) – 100 matches parallel

```
<replay_scenario id="2" description="100 matches parallel"
run_parallel="true">

... Match List ...

</replay_scenario>
```

Scenario 3) – Various sport events in parallel

```
<replay_scenario id="3" description="recommended sample events, 42 events
of various sports and products" run_parallel="true">

... Match List ...

</replay_scenario>
```

5.1.2. Play scenario

Queues up the sport-events in the specified scenario. Works very similar to “Start Replay”, but queues up the whole scenario and then starts the replay.

5.2. Replay Server – SDK Support

This section contains details about our Unified Feed Replay Server when using the SDK. The SDK replay server is available in both Java and .NET.

5.2.1. SDK methods

The following tables lists available methods when using the SDK for replay handling. For more detailed information about the SDK, consult our dedicated online documentation.

Table 53 - SDK replay server methods

Method name	Description
addSportEventToReplay **(**SportEvent event**)**	Add a SportEvent to the list of SportEvents whose recorded messages will be replayed. The SportEvents to add have to be older than 48hours, there is no max time, but typically Sportradar does not guarantee that SportEvents older than 30 days can be replayed.
addSportEventToReplay(string id**)**	Add a SportEvent to the list of SportEvents whose recorded messages will be replayed.

`clear()`	Stops playing recorded messages from the playlist, **and** clears the playlist.
`getPlayStatus()`	Get the current status of the replayer
`getReplayList()`	Get the list of SportEvents whose recorded messages are scheduled for replay
`play()`	Starts playing the messages for the SportEvents in the play list. The speed will be the default speed (10x faster than actual recorded speed) and with a maximum delay between events of 10 seconds (even if the actual time would have been longer).
`Play(double speedupFactor, int maxDelayInMs)`	Starts playing the messages for the SportEvents in the play list with additional arguments for speed (10x is default) and message delays. This is to avoid waiting for very long delays if two events are far apart.
`stop()`	Stops playing recorded messages from the playlist.

5.2.2. Replay server in the Java SDK

To use the replay server instead of the ordinary server, you create a ReplayOddsFeed instead of an OddsFeed. When you then create the actual session, it will be against the replay server, not against the normal server. Now you will also be able to call getReplayManager on the OddsFeedSession. This will give you access to the ReplayManager that you can use to change the playlist, start / stop the replay and check the status if a replay is ongoing. Messages are delivered as normally to your registered OddsFeedListener.

Note: No heartbeat checking is performed in a replay session.

5.2.3. Replay server in the .NET SDK

The Replay Server is accessible in the .NET SDK too. Please see the SDK specific documentation for more details.

5.3. Replay Server – API

In the following table you will find all the available API endpoints for manipulating the replay server to suit your needs (https://iodocs.betradar.com/replay#Replay-Scenarios-GET-List-scenarios).

Table 54 - Replay server API endpoints

HTTP	Endpoint-path from https://api.betradar.com/v1	Description
GET	`/replay/`	Lists all sport-events in the replay list
PUT	`/replay/events/(id)`	Add an event to the replay list
DELETE	`/replay/events/(id)`	Remove an event from the replay list
POST	`/replay/play`	Start replay (with specified parameters)
POST	`/replay/stop`	Stop replay
POST	`/replay/clear`	Stop replay (if playing) and clear playlist
GET	`/replay/status`	Get status of replay (playing or stopped)
GET	`/replay/scenario`	List scenarios
POST	`/replay/scenario/play{scenarioid}`	Start replay scenario
GET	`/sports/{language}/sport_events/{urn_type}:{id}/summary.xml`	Summary information about a replay event

5.3.1. Add an event to the replay list

Adds event {eventId} to the end of the replay queue. If there is no event in the replay queue, this event will appear on the first (and at the same time last) position of the replay queue. If there are already some events present, new event is added at the end of the queue. If {eventId} is already present in the queue before calling this endpoint and it is not at the end of queue, the event is moved to the last position in the queue and all events that were in the queue after this event are moved to 'their initial position - 1' position in the queue. If it is present and is already last, nothing will change.

If you want to start sending messages from say 5 minutes into the match, there is one extra parameter start_time that specifies how many minutes relative to match start to start sending messages from.

5.3.2. Remove an event from the replay list

If the specified event is present in the queue, it will be removed and all other affected events will change their position to initial position - 1 position. If the event is not present in the queue, nothing will change.

5.3.3. Start replay

Start replaying an event from replay queue. Events are played in the order they were played in reality, e.g. if there are some events that were played simultaneously in reality, they will be played in parallel as well on replay server. Adjust the parameters and specify the speed of the replay, and what should be the maximum delay between messages. If not specified, default speed = 10 and max_delay = 10000 are used. This means that messages will be sent 10x faster than in reality, and that if there was some delay between messages that was longer than 10 seconds it will be reduced to exactly 10 seconds/10 000 ms (this is helpful especially in pre-match odds where delay, this can be even a few hours or more).

Note The start scenario API will return a response immediately. If you try to start a new replay before replay setup is complete, you will receive this response:

```
<response>
<action>Unable start the player. Player status is currently:
SETTING_UP.</action>
<message>Player is currently fetching and preparing data for the replay.
Please wait.</message>
</response>
```

If you ask for a status **before** the replay setup is complete, you will get the following response:

```
<player_status status="SETTING_UP" last_msg_from_event="sr:match:12873164"
description="Player is currently fetching and preparing data for the
replay. Please wait." />
```

The node_id parameter: Can be used if there are multiple developers for the same client using the replay server. If node_id is set, messages will have the node_id in the routing-key, which can be used to ensure developers are not affected by each other’s replays.

The product_id parameter: Can be used to ensure that only messages from a specific producer is sent (for example if you only want to receive odds for the Live Odds producer).

The use_replay_timestamp: If you set this to true, the replay server will set the timestamp in the messages to the current time, if you set it to false the replay server will send the message with the original timestamp as it was sent.

5.3.4. Stop replay

Stop the player if it is currently playing. If player is already stopped, nothing will happen.

5.3.5. Reset replay

Stops the player if it is playing and removes all sport events from the play list (if any matches are in the list).

5.3.6. Replay status

Return the status of player. Possible values are:

- Player is playing
- Player is stopped
- Player was never playing.

5.3.7. Replay summary

There is a dedicated endpoint available on the replay server environment called */replay/sports/{language}/sport_events/{urn_type}:{id}/summary.xml*.

This endpoint is a copy of the summary endpoint for the real time API on the production environment (that returns the summary info of a sport event). However, this endpoint is used for replay matches, and will provide you with replayed API responses.

6. Unified Odds - Number betting service

Numbers betting delivers betting markets on various lotteries. The events in Numbers Betting represent a single draw of a lottery and uses event_ids like wns:draw:1234. The draws are connected to one lottery, similar to how matches are connected to a tournament. Numbers betting still uses the same concept of categories and a sport, 108. Most attributes about a draw are shared between all draws in a lottery. Only draw time, results and the match status is specific to a draw.

In order to start receiving *wns* (World Number Service) messages, you need to select what lottery types you want to receive from the ctrl feed options page.

6.1.1. Feed messages

All feed messages are generated by product 7 (wns). The format of the messages is the same as those generated by other producers (see section 3.3). The normal life cycle for a wns draw is in some rare circumstances draws are cancelled. A betstop and bet_settlement might then be sent before the draw start. The markets in the settlement will then be voided and use an appropriate void reason.

Table 55 - Feed messages from product 7

Time	Message	Notes
`Midnight before the event`	odds_change	Note that there isn't a fixture change message. In 99% of cases there is only one odds update per event.
`Minute before draw start`	bet_stop	
`Sometime after draw`	bet_settlement	

6.1.2. API – Number betting

Due to the physical differences in lottery draws compared to other sports, Numbers Betting has some different endpoints and responses. Self-service documentation of the API exists in https://iodocs.betradar.com/wns.

Table 56 - Number betting API endpoints

HTTP	Endpoint	Description
GET	`sports/(lang)/sport_events/(id)/fixture.xml`	Look up for a single draw. No additional information is found here that doesn't exist in the schedule.
GET	`sports/(lang)/sport_events/(id)/summary.xml`	Look up both the fixture and the result for a single draw. If event hasn't ended then result is omitted.
GET	`wns/(lang)/lotteries.xml`	Full list of all lotteries (tournaments) that are considered relevant
GET	`wns/(lang)/lotteries/(id)/schedule.xml`	Get the list of draws for a given lottery. Only relevant lotteries that are in the future or have recently ended are shown. This is similar to the tournament schedule endpoint in the Sports API
POST	`wns/recovery/initiate_requestt?after=(timestamp)[&request_id=(x)][&node_id=(y)]`	Perform recovery. Data is sent over the feed. See 4.6

Table 57 - Lottery attributes

Name	Description
`id`	Unique identifier of the lottery
`name`	Name of the lottery
`sport`	Sport element similar to Sports API
`category`	Category element similar to Sports API
`draw_info.draw_type`	Enum taking one of the values drum or rng. This indicates if the pick is performed using a drum or a random number generator.
`draw_info.time_type`	Enum taking one of the values fixed or interval. Interval lotteries offers frequent draws.
`draw_info.game_type`	String in the form "6/39". Where 6 indicates number of balls to be picked, while 39 is the number of balls used in the draw
`bonus_info.bonus_balls`	The number of bonus balls used in the draw
`bonus_info.bonus_drum`	Enumeration, either same or additional. If bonus drum is additional, it means that draws will be picked from a new drum, allowing the same balls to be picked from the regular draw
`bonus_range`	String in the form "1-49" indicating the range of values the bonus balls can take

Table 58 - Fixture attributes

Name	Description
`id`	Unique id of the draw
`draw_date`	Date and time of the draw in ISO 8601 format. Time is specified in UTC time.
`lottery`	Lottery element see (lottery attributes section)
`status`	Enum taking a value of either open, closed, finished, canceled. Open means that the draw has not started yet. Closed means that the lottery is being played. Finished means that we have a result and canceled indicates that the draw was aborted.
`draw_result`	Element for result if the draw is finished or canceled

`draw_result.draws.chronological`	Boolean specifying if the draws are specified in the order they were picked (chronological) or sorted by value
`draw_result.draws.draw.name`	The name of the single draw in the form "draw_3". If the lottery has bonus balls, this will be displayed as "draw_b1".
`draw_result.draws.draw.value`	The ball number of the draw.