GBE Asynchronous API
AAPI Version 2.2

Version Number:	2.2d
Author:	Trevor Bourke
Version Date:	07th Nov 2022

circulation, reverse engineering, decompilation and disassembly are prohibited in any event as to any software content.

Documentation Control

Revision History

Version	Date	Author	Comment
2.0a	28th Sept 2017	B Owens	Abridged from full specification
2.0b	4th Oct 2017	T Bourke	Updated minor details.
2.0c	11th Oct 2017	B Owens	Included Unsubscribe.
2.0d	2nd Nov 2017	Trevor Bourke	
2.0e	5th Jan 2018	Trevor Bourke	
2.1a	29th Mar 2018	Trevor Bourke	Breaking change to SubscribeEventHierarchy
2.1b	23rd April	Trevor Bourke	
2.2a	2nd May	Trevor Bourke	Moved Selection Blurb to a language independent topic
2.2b	22nd May 2018	Trevor Bourke	
2.2c	20th Aug 2019	Trevor Bourke	Abandoned – Selection Specific subscriptions
2.2d	Nov 2022	Trevor Bourke	PD-139679 Pd-141035 PD-142509 Added new topics and subscriptions to support subscribing to Selection level match information. PD-145840 (BettingType)

Version	Date	Author	Comment

Table of Contents

Introduction

The GBE Asynchronous API (AAPI) works on a subscription model whereby clients "Subscribe" to the data they are interested in. The server will then publish updates to the client as the data changes

The GBE Asynchronous API consists of messages asynchronously sent from a client to the server and messages asynchronously sent from the server to the client on a stateful connection (usually a Secure Web Socket connection (see https://tools.ietf.org/html/rfc6455#section-10.1) though stateful polling is also supported) between the client and AAPI Server. State is maintained between the client and server for the duration of the connection.

Before covering messages it is worth covering two different categories of data and how, in particular, a different approach is taken in handling these two categories.

The first category is data that either (i) a client is unlikely to be interested in being automatically notified about future changes to that data or (ii) that the cost or overhead of automatically being so notified is not worth the effort. An example of this type of information is the postings made to a punter's account during a period in the past – the set of posting made to an account during a period in the past will not change at any time in the future. This category is supported by the client requesting a copy of the data and the server sending the entire data set to the client. However, if there any changes to that data in the future, those future changes will not be notified to the client.

The second category is data that is likely to change in the future and where it is likely that the client will be interested in being automatically notified as and when that data does change. This category is supported by the client registering an interest in the data. Registering an interest in data causes both (i) the server to send the entire current state of the data to the client and (ii) the server asynchronously notifying the client of any future changes to the data. This type of data is organised as "Topics" (and the specific Topics are defined in 'Topic Hierarchy' on page 15). A Topic represents a logical grouping of data to which a client can subscribe.

Messages Send from Client to Server

Messages sent from the client are called *Commands*, and are all direct requests by the client that the server perform some action. Commands fall into one of three broad categories:

- *Session Management Commands* – these are concerned with the management of the connection and the session over that connection. They include functions like logging on and logging off.

- *Subscription Commands* – these enable the client to register an interest in data (or in terminating a previously registered interest).

- *General Application Commands* – these enable the client to request the server to perform a specific action. There is a wide variety of potential such actions, though the only one implemented is Ping. The server will normally send a message to the client in response to every command received.

Messages Send from Server to Client

Two types of messages are sent by the server to the client:

- In direct response to the client having issued a Command. These are called *Responses* and contain a return code and, in most cases, other return information as well. A command and its corresponding response can be viewed as the simulation of a synchronous request over an asynchronous medium.

- As a result of some information changing at some future time. These are called *Data Messages* and they contain only the smallest amount of information that enables the client to determine the current state of the data (that is, they must be applied to the most recent view of the data as understood by the client to yield the current state of that data).

Message Format and Encoding of Data in Commands and Responses

As stated, communication is in the form of WebSocket messages. We have defined a sub-protocol within these websocket messages. Messages consists of a header and a body. The header and body are delimited by the control character SOH (\u0001).The header consists of a series of fields common to most messages delimited by the control character STX (\u0002). The body consist of a set of name-value pairs. Each Name-Value pair within a message is delimited by the control character SOH (\u0001). The name is delimited from the value by the control character STX (\u0002). Name-value pairs for only changed attributes will be included in Data Messages but name-value pairs for all attributes will be included in all other messages (in particular the Topic Load Message which represents the current state of the Topic at the time of subscription).

Complex formatted data can be sent on Commands, Responses and Data Messages. As stated above, all of this data is encoded as simple name-value pairs. The names in name-value pairs contain either (i) the ordinal number of the attribute concerned or (ii) more complex information to enable the overall structure of the data can be understood by the receiver. More information about the information encoded into these names can be found in ‘Parameter Naming’ on page 10.

Some general points about the encoding of values:

1. Data is encoded as character strings separated by termination characters

2. All attributes are encoded as name value pairs. There will always be a name and there may be zero or one values. The absence of a value means that the attribute no longer exists.

3. The value (in name-value pairs) is always encoded as a character string (binary encodings are never used). The actual format of the character string depends on the type of data concerned:

a. Values of an enumeration are encoded as a string containing the integer assigned to the domain value concerned.

b. Booleans are encoded as the strings “T” for true and “F” for false.

c. Timestamps are encoded as a string containing UTC times using ISO 8601 combined date and time formats (e.g. “2008-12-31T23:21:14.355Z”).

d. Int and long values are encoded as a string containing decimal digits and using no more digits than are necessary (i.e. no leading 0s).

e. MoneyValue values are encoded as a string containing a fixed decimal scalar value, always consisting of two digits after the decimal point at and at least one digit before the decimal point. There are no leading 0s except in the case that the only digit appearing before the decimal point is a single 0.

“Price” is encoded as a decimal with *up to* 14 digits after the decimal point. However, no leading or trailing 0s are included.

All messages have a header containing the following fields

topicName : String(256)
: The name of the topic in the case of data messages. Empty for command and response messages

messageIdentifier : int
: The numeric identifier as specified in this document of command messages (and Command Responses)

messageType :String(1)
: Indicates if the message is a topic load, delta or delete message. A topic delete message is indicated by the string ‘X’ in the header. It is issued when a topic to which a client is subscribed is removed.
A Topic Load message is normally the first message received on subscription to a topic and contains the state of the topic at that moment in time and is indicated by the string “T”. Delta messages are those issued when a change occurs on a topic and is indicated by the string “F”

Topic Hierarchy

There are some general points about the Topic Hierarchy:

1. There are custom subscription messages that clients can send to the AAPI Server. These will subscribe the client to a range of topics of actual interest to them, for example to all market and selection information for a specific market in a specific language, currency and odds format

2. Topics will represent objects (which can have many attributes) as opposed to representing the attributes.

3. Given that (i) topics represent objects, (ii) individual attributes of objects can change independently and (iii) it is important to be able to distribute individual attributes as deltas a combination of topic and attribute names are used to uniquely identify attributes of objects within commands and responses, as defined in "Encoding of Data in Commands and Responses".

4. There are two types of topic in the topic hierarchy. The first are those with a fixed topic name, and these are mainly used to provide structure and facilitate identifying specific sub-trees. The second are those with variable topic names, where the topic name identifies the specific variable content covered by the content – for example, each market would be a topic with a unique topic name.

5. Variable topic names are named using some immutable identifier and not the English name of the underlying thing. For example, topics representing EventClassifiers are named using a non-changing identifier (that is not human-meaningful) and not names like 'Soccer' etc.

6. As a general pattern, topics will have *either* (i) child topics that have fixed names or (ii) child topics with variable names but where all those child topics are of the same type. In other words, a topic will never contain (i) a combination of children with fixed names and children with variable names or (ii) children with variable names but of different types. The objective of this general pattern is to remove ambiguity and to never require that the client has to either parse the name of a topic that has a variable name or to extract some data from a topic that has a variable name in order to establish the type of that topic.

7. The topic hierarchy represents, among other things, the event hierarchy. This is a more general hierarchy than that supported by the exchange – in particular the desire is to support a single hierarchy that could be the union of the exchange hierarchy, a number of tote hierarchies and other hierarchies (e.g. games or large roll-over pools). This leads to an extra level of indirection and complexity in the Topics used to represent EventClassifiers ('Event1'), Markets ('Market1') and Selections ('Selection1'). However, it is felt the benefits to the client of having a single unified hierarchy is worth this extra complexity.

8. Topics with variable names are generally prefixed with "E_" to indicate that these are exchange topics. This is in order to support the possible addition of topics in the future that represent data from other sources

9. All topics have a logical name and a short topic mnemonic. Descriptive names are used in all discussion and description of topics but would be unnecessarily inefficient (even with Topic Aliasing) at run-time. The mnemonic is therefore always used as the actual topic name in messages.

Parameter Naming

The hierarchical structure of parameters on messages and commands can be arbitrarily complex. There can be repeating groups of parameters and there can even be repeating groups within

repeating groups etc. Parameters on all messages are represented as simple name-value pairs. The 'name' in the name-value pairs is typically a fixed ordinal number. This works perfectly when there are no repeating groups, but how can this support repeating groups?

The answer is that the 'name' in name-value pairs contains full context information. The name consists of a number of parts, with each part identifying the value within its enclosing context, which is identified by previous parts of the name. These parts are separated by the '-' character. A repeating group itself is identified by an ordinal number, following by the 'V' character, following by the instance number of the repeating group. For example, 3V2 means the second instance of a repeating group where the repeating groups have an ordinal number of 3. Therefore, although all parameters are represented as a single list of name-value pairs the complete semantics of constructs like repeating groups, or even repeating groups within repeating groups, can be unambiguously represented and parsed.

This is best explained using an example. Assume that the parameters concerned are (and this is a fabricated example to illustrate the point, it is not the definition of parameters for any specific message or command):

requestId : long
correlationId : long
[variable] -- one for each multiple bet concerned
    wantAllOrNothing : Boolean
    [variable] -- one for each selection in the multiple bet
        selectionId : long
        selectionName : String
    [variable] -- one for each offer for the combination of selections
        stake : MoneyValue
        offerPrice : Price

Further assume that we want to represent the following sample data:

requestId = 123
correlationId = 456
-- first multiple bet
    wantAllOrNothing = 'T'
    -- first selection in first multiple bet
        selectionId = 345
        selectionName = 'First selection'
    -- second selection in first multiple bet
        selectionId = 535
        selectionName = 'Second selection'
    -- third selection in first multiple bet
        selectionId = 888
        selectionName = 'Third selection'
    -- first offer for first multiple bet
        stake = 10.00
        offerPrice = 12.5
    -- second offer for first multiple bet
        stake = 20.00
        offerPrice = 12.3
    -- third offer for first multiple bet

stake = 50.00
offerPrice = 12.1
-- fourth offer for first multiple bet
stake = 200.00
offerPrice = 12.0
-- fifth offer for first multiple bet
stake = 1000.00
offerPrice = 11.00
-- second multiple bet
wantAllOrNothing = 'F'
-- first selection in second multiple bet
selectionId = 666
selectionName = 'Ninth selection'
-- second selection in second multiple bet
selectionId = 777
selectionName = 'Tenth selection'
-- first offer for second multiple bet
stake = 50.00
offerPrice = 55.00
-- second offer for first multiple bet
stake = 80.00
offerPrice = 50.00

The ordinal numbers allocated to the original definition could be

requestId : long [1]
correlationId : long [2]
[variable] [3V]
wantAllOrNothing : Boolean [1]
[variable] [2V]
selectionId : long [1]
selectionName : String [2]
[variable][3V]
stake : MoneyValue [1]
offerPrice : Price [2]

The names of each of each parameter in the sample would be:

requestId = 123	**[1]**
correlationId = 456	**[2]**
-- first multiple bet	
wantAllOrNothing = 'T'	**[3V1-1]**
-- first selection in first multiple bet	
selectionId = 345	**[3V1-2V1-1]**
selectionName = 'First selection'	**[3V1-2V1-2]**
-- second selection in first multiple bet	
selectionId = 535	**[3V1-2V2-1]**
selectionName = 'Second selection'	**[3V1-2V2-2]**
-- third selection in first multiple bet	
selectionId = 888	**[3V1-2V3-1]**
selectionName = 'Third selection'	**[3V1-2V3-2]**

-- first offer for first multiple bet
stake = 10.00 **[3V1-3V1-1]**
offerPrice = 12.5 **[3V1-3V1-2]**
-- second offer for first multiple bet
stake = 20.00 **[3V1-3V2-1]**
offerPrice = 12.3 **[3V1-3V2-2]**
-- third offer for first multiple bet
stake = 50.00 **[3V1-3V3-1]**
offerPrice = 12.1 **[3V1-3V3-2]**
-- fourth offer for first multiple bet
stake = 200.00 **[3V1-3V4-1]**
offerPrice = 12.0 **[3V1-3V4-2]**
-- fifth offer for first multiple bet
stake = 1000.00 **[3V1-3V5-1]**
offerPrice = 11.00 **[3V1-3V5-2]**
-- second multiple bet
wantAllOrNothing = 'F' **[3V2-1]**
-- first selection in second multiple bet
selectionId = 666 **[3V2-2V1-1]**
selectionName = 'Ninth selection' **[3V2-2V1-2]**
-- second selection in second multiple bet
selectionId = 777 **[3V2-2V2-1]**
selectionName = 'Tenth selection' **[3V2-2V2-2]**
-- first offer for second multiple bet
stake = 50.00 **[3V2-3V1-1]**
offerPrice = 55.00 **[3V2-3V1-2]**
-- second offer for second multiple bet
stake = 80.00 **[3V2-3V2-1]**
offerPrice = 50.00 **[3V2-3V2-2]**

As the actual parameters consists of a simple list of name-value pairs, the actual parameters for this sample would therefore be:

Name	**Value**
1	'123'
2	'456'
3V1-1	'T'
3V1-2V1-1	'345'
3V1-2V1-2	'First selection'
3V1-2V2-1	'535'
3V1-2V2-2	'Second selection'
3V1-2V3-1	'888'
3V1-2V3-2	'Third selection'
3V1-3V1-1	'10.00'
3V1-3V1-2	'12.5'
3V1-3V2-1	'20.00'
3V1-3V2-2	'12.3'
3V1-3V3-1	'50.00'
3V1-3V3-2	'12.1'
3V1-3V4-1	'200.00'
3V1-3V4-2	'12.0'
3V1-3V5-1	'1000.00'
3V1-3V5-2	'11.00'

3V2-1	‘F’
3V2-2V1-1	‘666’
3V2-2V1-2	‘Ninth selection’
3V2-2V2-1	777’
3V2-2V2-2	‘Tenth selection’
3V2-3V1-1	‘50.00’
3V2-3V1-2	‘55.00’
3V2-3V2-1	‘80.00’
3V2-3V2-2	‘50.00’

The order in which name-value pairs appear is significant Name-value pairs for parameters within the same context *must* be contiguous to each other, but parameters within a particular context do not have to be in any specific order. For example, consider the following sub-set of the example above:

1	‘123’
2	‘456’
3V1-1	‘T’
3V1-2V1-1	‘345’
3V1-2V1-2	‘First selection’

A valid order of these nave-value pairs would be:

3V1-1	‘T’
3V1-2V1-1	‘345’
3V1-2V1-2	‘First selection’
2	‘456’
1	‘123’

Another valid order of these name-value pairs would be:

1	‘123’
2	‘456’
3V1-2V1-1	‘345’
3V1-2V1-2	‘First selection’
3V1-1	‘T’

However, an invalid order of these name-value pairs would be:

1	‘123’
2	‘456’
3V1-2V1-1	‘345’
3V1-1	‘T’
3V1-2V1-2	‘First selection’

This is invalid because the two parameters 3V1-2V1-1 and 3V1-2V1-2, which are within the same context, are not contiguous to each other.

Versioning and Interface Evolution

The AAPI will evolve over time. Every change to the AAPI is either a *breaking change* or a *non-breaking change*, and these are supported by the AAPI in different ways.

There is an explicit *AAPI version* and multiple versions of the AAPI will be supported at the same time and breaking changes are only ever introduced in new AAPI versions. The client explicitly specifies the AAPI version it supports when it creates a session (see “Session Management Commands” on page 34) and all subsequent interactions on that session will conform to that version of the AAPI. The AAPI version is a string containing the version number as identified on the cover of this document, for example “120”. Thus, because breaking changes are only introduced in new AAPI versions and that new AAPI version will not be available to a client until the client explicitly specifies that it supports that new AAPI version, existing clients will operate unchanged even when breaking changes are introduced. Clients, of course, will not be able to benefit from function introduced in new versions of the AAPI until they have been changed to explicitly support the new AAPI function.

Multiple versions of the AAPI will be supported at the same time but older versions will not be supported indefinitely. The dates after which older versions of the AAPI will no longer be supported will be clearly published in advance.

When non-breaking changes are introduced existing clients will continue to operate unchanged, provided that the client observes some rules:

1. New response parameters may be added. Clients must therefore ignore any response parameters that they do are not expecting.

2. New domain values may be added to enumerations. Clients must therefore be capable of operating correctly or at least gracefully if an unexpected enumeration domain value is encountered.

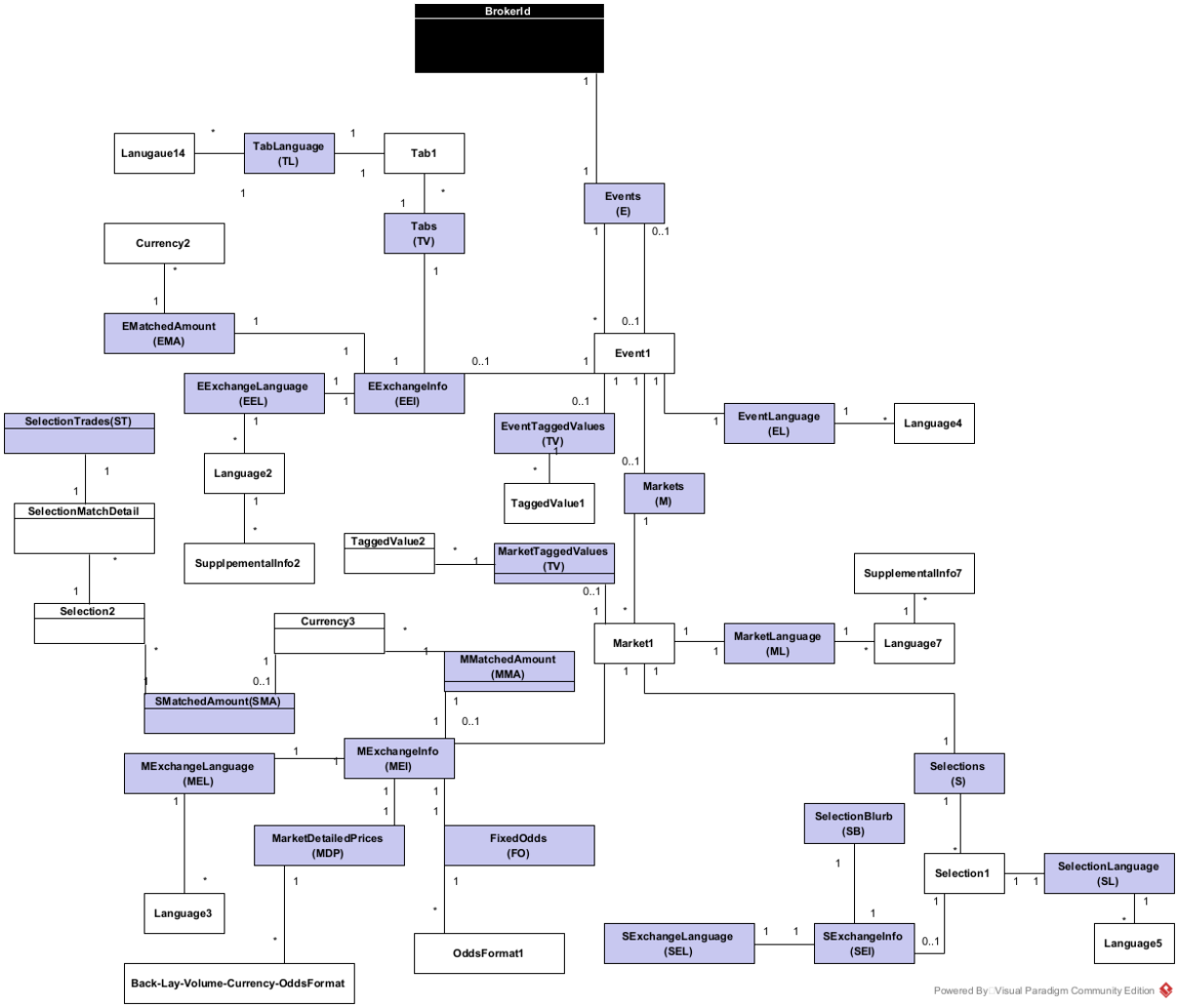
3. New optional input parameters may be added. However if a value for the new input parameter is not specified a default value will be used, and that default value will be one that most closely matches the behaviour of the system before the new input parameter was introduced.

4. Existing input parameters may be deleted. However specifying a value for non-existing input parameters will not result in an error – those values will simply be ignored.

5. New return codes can be added. Clients must therefore be capable of operating correctly if an unexpected return code is returned.

Topic Hierarchy

The topic hierarchy is illustrated in the following diagram. Topics with a black background are top level topics, those with a shaded background have fixed names and those with a white background have variable names.



Events

Description: Topic that acts as a container for instances of the variably-named topic Event1.

Topic Name: Fixed – "E"

Top Level Topic? No

Attributes: None.

Contains: Event1 (multiple)

Event1

Description: Represents a node in the event hierarchy. The event hierarchy is an abstract one. Individual nodes in the hierarchy can correspond to exchange events, tote events, both exchange and tote events or other things like games or large roll-over pools. The only role the abstract hierarchy serves is to act as an integration point for exchange events and markets and tote events and markets etc.

Topic Name: Variable. Will be of the form of either "E_" suffixed by the character representation of a unique exchange identifier (the EventClassifierId). Examples include:

"E_901644"

Top Level Topic? No

Attributes:

displayOrder : int [1]
[optionally]

Information about the current score of the event.

This is a generic concept and the way in which it is used is defined on a sport-by-sport basis. In general, though, this can be used in one of two main ways. The first is that there is a single *score* for an event, which contains the latest running score in a format defined on a sport-by-sport basis (for example "2-1" for soccer). The second is that there is a *score* for each relevant occurrence contributing to the current score, with an optional time at which that occurrence occurred. For example, a *score* for each goal scored, optionally listing the time at which the goal was scored. In this second case the client would need to calculate the actual latest score from the totality of the *score* values.

[variable] [2V]

[optionally]

occurredAt : Timestamp [1]

The time (UTC) at which the occurrence identified by *score* occurred or the time at which *score* was last updated.

score : String (1024) [2]

The actual occurrence or current score. The format of this is defined on a sport-by-sport basis.

[optionally]

Information about times of significance for the event – for example the time at which it is anticipated that the event will commence or the time at which the event actually commenced.

This is a generic concept and the way it is used is defined on a sport-by-sport basis.

[variable] [3V]

occurrenceType : String (64) [1]

A string defining the type of occurrence concerned. This is defined on a sport-by-sport basis. In soccer, for example occurrenceType could be defined as 'MatchStarted' or as 'FirstHalfStarted', 'FirstHalfEnded', 'SecondHalfStarted', 'SecondHalfEnded', 'ExtraTimeFirstHalfStarted' and 'ExtraTimeFirstHalfEnded' etc.

[either]

predictedTime : Timestamp [2]

The time (UTC) at which it is predicted that the event will occur.

[or]

actualTime : Timestamp [3]

The actual time (UTC) at which the event occurred.

Contains: Events (optional)
Markets (optional)
EventLanguage (one)
EExchangeInfo (optional)
EventTaggedValues (optional)

Language4

Description: Contains the language-specific attributes of Event1.

Topic Name: The ISO language code of the language concerned *in lowercase*.
Examples include:
"en"
"es"
"hk"
"it"
"ja"
"ml"
"ru"

Top Level Topic? No

Attributes: *name : String (256) [1]*

Contains: Nothing

EExchangeInfo

Description: Contains the exchange-specific attributes of Event1.

Topic Name: Fixed – "EEI"

Top Level Topic? No

Attributes: *eventClassifierId : long [1]*
isEnabledForMultiples : Boolean [2]
[optionally]
startTime: Timestamp [3]

Contains: EExchangeLanguage (one)
Tabs (one)

Language2

Description: Contains the language-specific attributes of EExchangeInfo.

Topic Name: The ISO language code of the language concerned *in lowercase*. Examples include:
"en"
"es"
"hk"
"it"
"ja"
"ml"
"ru"

Top Level Topic? No

Attributes: *name : String (256) [1]*
[optionally]
raceGrade : String (2048) [2]
String containing the encoded race grade and prize money information, if available.

Contains: Nothing

TaggedValue1

Description: Contains event-specific supplemental information published via a TaggedValue created on the exchange. The specifics of the content of will be agreed between the TaggedValue creator and the Topic consumer.

Topic Name: Variable. The Topic name associated with the TaggedValue in question though not necessarily the TaggedValue name.

Top Level Topic? No

Attributes: *value : String (unlimited) [1]*
The actual content.

Contains: Nothing.

Markets

Description: Topic that acts as a container for instances of the variably-named topic Market1.

Topic Name: Fixed – "M"

Top Level Topic? No

Attributes: None.

Contains: Market1 (multiple)

Market1

Description: Represents a market in the event hierarchy. This topic is quite abstract and can correspond to an exchange market, a tote market or both exchange and tote markets. The only role this topic serves is to act as an integration point for exchange markets and tote markets.

Topic Name: Variable. Will be of the form of "E_" suffixed by the character representation of a unique exchange identifier (the MarketId) . Examples include:

"E_2141742"

Top Level Topic? No

Attributes:

displayOrder : int [1]
[optionally]

Information about the current score of the market. Score information would usually be provided only at an event level but can occur at market level in some circumstances – for example, in "To score" markets.

This is a generic concept and the way in which it is used is defined on a sport-by-sport basis.

[variable] [2V]

[optionally]

occurredAt : Timestamp [1]

The time (UTC) at which the occurrence identified by *score* occurred or the time at which *score* was last updated.

score : String (1024) [2]

The actual occurrence or current score. The format of this is defined on a sport-by-sport basis.

Contains: MarketLanguage (one)
MExchangeInfo (optional)

Selections (one)
MarketTaggedValues (optional)

Language7

Description: Contains the language-specific attributes of Market1.

Topic Name: The ISO language code of the language concerned *in lowercase*.
Examples include:
"en"
"es"
"hk"
"it"
"ja"
"ml"
"ru"

Top Level Topic? No

Attributes: *name : String (256) [1]*

Contains: Nothing

MExchangeInfo

Description: Contains the exchange-specific attributes of Market1.

Topic Name: Fixed – "MEI"

Top Level Topic? No

Attributes:
marketId : long [1]
marketType: MarketType [2]
isPlayMarket : Boolean [3]
canBeInRunning : Boolean [4]
managedWhenInRunning : Boolean [5]
isVisibleAsTradingMarket : Boolean [6]
isVisibleAsPricedMarket : Boolean [7]
isEnabledForMultiples : Boolean [8]
isCurrentlyInRunning : Boolean [9]
status : MarketStatus [10]
withdrawAction : WithdrawAction [11]
ballotOutAction : BallotOutAction [12]
canBeDeadHeated : Boolean [13]
[optionally]
startTime: Timestamp [14]
Implementation note, the effective time at which it is anticipated that the market will start regardless of whether this value was specified explicitly on the market concerned or inherited from an EventClassifier.
[optionally]
delayFactor : int [15]
numberOfWinningSelections : int [16]
withdrawalSequenceNumber : int [17]

[optionally]
resultString : String (256) [18]
numberOfSelections : int [19]
The number of Selections within this for which SExchangeInfo topics will be sent. This is not necessarily the total number of Selections in the Market – for example, if topic information for withdrawn Selections would not be sent then this number should exclude withdrawn Selections.

[optionally]
bettingTypeId : BettingType [25]

Contains: MarketDetailedPrices (one)
MExchangeLanguage (one)
MMatchedAmount (one)
FixedOdds(one)

Back-Lay-Volume-Currency-Format

Description: Contains *all* current market prices for all selections within SExchangeInfo for a specific combination of (i) number of back prices, (ii) number of lay prices, (iii) market by volume, (iv) in a particular currency and (v) in a particular odds format.

Topic Name: Variable. A concatenation of:

- nn – the number of columns of back prices desired
- "_"
- nn – the number of columns of lay prices desired
- "_"
- nnnn – the market-by-volume amount desired (in the whole currency units)
- "_"
- xxx – the ISO currency code of the currency concerned *in uppercase*.
- "_"
- "1" (for decimal odds), "2" (for fractional odds) or "3" (for American odds).

Examples include:
"1_0_0_USD_1"
"3_3_100_EUR_2"
"4_2_24_USD_3"

Top Level Topic? No

Attributes: *[variable] [1V]*
selectionId : long [1]
[variable] [2V]

Prices available to be backed.
displayPrice : String(6) [1]
Containing the string that should be displayed to the user to represent this price. This will be in the odds format concerned.
[optionally]
stake : MoneyValue [2]
The amount of stake available at the price concerned and in the currency concerned. If the value is not specified (or is zero) it means that no stake is available any longer at that price.

[variable] [3V]
Prices available to be layed.
displayPrice : String(6) [1]
Containing the string that should be displayed to the user to represent this price. This will be in the odds format concerned.
[optionally]
stake : MoneyValue [2]
The amount of stake available at the price concerned and in the currency concerned. If the value is not specified (or is zero) it means that no stake is available any longer at that price.

[optionally]
redboxDisplayPrice : String(6) [4]
redboxFractionalPrice : String (unlimited) [5]

Contains: Nothing.

Language3

Description: Contains the language-specific attributes of MExchangeInfo.

Topic Name: The ISO language code of the language concerned *in lowercase*. Examples include:
"en"
"es"
"hk"
"it"
"ja"
"ml"
"ru"

Top Level Topic? No

Attributes: *name : String (256) [1]*
[optionally]

description : String (2048) [2]

Contains: Nothing

MMatchedAmount

Description: Topic that acts as a container for instances of the variably-named topic Currency3.

Topic Name: Fixed – "MMA"

Top Level Topic? No

Attributes: None.

Contains: Currency3 (multiple)

Currency3

Description: Contains the currency-specific matched volume of MExchangeInfo.

Topic Name: The ISO currency code of the currency concerned *in uppercase*.
Examples include:
"EUR"
"GBP"
"INR"
"JPY"
"NOK"
"USD "

Top Level Topic? No

Attributes: *forSideAmount : MoneyValue [1]*
The sum of for side amounts matched.
againstSideAmount : MoneyValue [2]
The sum of against side amounts matched.

Contains: SMatchedAmount.

MarketTaggedValues

Description: Topic that acts as a container for instances of the variably-named topic TaggedValues2.

Topic Name: Fixed – "TV"

Top Level Topic? No

Attributes: None.

Contains: TaggedValue2 (multiple)

TaggedValue2

Description: Contains market-specific supplemental information published via a TaggedValue created on the exchange. The specifics of the content of will be agreed between the TaggedValue creator and the Topic consumer.

Topic Name: Variable. The Topic name associated with the TaggedValue in question though not necessarily the TaggedValue name.

Top Level Topic? No

Attributes: *value : String (unlimited) [1]*
The actual content.

Selections

Description: Topic that acts as a container for instances of the variably-named topic Selection1.

Topic Name: Fixed – "S"

Top Level Topic? No

Attributes: None.

Contains: Selection1 (multiple)

Selection1

Description: Represents a selection in the event hierarchy. This topic is quite abstract and can correspond to an exchange selection, a tote selection or both exchange and tote selections. The only role this topic serves is to act as an integration point for exchange selections and tote selections. In general, all *Selection1*s within the same *Selections* will have either (i) only SExchangeInfos, (ii) only SToteInfos or (iii) both SExchangeInfos and EToteInfos.

Topic Name: Variable. Will be of the form of "E_" suffixed by the character representation of a unique exchange identifier (the SelectionId. Examples include:
"E_11422686"

Top Level Topic? No

Attributes: *displayOrder : int [1]*
[optionally]
selectionIcon : IconReference [2]

Contains: SelectionLanguage (one)
SExchangeInfo (optional)

SExchangeInfo

Description: Contains the exchange-specific attributes of Selection1.

Topic Name: Fixed – "SEI"

Top Level Topic? No

Attributes:
selectionId : long [1]
status : SelectionStatus [2]
selectionResetCount : int [3]
[optionally]
withdrawalFactor : Percentage [4]
[optionally]
settledTime : Timestamp [5]
[optionally]
resultString : String (256) [6]
voidPercentage : Percentage [7]
leftSideFactor : Percentage [8]
rightSideFactor : Percentage [9]

Contains: SExchangeLanguage (one)
SelectionBlurb (one)

Language6

Description: Contains the language-specific attributes of SExchangeInfo.

Topic Name: The ISO language code of the language concerned *in lowercase*. Examples include:
"en"
"es"
"hk"
"it"
"ja"
"ml"
"ru"

Top Level Topic? No

Attributes:
name : String (256) [1]
selectionBlurb : String (256) [2]

Contains: Nothing

SelectionBlurb

Description: Contains the language-independent Selection Blurb.

Topic Name: Fixed – "SB"

Top Level Topic? No

Attributes: *blurb : String (256) [1]*

Contains: Nothing

Language5

Description: Contains the language-specific attributes of Selection1.

Topic Name: The ISO language code of the language concerned *in lowercase*. Examples include:

"en"
"es"
"hk"
"it"
"ja"
"ml"
"ru"

Top Level Topic? No

Attributes: *name : String (256) [1]*

Contains: Nothing

Tabs

Description: Topic that acts as a container for instances of the variably-named topic Tab1.

Topic Name: Fixed – "TAB"

Top Level Topic? No

Attributes: None.

Contains: Tab1 (multiple)

Tab1

Description: Represents a grouping of markets (represented by a list of marketIds) within an EventClassifier. The set of markets in the list are all or a subset of the direct child markets of the EventClassifier in question.

Topic Name: Variable. The name of the grouping in question.

Top Level Topic? No

Attributes: *displayOrder : int [1]*
marketIds : String [2]
A string containing the marketIds contained in the tab (delimited by the '~' character).

[optionally]
numberOfMarketsToExpand: int [3]
Indicates to the UI the desirable number of markets to display

Contains: TabLanguage (one)

TabLanguage

Description: Topic that acts as a container for instances of the variably-named topic Language14.

Topic Name: Fixed – "TL"

Top Level Topic? No

Attributes: None.

Contains: Language14 (multiple)

Language14

Description: Contains the language-specific attributes of Tab1.

Topic Name: The ISO language code of the language concerned *in lowercase*.
Examples include:
"en"
"es"
"hk"
"it"
"ja"
"ml"
"ru"

Top Level Topic? No

Attributes: *name : String (256) [1]*

Contains: None

FixedOdds

Description: Topic that acts as a container for instances of the variably-named topic OddsFormat.

Topic Name: Fixed – "FO"

Top Level Topic? No

Attributes: None.

Contains: OddsFormat(multiple)

OddsFormat

Description: Contains the current fixed odds prices for all selections within MExchangeInfo for a particular odds format.

Topic Name: Variable, consisting of "D" (for decimal odds), "F" (for fractional odds) or "A" (for American odds). Examples include:

"D"
"F"
"A"

Top Level Topic? No

Attributes: *[variable] [1V]*

selectionId : long [1]
[optionally]
backFixedOddsDisplayPrice : String (6) [2]
Containing the string that should be displayed to the user to represent this price. This will be in the odds format concerned.
layFixedOddsDisplayPrice: String (6) [3]
Containing the string that should be displayed to the user to represent this price. This will be in the odds format concerned.

Contains: Nothing.

SMatchedAmount

Description: Topic that acts as a container for instances of the variably-named topic Selection2.

Topic Name: Fixed – "SMA"

Top Level Topic? No

Attributes: None.

Contains: Selection2 (multiple)

Selection2

Description: Represents the Matched Amounts on a selection in the parent topic currency (Currency3) and acts as a container for instances of the variably-named topic SelectionMatchDetail.

Topic Name: Variable – a string representation of the exchange's unique selection identifier (SelectionId). Examples include:
"11422686"
"11422687"

Top Level Topic? No

Attributes:
forSideAmount : MoneyValue [1]
The sum of for side amounts matched on this selection.
againstSideAmount : MoneyValue [2]
The sum of against side amounts matched on this selection.

Contains: SelectionMatchDetail (multiple)

SelectionMatchDetail

Description: Contains details of matching information that occurs at same contiguous price. Each change of price at which a match is occurs is represented by a unique topic. For example consider a Selection that was matched at 1.98 then 2.0 and then again at 1.98 then this would be represented by three topics.

Topic Name: Variable a string representation of Price and of the Java timestamp in milliseconds when the first match at the contiguous price occurred at. Examples include:
"1.98_1447402821007"
"2_1447402861012"
"1.98_1447415681892"

Top Level Topic? No

Attributes:
price : Price [1]
The contiguous price this topic represents.
matchedForSideAmountAtSamePrice : MoneyValue [2]
The total amount of for side stake that was matched at the same contiguous price as the last match that occurred on this Selection. For example, if $10 was matched at 1.98, then $8 at 2.0, then $5 at 1.98 and then $20 at 1.98 this value of this would be $25.
matchedAgainstSideAmountAtSamePrice : MoneyValue [3]
The total amount of against side stake that was matched at the same contiguous price as the last match that occurred on this Selection.
firstMatchAtSamePriceOccurredAt : Timestamp [4]

The time (UTC) at which the first match occurred at the same contiguous price this topic represents.

lastMatchedOccurredAt : Timestamp [5]

The time (UTC) at which the last match occurred at the same contiguous price this topic represents.

Contains: SelectionTrades (one or more)

SelectionTrades

Description: Contains the individual trades that have occurred on a particular SelectionMatchDetailTopic.

Topic Name: Variable. A concatenation of:

- "ST" –Fixed appreviation for SelectionTrades
- "_"
- nn –the Java timestamp in milliseconds when the first match at the contiguous price occurred at
- "_"
- n – An ordinal number. Used to give a unique identity to the SelectionTrades topic as there can be multiple under a single SelectionMatchDetailTopic where the number of SelectionTrade items at a contiguous price exceeds a system configure limit.

Examples include:

"ST_1447402821007_1"
"ST_1447402821007_2"

Top Level Topic? No

Attributes: *[variable] [1V]*

It is not guaranteed that the ordinal number following the "1V" will be contiguous, but it will indicate the time sequence in which the trades occurred (with older trades having smaller numbers). These ordinal numbers are guaranteed (i) to be unique within any selection (ii) to be constant over different invocations of ITL and delta messages for the same selection. They are not guaranteed to be contiguous for any selection but it is guaranteed that for any two trades on the same selection that the one with the smaller ordinal number will have occurred earlier.

occurredAt : Timestamp [1]

price : Price [2]

backersStake : MoneyValue [3]

The backer's stake of the trade concerned in the currency concerned.

layersLiability : MoneyValue [4]

The layer's liability of the trade concerned in the currency concerned.

tradeType : TradeType [5]
An indication of whether this trade was the result of a 'for' order taking an available 'against' price or vice versa.

Contains: Nothing.

Connections, Commands and Messages

All communication from the client to the AAPI server is performed by the client issuing *commands* This includes subscribing to topics (which can only be performed by issuing custom subscription commands) and authentication. All communication from the AAPI server to the client is performed through the server sending *messages*.

Authentication is performed explicitly though application commands AAPI connections can be in one of a number of application-specific states and there are explicit commands used to transition the connection from one state to another (the *Session Management Commands*). Most commands can only be issued when the connection is in a specific state and the states that a connection must be in are specified in the definition of each command below.

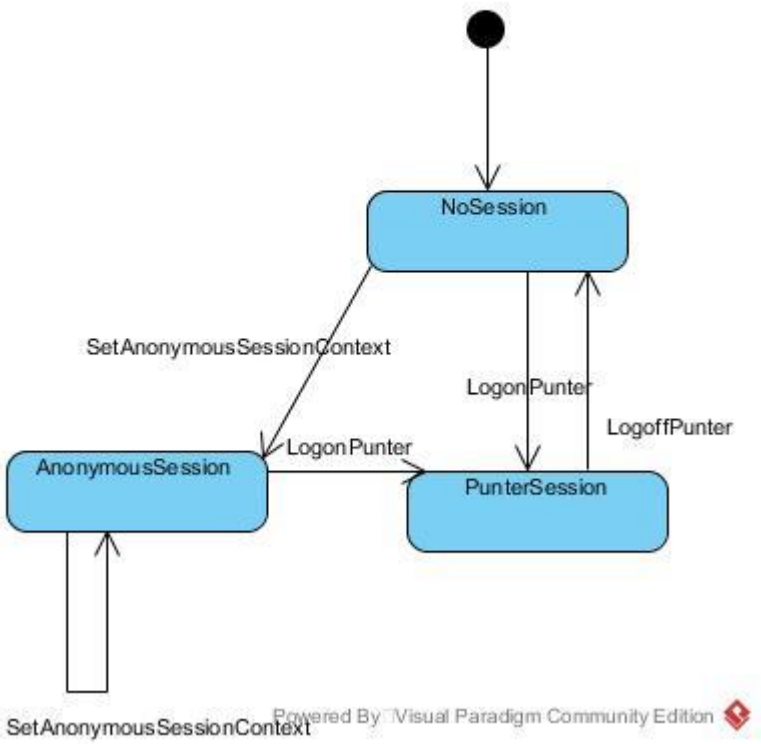
These states are:

- *NoSession* – the AAPI connection has been created but no application-specific state information has yet been specified and so the connection cannot be used for any commands other than those specifying application-specific context information (that is, the Session Management Commands). In other words, this represents a connection that cannot yet be used for commands other than the Session Management Commands.

- *AnonymousSession* – application-specific session information has been specified for an anonymous (not logged-on) punter. In other words, this represents a connection that can be used for any application commands that do not require a logged-in punter.

- *PunterSession* – application-specific session information has been specified and the punter concerned has been authenticated. In other words, this connection represents a logged-on punter.

These states and the commands that can be used to transition between those states are illustrated in the following diagram.



Most commands result in a special response message (called *responses*) being sent to the client in response to the command having been issued. Responses are used to communicate information to a client in direct response to the command. *Responses* contain return code information and / or output parameters. Responses are only ever sent if the command concerned was issued when the connection was in one of the states defined as "Valid States" for the command concerned – commands issued when the connection is in any other state are simply ignored with no response being sent.

When a client subscribes to topics (by issuing a custom subscription command) the server responds by asynchronously sending one or more messages containing the current value of the relevant topics in addition to the server sending a response message for the custom subscription command concerned. The server subsequently sends delta messages as and when any of the information changes. These messages containing topic data are called *data messages*.

Commands contain (i) common parameters and (ii) parameters that are specific to the type of command concerned. The parameters that are specific to the type of command concerned are specified in the definition of each command below. The common parameters are:

- *correlationId : long*
  A number specific by the client. Any subsequent responses (but not data messages) sent as a direct result of this command will contain this *correlationId* value. This provides the client

with a way to unambiguously correlate response messages with the command concerned. The correlationId input and output parameters are not explicitly specified in the definition of each command below, but they are always present in both input and output parameters and have an ordinal number of 0 in all cases.

If an attempt is made to invoke an AAPI that doesn't exist then *RC658 AAPIDoesNotExist* will be returned.

Session Management Commands

SetAnonymousSessionContext (1)

Description: Set or update the context applicable to an anonymous session (change the state of the session into AnonymousSession).

Input Parameters:

currency : String [1]
If the currency specified is not a play currency then all subsequent interaction on the Session that is established will be for non-play markets but if the currency specified is a play currency than all subsequence interaction on the Session will be for play markets.
language : String [2]
priceFormat : PriceFormat [3]
[optionally]
integrationPartnerId : long [5]
aAPIVersion : String (8) [6]
The version of the interface that the client supports. See “Versioning and Interface Evolution” on page 14 for further information.
clientSpecifiedGuid : GUID [7]
GUID created by the client to uniquely identify an execution of the client. Will be ignored if a clientSpecifiedGuid was previously specified on the connection concerned (that is, if a clientSpecifiedGuid was specified on a previous SetAnonymousSessionContext, LogonPunter or LogonTelebetUser command on the same connection).
granularChannelType : GranularChannelType [8]
channelInformation : String (256) [9]
[optionally]
clientIdentifier : String (64) [10]
A string defined by the client implementation used to identify the client (and version) of the client software concerned. This has no semantic significance to the operation of the system. The intent of this is facilitate the analysis of usage of different client types and for problem resolution.

Valid States: NoSession
AnonymousSession

Response Parameters:

maximumMessageSize : long [2]
The maximum physical message size allowed. See “Long Messages and Chunking” for more information about chunked messages.
[optionally]
maximumMarketInformationMarketsCount : int [3]

Specified when a user is restricted to a maximum number of markets that can concurrently be subscribed for Market Information.

maximumMarketPricesMarketsCount : int [4]
Specified when a user is restricted to a maximum number of markets that can concurrently be subscribed for Market Prices.

maximumMarketMatchedAmountsMarketsCount : int [5]
Specified when a user is restricted to a maximum number of markets that can concurrently be subscribed for Market Matched Amounts.

maximumMarketFixedOddsPricesMarketsCount : int [6]
Specified when a user is restricted to a maximum number of markets that can concurrently be subscribed for Fixed Odds.

Return Codes:

RC000	Success
RC001	ResourceError
RC002	SystemError
RC023	CurrencyNotValid
RC071	LanguageDoesNotExist
RC105	CurrencyDoesNotExist
RC113	ParameterFormatError
RC134	ParameterMissingError
RC308	IncorrectVersionNumber
RC504	IntegrationPartnerDoesNotExist
RC531	DeprecatedAPIVersion
RC672	ConnectionInInvalidState
RC673	PunterNotAuthorisedForAAPI
RC701	AAPINotSupported

Subscribes To: N/A.

LogonPunter (2)

Description: Log a punter onto the session (change the state of the session into PunterSession).

If the (AnonymousSession) connection has any active subscriptions then those subscriptions will remain active provided that the *currency*, *language*, *priceFormat* and *wantPlayMarkets* values for the Punter concerned are the same as those in effect for the AnonymousSession. If the values of any of these properties are different then all subscriptions currently active for this connection will be terminated.

Input Parameters: *[either]*

partnerToken : PartnerToken [1]

[or]

aAPISessionToken : AAPISessionToken [2]

The *aAPIsessionToken* that was returned by a previous LogonPunter command.

[or]

[optionally]

integrationPartnerId : long [3]

partnerUsername : String (64) [4]

cleartextPassword : String (256) [5]

As the password is clear text the connection between client and server must be over TLS.

currency : Currency [6]

This is only referenced if a punter with the *partnerUsername* specified does not exist and that punter will be auto-registered – in which case the currency of that punter will be this *currency*. It is ignored in all other cases.

[optionally]

language : String (2) [7]

This defines the language for the session.. If not specified the Punter's default language is used.

[or]

integrationPartnerId : long [13]

arbitrarySessionInformation : String (unlimited) [12]

[or]

integrationPartnerId : long [14]

sessionToken : SessionToken [15]

aAPIVersion : String (8) [8]

The version of the interface that the client supports. See "Versioning and Interface Evolution" on page 14 for further information.

clientSpecifiedGuid : GUID [9]

GUID created by the client to uniquely identify an execution of the client. Will be ignored if a clientSpecifiedGuid was previously specified on the connection concerned (that is, if a clientSpecifiedGuid was specified on a previous SetAnonymousSessionContext, LogonPunter command on the same connection).

granularChannelType : GranularChannelType [10]

channelInformation : String (256) [12]

[optionally]

clientIdentifier : String (64) [13]

A string defined by the client implementation used to identify the client (and version) of the client software concerned. This has no semantic significance to the operation of the system. The intent of this is facilitate the analysis of usage of different client types and for problem resolution.

Valid States: NoSession
AnonymousSession

Response Parameters:
debitSportsbookStake : Boolean [2]
debitExchangeStake : Boolean [3]
purseIntegrationMode : PurseIntegrationMode [4]
canPlaceForSideOrders : Boolean [5]
canPlaceAgainstSideOrders : Boolean [6]
restrictedToFillKillOrders : Boolean [7]
currency : String [8]
language : String [9]
priceFormat : PriceFormat [10]
marketByVolumeAmount : MoneyValue [11]
aAPISessionToken : AAPISessionToken [13]
Representing the authenticated session. If the connection to AAPI Server gets dropped for any reason this *aAPISessionToken* can be presented on a subsequent LogonPunter command as opposing to having to specify a PartnerToken (which may have expired in the meantime) or a *cleartextPassword* on the subsequence LogonPunter command.
maximumMessageSize : long [14]
The maximum physical message size allowed
[optionally]
maximumMarketInformationMarketsCount : int [15]
Specified when a user is restricted to a maximum number of markets that can concurrently be subscribed for Market Information.
maximumMarketPricesMarketsCount : int [16]
Specified when a user is restricted to a maximum number of markets that can concurrently be subscribed for Market Prices.
maximumMarketMatchedAmountsMarketsCount : int [17]
Specified when a user is restricted to a maximum number of markets that can concurrently be subscribed for Market Matched Amounts.
maximumMarketFixedOddsPricesMarketsCount : int [18]
Specified when a user is restricted to a maximum number of markets that can concurrently be subscribed for Fixed Odds.

Return Codes:

RC000	Success
RC001	ResourceError
RC002	SystemError
RC113	ParameterFormatError
RC134	ParameterMissingError
RC208	PunterSuspended
RC308	IncorrectVersionNumber

RC437 UnacceptableIPAddress
RC500 PunterNotRegisteredToIntegrationPartner
RC504 IntegrationPartnerDoesNotExist
RC511 PartnerTokenNotAuthenticated
RC512 SessionTokenNotAuthenticated
RC513 PunterIntegrationPartnerMismatch
RC514 SessionTokenNoLongerValid
RC518 UsernameDoesNotExist
RC521 PasswordAuthenticationNotAllowed
RC531 DeprecatedAPIVersion
RC612 PunterNotAuthenticated
RC671 ConcurrentSessionLimitReached
RC672 ConnectionInInvalidState
RC675 PunterIsBanned
RC701 AAPINotSupported

Subscribes To: N/A.

LogoffPunter (3)

Description: Log a punter off (change the state of the session into NoSession).
Additionally, all subscriptions currently active for this connection will be terminated.

Input Parameters: N/A.

Valid States: PunterSession

Response Parameters: N/A.

Return Codes:
RC000 Success
RC001 ResourceError
RC002 SystemError
RC113 ParameterFormatError
RC134 ParameterMissingError
RC672 ConnectionInInvalidState
RC701 AAPINotSupported

Subscribes To: N/A.

SetRefreshPeriod (60)

Description: Set the refresh period for the connection.
All delta data messages can be sent as soon as they arise. An alternative is for changes that occur during a refresh period to be batched together with only the net effect of all those changes being sent

to the client. Having such a refresh period can dramatically reduce the amount of bandwidth needed, which can be important for mobile clients. For example, consider an extreme case where the liquidity on a back price on a selection changed 200 times during a 2 second period. If no refresh period is in effect then 200 data messages would be sent to the client. However, if a refresh period with a period of 2000ms was in effect then just a single delta data message would be sent to the client resulting in a net saving of 99.5% of bandwidth usage.

Not all delta data messages are batched into refresh periods – some delta data messages are sent as soon as they arise regardless of the refresh period. In general, delta data messages that are critical are always sent as soon as they arise. Information about specific orders and the status of markets (for example if the market is suspended) fall into this category. Delta data messages that are batched into refresh periods include market prices, matched amounts and selection P&Ls. Responses to commands explicitly issued (including initial topic loads) are never subjected to a refresh period.

The refresh period applies to all subscriptions on a single connection. For example, if a connection has subscribed to prices and matched amounts on 3 different markets and if a refresh period of 3000ms is in effect for that connection then all price and matched amount changes that occurred on any of the three markets are sent at the same time at the end of the refresh period.

Minimum refresh periods can apply to clients. If a minimum refresh period applies to a client and if the client attempts to set the refresh period to a value less than the minimum refresh period applicable to that client the refresh period will be set to the minimum refresh period concerned. The effective refresh period in effect after this command has been processed is returned as a response parameter.

Input Parameters:

refreshPeriodMS : int [1]

The refresh period requested (in milliseconds). A value of 0 means that no refresh period will apply.

Valid States: All states.

Response Parameters:

refreshPeriodMS : int [2]

The refresh period in effect after this command has been processed. This will be the *refreshPeriodMS* requested unless the requested *refreshPeriodMS* is less than the minimum refresh period for the client.

Return Codes:

RC000	Success
RC001	ResourceError
RC002	SystemError
RC113	ParameterFormatError
RC134	ParameterMissingError
RC406	PunterIsBlacklisted
RC672	ConnectionInInvalidState
RC673	PunterNotAuthorisedForAAPI

RC701 AAPINotSupported

Subscribes To: N/A.

GetRefreshPeriod (61)

Description: Get the fresh period in effect for the client.

Input Parameters: N/A.

Valid States: All states.

Response Parameters: *refreshPeriodMS : int [2]*

Return Codes:
RC000 Success
RC001 ResourceError
RC002 SystemError
RC113 ParameterFormatError
RC134 ParameterMissingError
RC406 PunterIsBlacklisted
RC672 ConnectionInInvalidState
RC673 PunterNotAuthorisedForAAPI
RC701 AAPINotSupported

Subscribes To: N/A.

Custom Subscription Commands

Custom subscription commands enable the client to subscribe to a range of topics of interest, for example to all market and selection information for a specific market in a specific language, currency and odds format.

Custom subscription commands enable the client to become subscribed to a set of topics These custom subscription commands are the only mechanism through which normal clients can get subscribed to topics.

The response for each custom subscription command contains a unique *subscriptionId* for the subscription just created. The subscription remains in place until the *subscriptionId* is explicitly specified on a subsequent Unsubscribe command. A client may be directly or indirectly subscripted to the same topic more than once. Regardless of the number of subscriptions concerned only a single data message for each change will be sent to the client. However delta data messages will continue to be sent to the client until Unsubscribe commands have been explicitly issued for all *subscriptionIds* covering the topic concerned. A client may get a list of all its current subscriptions using the ListSubscriptions command.

Custom subscription commands can also be used to only obtain an initial topic load without actually subscribing the client to the topics concerned. This enables the client to more tightly control the use of bandwidth, which may be desirable for mobile or other low bandwidth clients.

The custom subscription requests are defined in this section.

SubscribeMarketInformation (9)

Description: Get and optionally subscribe to general market information for one or more markets specified. This does not include any price-related information.

Input Parameters:

[either]

eventClassifierId : long [2]

[optionally]

[either]

marketTypesToExclude : String [3]

A string containing the marketTypes concerned (delimited by the '~' character).

[or]

marketTypesToInclude : String [4]

A string containing the marketTypes concerned (delimited by the '~' character).

wantDirectDescendantsOnly : Boolean [5]

[or]

marketIds : String [6]

A string containing the marketIds concerned (delimited by the '~' character).

fetchOnly : Boolean [7]

Deprecated flag which must always be set to false.*[optionally]*

wantSelectionInformation : Boolean [8]
If true /BrokerId/Events/Event1/Markets/Market1/Selections topic and its descendants are included in the subscription. If not specified defaults to true, ie Selection topics are included.

[optionally]
wantExchangeLangugeInformationOnly : Boolean [9]
If specified and true then only Exchange Language topics are included in the subscription, ie other Language Related topics are excluded. Specifically /BrokerId/Events/Event1/Markets/Market1/MExchangeInfo /MExchangeLanguage/Language3 and optionally /BrokerId/Events/Event1/Markets/Market1/Selections/Selection1 /SexchangeInfo/SExchangeLanguage/Language6 topics are included. If *excludeLanguageTopics* is true then this property has no effect

[optionally]
marketTaggedValueTopicNames :String [10]
A string containing a list of Market level TaggedValue Topic Node names to which to subscribe. (delimited by the '~' character)

[optionally]
excludeLanguageTopics :Boolean [11]
If true all language specific topics are excluded from the subscription.

wantSelectionBlurb :Boolean [12]
If true and wantSelectionInformation is true then caller will be subscribed to SelectionBlurb topics

Valid States:
AnonymousSession
PunterSession

Response Parameters:

[optionally]
subscriptionId : long [2]

[optionally]
marketIds : String [3]
A string containing the list of any of marketIds (delimited by the '~' character) explicitly specified as input parameter that are not currently active.

[optionally]
availableMarketsCount : int [4]
When a user is restricted to a maximum number of concurrent Market Information Subscriptions this parameter will indicate the current number of available markets for this user.

Return Codes:

RC000	Success
RC001	ResourceError
RC002	SystemError
RC005	EventClassifierDoesNotExist
RC113	ParameterFormatError
RC134	ParameterMissingError
RC406	PunterIsBlacklisted
RC672	ConnectionInInvalidState

RC673 PunterNotAuthorisedForAAPI
RC701 AAPINotSupported
RC961 MaximumSubscribedMarketsReached

Subscribes To:

- /BrokerId/Events/Event1/Markets/Market1
- /BrokerId/Events/Event1/Markets/Market1/MExchangeInfo

[optionally]

- /BrokerId/Events/Event1/Markets/ Market1/MExchangeInfo/MExchangeLanguage/Language3

[optionally]

- /BrokerId/Events/Event1/Markets/Market1/MarketLanguage/Language7

[optionally]

- /BrokerId/Events/Event1/Markets/Market1/Selections/Selection1
- /BrokerId/Events/Event1/Markets/ Market1/Selections/Selection1/SExchangeInfo
- /BrokerId/Events/Event1/Markets/ Market1/Selections/Selection1/SExchangeInfo/SExchangeLanguage/Language6

[optionally]

- /BrokerId/Events/Event1/Markets/ Market1/Selections/Selection1/SelectionLanguage/Language5

[optionally]

- /BrokerId/Events/Event1/Markets/ Market1/MarketTaggedValues/TaggedValue2

SubscribeDetailedMarketPrices (10)

Description: Get and optionally subscribe to detailed price information for one or more markets specified.

Input Parameters:

[either]
eventClassifierId : long [1]
[optionally]
[either]
marketTypesToExclude : String [2]
A string containing the marketTypes concerned (delimited by the '~' character).
[or]
marketTypesToInclude : String [3]
A string containing the marketTypes concerned (delimited by the '~' character).
wantDirectDescendantsOnly : Boolean [4]
[or]
marketIds : String [5]
A string containing the marketIds concerned (delimited by the '~' character).
numberBackPrices : int [6]
numberLayPrices : int [7]
filterByVolume : MoneyValue [8]
Should be whole currency unit – any cent values will be ignored.
fetchOnly : Boolean [11]
Deprecated flag which must always be set to false.

Valid States: AnonymousSession
PunterSession

Response Parameters:

[optionally]
subscriptionId : long [2]

[optionally]
marketIds : String [3]
A string containing the list of any of marketIds (delimited by the '~' character) explicitly specified as input parameter that are not currently active.
[optionally]
availableMarketsCount : int [4]
When a user is restricted to a maximum number of concurrent Market prices subscriptions this parameter will indicate the current number of available markets for this user.

Return Codes:
RC000 Success
RC001 ResourceError
RC002 SystemError
RC005 EventClassifierDoesNotExist
RC113 ParameterFormatError
RC134 ParameterMissingError
RC406 PunterIsBlacklisted
RC672 ConnectionInInvalidState
RC673 PunterNotAuthorisedForAAPI
RC701 AAPINotSupported
RC961 MaximumSubscribedMarketsReached

Subscribes To:

- /BrokerId/Events/Event1/Markets/Market1/MExchangeInfo/MarketDetailedPrices/Back-Lay-Volume-Currency-OddsFormat

SubscribeFixedOddsPrices (11)

Description: Subscribe to fixed odds price information for one or more markets specified. Not that this command is only available to designated clients

Input Parameters:
[either]
eventClassifierId : long [1]
[optionally]
[either]
marketTypesToExclude : String [2]
A string containing the marketTypes concerned (delimited by the '~' character).
[or]
marketTypesToInclude : String [3]
A string containing the marketTypes concerned (delimited by the '~' character).
wantDirectDescendantsOnly : Boolean [4]
[or]
marketIds : String [5]
A string containing the marketIds concerned (delimited by the '~' character).

Valid States: AnonymousSession

PunterSession

Response Parameters:

[optionally]
subscriptionId : long [2]
[optionally]
marketIds : String [3]
A string containing the list of any of marketIds (delimited by the '~' character) explicitly specified as input parameter that are not currently active.
[optionally]
availableMarketsCount : int [4]
When a user is restricted to a maximum number of concurrent Fixed Odds prices subscriptions this parameter will indicate the current number of available markets for this user.

Return Codes:

RC000 Success
RC001 ResourceError
RC002 SystemError
RC005 EventClassifierDoesNotExist
RC113 ParameterFormatError
RC134 ParameterMissingError
RC406 PunterIsBlacklisted
RC672 ConnectionInInvalidState
RC673 PunterNotAuthorisedForAAPI
RC701 AAPINotSupported
RC961 MaximumSubscribedMarketsReached

Subscribes To:

- /BrokerId/Events/Event1/Markets/Market1/MExchangeInfo/FixedOdds/OddsFormat

SubscribeEventHierarchy (12)

Description: Get and optionally subscribe to event hierarchy information.
The event hierarchy information under a specific event classifier can be requested. Additionally only event and market information or all information including selection information can be requested.

Input Parameters:

eventClassifierId : long [2]
The id for the Event under which the event hierarchy is required. The Root EventClassifier(1) may be specified only where the wantDirectDescendantsOnly parameter is specified as true
wantDirectDescendantsOnly : Boolean [3]
wantSelectionInformation : Boolean [4]
fetchOnly : Boolean [5]
Deprecated flag which must always be set to false. *[optionally]*
[either]
marketTypesToExclude : String [6]

A string containing the marketTypes concerned (delimited by the '~' character).

[or]

marketTypesToInclude : String [7]

A string containing the marketTypes concerned (delimited by the '~' character).

[optionally]

wantExchangeLangugeInformationOnly : Boolean [8]

If specified and true then only Exchange Language topics are included in the subscription, ie other Language Related topics are excluded. Specifically /BrokerId/Events/Event1/Markets/Market1 /MExchangeInfo/MExchangeLanguage/Language3 and /BrokerId/Events/Event1/EExchangeInfo/EExchangeLanguage /Language2 and optionally /BrokerId/Events/Event1/Markets /Market1/Selections/Selection1/SexchangeInfo/SExchangeLanguag e /Language6 topics are included.

eventTaggedValueTopicNames : String [9]

A string containing a list of Event level TaggedValue Topic Node names to which to subscribe. (delimited by the '~' character).

marketTaggedValueTopicNames : String [10]

A string containing a list of Market level TaggedValue Topic Node names to which to subscribe. (delimited by the '~' character)

ecxludeMarketInformation : Boolean [11]

wantTabInformation : Boolean [12]

Indicates whether to include Tabs topics.

excludeLanguageTopics :Boolean [13]

If true all language specific topics are excluded from the subscription.

wantSelectionBlurb :Boolean [14]

If true and wantSelectionInformation is true then caller will be subscribed to SelectionBlurb topics

[optionally]

languages : String [15]

Allows a subscription to multiple language topics. If specified it will overwrite the language associate with the AAPI session. The list is delimited by the '~' character.

[optionally]

wantPlayAndRealMarkets : Boolean [16]

Allows a subscription to include both Play and real markets. If specified it will ignore the designation (real or play) associated with the AAPI Session currency.

Valid States: AnonymousSession
PunterSession

Response *[optionally]*

Parameters: *subscriptionId : long [2]*
[optionally]
availableMarketsCount : int [4]
When a user is restricted to a maximum number of concurrent Market Information subscriptions this parameter will indicate the current number of available markets for this user.

Return Codes:
RC000 Success
RC001 ResourceError
RC002 SystemError
RC005 EventClassifierDoesNotExist
RC071 LanguageDoesNotExist
RC113 ParameterFormatError
RC134 ParameterMissingError
RC406 PunterIsBlacklisted
RC672 ConnectionInInvalidState
RC673 PunterNotAuthorisedForAAPI
RC701 AAPINotSupported
RC961 MaximumSubscribedMarketsReached

Subscribes To:

- /BrokerId/Events/Event1
- /BrokerId/Events/Event1/EExchangeInfo
- *[optionally]*
  - /BrokerId/Events/Event1/EventLanguage/Language4
  - /BrokerId/Events/Event1/EExchangeInfo/EExchangeLanguage/Language2

[optionally]

- /BrokerId/Events/Event1/Markets/Market1
- /BrokerId/Events/Event1/Markets/Market1/MExchangeInfo
- /BrokerId/Events/Event1/Markets/Market1/MExchangeInfo/MExchangeLanguage/Language3

[optionally]

- /BrokerId/Events/Event1/EventLanguage/Language4
- /BrokerId/Events/Event1/Markets/Market1/MarketLanguage/Language7

[optionally]

- /BrokerId/Events/Event1/Markets/Market1/Selections/Selection1
- /BrokerId/Events/Event1/Markets/Market1/Selections/Selection1/SExchangeLanguage/Language6
- /BrokerId/Events/Event1/Markets/Market1/Selections/Selection1/SexchangeInfo

[optionally]

- /BrokerId/Events/Event1/Markets/Market1/Selections/Selection1/SelectionLanguage/Language5

[optionally]

- /BrokerId/Events/Event1/EventTaggedValues/TaggedValue1
- /BrokerId/Events/Event1/Markets/Market1/MarektTaggedValues/TaggedValue2

[optionally]

- /BrokerId/Events/Event1/ EExchangeInfo /Tabs/Tab1
- /BrokerId/Events/Event1/ EExchangeInfo /Tabs/Tab1/TabLanguage/Language14

SubscribeSelectionMatchedAmounts (13)

Description: Get and optionally subscribe to the matched volumes for all selections for one or more markets specified.

Input Parameters: *marketIds : String [1]*
A string containing the marketIds concerned (delimited by the '~' character).

includeSelectionMatchDetail : Boolean [2]
[optionally]
detailFrom : Timestamp [3]
Controls the SelectionMatchedDetails topics for historical matches that are included in the subscription. The caller is always subscribed to the SelectionMatchedDetails topic containing the latest match on this selection regardless of the value of *detailFrom*. Other SelectionMatchedDetails topics that contain matches that occurred after the *detailFrom* will also be included in the subscription.

fetchOnly : Boolean [4]
Deprecated flag which must always be set to false.

Valid States:
AnonymousSession
PunterSession
TelebetPunterSession

Response Parameters:
[optionally]
subscriptionId : long [2]
[optionally]
marketIds : String [3]
A string containing the list of any of marketIds (delimited by the '~' character) explicitly specified as input parameter that are not currently active.

Return Codes:

RC000	Success
RC001	ResourceError
RC002	SystemError
RC005	EventClassifierDoesNotExist
RC113	ParameterFormatError
RC134	ParameterMissingError
RC406	PunterIsBlacklisted
RC672	ConnectionInInvalidState
RC673	PunterNotAuthorisedForAAPI
RC701	AAPINotSupported

Subscribes To:

- /BrokerId/Events/Event1/Markets/Market1/MExchangeInfo/MMatchedAmount/Currency3/SMatchedAmount/Selection2 *[optionally]*
  - /BrokerId/Events/Event1/Markets/Market1/MExchangeInfo/MMatchedAmount/Currency3/SMatchedAmount/Selection2/SelectionMatchDetail

SubscribeMarketMatchedAmounts (14)

Description: Get and optionally subscribe to the matched volumes for one or more markets specified.

Input Parameters:
[either]
eventClassifierId : long [1]
[optionally]

[either]
marketTypesToExclude : String [2]
A string containing the marketTypes concerned (delimited by the '~' character).
[or]
marketTypesToInclude : String [3]
A string containing the marketTypes concerned (delimited by the '~' character).
wantDirectDescendantsOnly: Boolean [4]
[or]
marketIds : String [5]
A string containing the marketIds concerned (delimited by the '~' character).
fetchOnly : Boolean [7]
Deprecated flag which must always be set to false.

Valid States: AnonymousSession
PunterSession

Response Parameters:
[optionally]
subscriptionId : long [2]
[optionally]
marketIds : String [3]
A string containing the list of any of marketIds (delimited by the '~' character) explicitly specified as input parameter that are not currently active.
[optionally]
availableMarketsCount : int [4]
When a user is restricted to a maximum number of concurrent Market Matched Amount subscriptions this parameter will indicate the current number of available markets for this user.

Return Codes:
RC000 Success
RC001 ResourceError
RC002 SystemError
RC005 EventClassifierDoesNotExist
RC113 ParameterFormatError
RC134 ParameterMissingError
RC406 PunterIsBlacklisted
RC672 ConnectionInInvalidState
RC673 PunterNotAuthorisedForAAPI
RC701 AAPINotSupported
RC961 MaximumSubscribedMarketsReached

Subscribes To:

- /BrokerId/Events/Event1/Markets/Market1/MExchangeInfo/MMatchedAmount/Currency3

SubscribeSelectionTrades (19)

Description: Get and optionally subscribe to the all trades occurring at the same contiguous price as the last price matched or all trades from the *tradesFrom* input parameter whichever is earlier.

Input Parameters:

marketIds : String [1]
A string containing the marketIds concerned (delimited by the '~' character).

[optionally]

tradesFrom : Timestamp [2]
Controls SelectionTrades topics for historical trades that are included in the subscription. The caller is always subscribed to the SelectionTrades topic containing the latest trade on this selection regardless of the value of TradesFrom. Other SelectionTrades topics that contain selectionTrades that occurred after the tradesFrom will also be included in the subscription.

*fetchOnly : Boolean [3]*Deprecated flag which must always be set to false.

Valid States: AnonymousSession
PunterSession

Response Parameters:

[optionally]

subscriptionId : long [2]

[optionally]

marketIds : String [3]
A string containing the list of any of marketIds (delimited by the '~' character) explicitly specified as input parameter that are not currently active.

Return Codes:

RC000	Success
RC001	ResourceError
RC002	SystemError
RC113	ParameterFormatError
RC134	ParameterMissingError
RC406	PunterIsBlacklisted
RC672	ConnectionInInvalidState
RC673	PunterNotAuthorisedForAAPI
RC701	AAPINotSupported

Subscribes To:

- /BrokerId/Events/Event1/Markets/Market1/MExchangeInfo/MMatchedAmount/Currency3/SMatchedAmount/Selection2/SelectionMatchDetail/SelectionTrades

Unsubscribe (20)

Description: Unsubscribe from the subscriptions explicitly specified.

Input *[optionally]*

Parameters: *subscriptionIds : String [1]*
A string containing the subscriptionIds concerned (delimited by the '~' character). If no subscriptionIds are explicitly specified then all current subscriptions will be unsubscribed.

Valid States: AnonymousSession
PunterSession

Response Parameters: *[optionally]*
subscriptionIds : String [3]
A list of any of subscriptionIds explicitly specified as input parameter that are not currently active.

Return Codes:

RC000	Success
RC001	ResourceError
RC002	SystemError
RC113	ParameterFormatError
RC134	ParameterMissingError
RC406	PunterIsBlacklisted
RC672	ConnectionInInvalidState
RC673	PunterNotAuthorisedForAAPI
RC701	AAPINotSupported

Subscribes To: N/A.

General Application Commands

A response will be sent to all commands. This response will always contain a return code. If the return code is *RC000 Success* then the response will also contain parameters defined in the "Response" section of the command definitions below. If a command is received when the connection is in a state other than those defined in the "Valid States" section of the definition of the relevant command then no response at all will be sent to the client.

Ping (22)

Description: Ping the server application.

This provides the client with a way to measure the total current round-tip time between issuing a command to the server and receiving the response from the server. This command is serviced by the server application in the same way that any other command is serviced for the punter concerned and so gives a very accurate indication of the total round-trip time (that is, communications delays and server processing delays).

Input parameters enable the caller to specify the ping values experienced the previous time the Ping command was called. This enables the server to maintain a history of the ping round-trip times for each specific client.

Input Parameters: *[optionally]*

currentClientTime : Timestamp [1]

The current time (in UTC) on the client. This is required to enable the server to calculate the time drift between the client and server so that the value specified by *lastPingedAt* below can be converted into server time.

lastPingRoundtripMS : long [2]

The total round-trip time (in MS) of the last Ping command issued.

lastPingedAt : Timestamp [3]

The time (as measured in UTC by the client) at which the last Ping command was issued.

Valid States: All states.

Response Parameters: *messagesInQueue : int [2]*

The number of messages in the client's queue when the Ping command was executed. This number should be zero or very close to it. If it is a larger number it indicates that messages are being created for the client at a faster rate than that at which the client can receive them and the client should either set a refresh period (or a larger refresh period) or unsubscribe from some topics.

This count does not include delta messages that will be batched together at the end of a refresh period (see "SetRefreshPeriod (60)" for more details).

Return Codes:

RC000	Success
RC001	ResourceError
RC002	SystemError
RC113	ParameterFormatError
RC134	ParameterMissingError
RC406	PunterIsBlacklisted
RC672	ConnectionInInvalidState
RC673	PunterNotAuthorisedForAAPI
RC701	AAPINotSupported

Subscribes To: N/A.