

L_{nter} **T**_{ote} **S**_{ystem} **P**_{rotocol}

Version
05.18

Table of Contents

Syntax	3
Protocol.....	7
Header	9
Link	14
Configuration	16
Race Status.....	20
Pools	25
Scan Request Begin	28
Scans.....	30
Totals	35
Payoffs	36
Results.....	40
Alert.....	41
Will Pay.....	42
Ascii File Transfer.....	44
Appendix A - Variations for Italy.....	46

Syntax

<protocol>	:= <transmission> <transmission><protocol>
<transmission>	:= <stx><message><etx>
<message>	:= <header><checksum>[<data><checksum>]
<checksum>	:= {<octal>*5}
<header>	:= <message type><data type><source><sequence><event code><race number> <pool code> <reason><time><length>
<message type>	:= <P> <R> <D> <A>
<data type>	:= <C> <P> <X> <W> <T> <\$> <S> <R> <A> <L> <F>
<source>	:= {<alpha>*3}
<sequence>	:= {<digit>*2}
<event code>	:= {<alpha>*3} <no event>
<no event>	:= {<period>*3}
<race number>	:= {<digit>*2} <no race>
<no race>	:= {<period>*2}
<pool code>	:= {<alpha>*3}
<reason>	:= <blank> <e> <f> <h> <r> <H> <D> <R> <P> <C> <S> <I> <A> <T> <X> <?>
<length>	:= {<digit>*4}
<data>	:= <link> <configuration> <pool> <scan request begin> <scan> <pool total> <payoff> <race status> <results> <alert> <will pay> <file transfer>
<link>	:= <identifier>[<text>]
<identifier>	:= <document><version>
<document>	:= {<alpha>*4}
<version>	:= <version number><period><revision number>
<version number>	:= {<digit>*2}
<revision number>	:= {<digit>*2}
<text>	:= {<alpha>*l}
<configuration>	:= <mode><calculation><date><time><performance> <close bet delay><close cancel delay><race list><pools>{<pool configuration>*p}
<mode>	:= <H> <R>
<calculation>	:= <S> <N>
<performance>	:= {<digit>*4}
<close bet delay>	:= {<digit>*2}
<close cancel delay>	:= {<digit>*2}
<race list>	:= <races>{<race>*r}
<races>	:= {<digit>*2}
<race>	:= {<digit>*2}
<pools>	:= {<digit>*2}
<pool configuration>	:= <pool code><exchange part><scan mode> <send gross pool><receive gross pool> <send net pool><receive net pool> <send total><receive total><xfer will pay><pool unit> <min payoff><break><commissions>{<commission rate>*c}
<exchange part>	:= <alpha>
<send gross pool>	:= <when>
<receive gross pool>	:= <when>

<send net pool> := <when>
 <receive net pool> := <when>
 <send total> := <when>
 <receive total> := <when>
 <xfer will pay> := <when>
 <when> := <N>|<F>|<A>
 <pool unit> := <amount>
 <min payoff> := <amount>
 <break> := <amount>
 <commissions> := <digit>
 <commission rate> := <rate><rounding>
 <rate> := <amount>
 <rounding> := <U>|<D>|<R>

<race status> := <status><post time><display><runners>{<runner status>*r}
 <brackets>{<bracket status>*b}
 <pools>{<pool definition>*p}<scratch pools>{<scratch info>*s}

<status> := <O>|<C>|<X>|<P>|<F>|<U>
 <post time> := <time>|<no post>
 <no post> := {<period>*6}
 <display> := <M>|<C>|<P>|<F>
 <brackets> := {<digit>*2}|<no brackets>
 <no brackets> := {<period>*2}>
 <bracket status> := <numeric range>
 <runners> := {<digit>*2}|<no runners>
 <no runners> := {<period>*2}
 <runner status> := <digit>|<scratch>
 <scratch> := <alpha>
 <pools> := {<digit>*2}|<no pools>
 <no pools> := {<period>*2}
 <pool definition> := <pool code><pool status><pool race list><min bet><net addin><gross addin>
 <pool code> := {<alpha>*3}
 <pool status> := <O>|<C>|<X>|<E>
 <pool race list> := {<race>*l}
 <race> := {<digit>*2}
 <min bet> := <amount>
 <net addin> := <amount>|<no addin>
 <gross addin> := <amount>|<no addin>
 <no addin> := <period>
 <scratch pools> := {<digit>*2}|<no pools>
 <scratch info> := <pool code><race><pool runners>
 <pool runners> := {<numeric range>*n}

<pool> := <pool header>[<pool data>]
 <pool header> := <pool mode><segments><segment>
 <pool mode> := <G>|<N>
 <segments> := {<digit>*2}
 <segment> := {<digit>*2}
 <pool data> := <rows><columns>{<row data>*r}<segment total><total><net total>
 <rows> := {<digit>*2}
 <columns> := {<digit>*2}
 <row data> := {<amount>*c}<row total>
 <row total> := <amount>

<segment total>	:= <amount>
<total>	:= <amount>
<net total>	:= <amount>
<scan request begin>	:= <scan mode><scan runner list><leg>
<scan mode>	:= <E> <L> <S> <X> <Q> <A> <C> <K> <P>
<scan runner list>	:= <combos>{<scan runners>*c}<period><live runners><period> <favorite runners><period><sub runners>
<leg>	:= {<digit>*2}
<combos>	:= <digit><digit>
<scan runners>	:= <runner list>
<live runners>	:= <runner list>
<favorite runners>	:= <runner list>
<sub runners>	:= <runner list>
<scan>	:= [<scan mode><scan runner list>]<scan header>[<pool data><scan total><live total>]
<scan header>	:= <leg><pool header>
<scan total>	:= <amount>
<pool total>	:= <live total><net total>
<live total>	:= <amount>
<payoff>	:= <live runners><refund><carry over><commissions>{<commission>*c} <prices>{<price definition>*p}
<live runners>	:= <runner list>
<refund>	:= <liability>
<carry over>	:= <amount>
<commissions>	:= <digit>
<commission>	:= <amount>
<prices>	:= {<digit>*2}
<price definition>	:= <status><winners><runner list><price><winning><liability><break amount>
<status>	:= <R> <E> <\$> <P> <A> <C> <Q> <r>
<winners>	:= {<digit>*2}
<runner list>	:= {<numeric range>*r}
<numeric range>	:= [<numeric span>[{{<, ><numeric span>}...}]</>
<numeric span>	:= <number> <number><-><number>
<number>	:= <digit> <digit><number>
<price>	:= <signed amount>
<winning>	:= <amount>
<liability>	:= <signed amount>
<break amount>	:= <signed amount>
<results>	:= <finishers>{<finish>*f}<favorite list>
<finishers>	:= {<digit>*2}
<finish>	:= <runner><entry><position>
<runner>	:= {<digit>*2}
<entry>	:= <alpha>
<position>	:= <digit>*2
<favorite list>	:= {<numeric range>}
<alert>	:= [<alert type><text>]
<alert type>	:= <alpha>

<will pay>	:= <rows><columns>[<will pay row>*r]
<will pay row>	:= <runner list>{<will pay item>*c}
<will pay item>	:= <winners><price><winning>
<file transfer>	:= <file header> <file header><file data>
<file header>	:= <destination><filename><modification date><file size><file segments> <current segment><current segment size>
<destination>	:= <source> .H .R.
<filename>	:= {<alpha>*32}
<modification date>	:= <date><time>
<file size>	:= <whole amount>
<file segments>	:= <whole amount>
<current segment>	:= <whole amount>
<current segment size>	:= <whole amount>
<whole amount>	:= <amount> with restriction that amount may not have fractional portion
<file data>	:= {<ascii>*css}
<running_checksum>	:= {<octal>*5}
<ascii>	:= Standard ASCII codes from 20 ₁₆ - 7F ₁₆ , 0A ₁₆ (new line) for line termination, and 0C ₁₆ (form feed)
<date>	:= <month><day><year>
<month>	:= {<digit>*2}
<day>	:= {<digit>*2}
<year>	:= {<digit>*4}
<time>	:= <hours><minutes><seconds>
<hours>	:= {<digit>*2}
<minutes>	:= {<digit>*2}
<seconds>	:= {<digit>*2}
<signed amount>	:= <sign><amount>
<sign>	:= <+> <->
<amount>	:= {<digit>*d ₁ }<last digit> {<digit>*d ₁ }{<decimal digit>*d ₂ }<last decimal>
<last digit>	:= digit code+10 ₁₆
<decimal digit>	:= digit code+20 ₁₆
<last decimal>	:= decimal digit code+10 ₁₆
<stx>	:= 02 ₁₆
<etx>	:= 03 ₁₆
<blank>	:= 20 ₁₆
<period>	:= 2E ₁₆
<alpha>	:= Standard ASCII codes from 20 ₁₆ to 7F ₁₆ .
<digit>	:= Standard ASCII codes for the digits '0' thru '9'.
<octal>	:= Standard ASCII codes for the digits '0' thru '7'

Protocol

Syntax

```

<protocol>           := <transmission>|<transmission><protocol>
<transmission>      := <stx><message><etx>
<message>           := <header><checksum>[<data><checksum>]
<checksum>          := {<octal>*5}

```

Semantics

1. The protocol syntax is symmetric with respect to host and remote systems, permitting the same messages to be used by either system to request or send data.
2. The host system controls the link. Host control is implemented by imposing the following five rules on the protocol.
 - 2.1. During 'normal' operations the only message sequences to be initiated by the remote system are:
 - A link message with reason *Remote* to indicate remote system startup
 - Pool pending messages at post time indicating that final pools are available.
 - Pool pending final messages for individual pools. Under unusual conditions, the remote may need to schedule a pool for early closing.
 - A scan pending message whenever the remote system is ready to transfer a scan to the host
 - Operator alerts
 - Configuration and race card messages due to remote system initiated changes

The remote may only initiate a message when the link is "idle" (i.e. a regular or extended message sequence is not in progress).

- 2.2. If a negative acknowledge occurs at any point in a message sequence, the sequence in progress is terminated. If the message was initiated by the remote system, it is no longer pending on the remote system. Any and all retries will be initiated by the host. With the exception of a race status message with reason *End*, the host is not required to retry any sequence which has failed.
- 2.3. If the host fails to respond to remote system initiated message, (i.e. a time-out occurred on the remote system), the message remains pending on the remote system and may be reinitiated by the remote system subject to normal collision resolution rules and retry and time-out limitations. This is the only condition under which the remote system initiates retries.
- 2.4. Retries are considered to be totally independent message sequence's, which are not connected in any way to the sequence which caused the retry to be initiated.

Inter-Tote System Protocol

- 2.5. A collision occurs if a system receives a response to a message that does not have the same sequence number or is not the expected response to a previously transmitted message. If a collision occurs between host and remote system initiated messages, the remote system will terminate its sequence and honor the host's message. The remote system message remains pending and may be reinitiated by the remote system upon completion of the host sequence. The host system will ignore a received message that generates a collision except if the received message is a link message. The host system will delay at least 1 second after completing a message sequence in which a collision occurs before initiating a new message sequence to prevent the remote system from being locked out indefinitely by host message traffic.
3. The host system will guarantee that a transmission occurs at least once every 15 seconds. If no transmission would have occurred during a 15 second interval a test message will be sent to permit monitoring of the communications line by the host and remote systems.
4. The link is considered to have failed if the host or remote system is unable to send a message after the specified number of retries, where each transmission failed due to a checksum error or timeout. The remote may also sense a link failure if it does not receive a transmission within twice the time specified in the preceding paragraph. Once either system has sensed a link failure it must enter its link connect state to reestablish data base integrity on the remote system.
5. The definition of retry limits and time outs are determined based on implementation considerations and operational experience, with retries ranging from 1 to 3 and time-outs in the range of 5 to 30 seconds.
6. The checksum is the sum of all characters in the portion of the message with which it is associated modulo 100000₈. Except for local logging of an error, messages with header checksum errors will be treated as though no message was received. This will cause error recovery for header checksum errors to occur through message timeout.
7. It should be noted that although there is no explicit flow control, flow control can be established by delaying the acknowledge (within the limits of the time-out) associated with every message until the acknowledging system is prepared to accept another transmission. The only situation in which this discipline breaks down is a host initiated sequence following a host acknowledge to a remote system initiated message. The receiving system must allow for back-to-back transmission of an acknowledge followed by any message without the loss of the second message. In the event the remote system were to miss a message under these conditions, normal retries by the host would reestablish synchronization.
8. Unless otherwise noted, all characters are considered case sensitive.
9. Unless otherwise noted, all **<amount>** fields are in dollars. Either d₁ or d₂ or both in the **<amount>** field may be zero.

Digit	Last Digit	Decimal Digit	Last Decimal
0	@	P	`
1	A	Q	a
2	B	R	b
3	C	S	c
4	D	T	d
5	E	U	e
6	F	V	f
7	G	W	g
8	H	X	h
9	I	Y	i

10. A **<numeric range>** is used to define a list of numbers that may be put in ascending order with no repetition. Sequential spans of numbers are separated by a dash, and numbers and spans are separated by commas. All values must be in ascending order. The whole range may be empty, consisting of just the terminating '/'. For example, '1-8/' means numbers 1 2 3 4 5 6 7 8, while '1-6,8-10/' means 1 2 3 4 5 6 8 9 10.

Header

Syntax

<header>	:= <message type><data type><source><sequence><event code><race number> <pool code><reason><time><length>
<message type>	:= <P> <R> <D> <A>
<data type>	:= <C> <P> <X> <W> <T> <\$> <S> <R> <A> <L> <F>
<source>	:= {<alpha>*3}
<sequence>	:= {<digit>*2}
<event code>	:= {<alpha>*3}
<race number>	:= {<digit>*2} <no race>
<no race>	:= {<period>*2}
<pool code>	:= {<alpha>*3}
<reason>	:= <blank> <e> <f> <h> <r> <H> <D> <R> <P> <C> <S> <I> <A> <T> <X> <?>
<time>	:= <hours><minutes><seconds>
<length>	:= {<digit>*4}

Semantics

- The valid <message type> values are:

P - Pending
R - Request
D - Data
A - Acknowledge

Except as otherwise noted in the protocol, messages of <message type> *Pending* and *Request* do not have a data section and messages of <message type> *Data* must have a data section.

- The valid <data type> values and their characteristics are:

<u>Data Type</u>	<u>Host Message Types</u>	<u>Remote Message Types</u>	<u>Message Sequence</u>	<u>Direction Sent</u>
C - Configuration	P,R,D,A	P,R,D,A	Full	Both Ways
P - Pools	P,R,D,A	P,R,D,A	Full	Both Ways
X - Scan	P,R,D,A	P,R,D,A	Full	Both Ways
W- Will Pay	P,D	R,A	Full	To Remote
T - Pool Total	R,D,A	D,A	Short	Both Ways
\$ - Payoffs	D	A	Short	To Remote
S - Race Status	R,D,A	D,A	Short	Both Ways
R - Results	D	A	Short	To Remote
A - Alert	D,A	D,A	Short	Both Ways
L - Link	D,A	D,A	Short	Both Ways
F - Ascii File Transfer	P,R,D,A	P,R,D,A	Full	Both Ways

Full and Short message sequences are described in this section, below. The full or short message sequence will always be used as specified, unless an exception is noted in the message description for a data type.

Inter-Tote System Protocol

3. The **<source>** field is a unique source code for the system sending the message. These codes are the sending system's node id in the network, and should be constant across all events being processed by the system.
4. The **<sequence>** field is the message sequence number for this message sequence and is controlled by the system originating the sequence. The host system will use even numbers and the remote system will use odd numbers modulo 100₁₀ in increasing order as sequence numbers. All messages which are part of the same message sequence will have the same sequence number. Retries and messages sequences that are parts of an extended message sequence are independent message sequences and will have differing sequence numbers.

A sequence error is the defining condition for a message collision and must be handled under the collision resolution rules.

5. The **<event code>** field indicates the wagering event to which this message applies. The host system **<event code>** is controlling.
6. The **<race number>** field indicates the race to which this message applies. If the message being sent is not race dependent the **<race number>** field must be **<no race>**.
7. The valid **<pool code>** values are:

<u>Code</u>	<u>Name</u>	<u>Transfer</u>	<u>Description</u>
...	No Pool	None	No pool is specified.
WIN	Win	Single Leg	Choose the runner to finish first in one race.
PLC	Place	Single Leg	Choose the runner to finish first or second in one race.
SHW	Show	Single Leg	Choose the runner to finish first, second, or third in one race.
DD	Daily Double	Double Leg	Choose the runner to finish first in two designated races.
EX	Exacta/Perfecta	Double Leg	Choose the runners to finish first and second in exact order in one race.
QU	Quinella	Double Leg Symmetric	Choose the runners to finish first and second in either order in one race.
TRI	Trifecta	Triple Leg	Choose the runners to finish first, second, and third in exact order in one race.
SPR	Superfecta	Late Scan	Choose the runners to finish first, second, third, and fourth in exact order in one race.
BP	Big Perfecta	Double Leg	Choose the runners to finish first and second in exact order for two designated races with an exchange to the second race.

Inter-Tote System Protocol

BQ	Big Quinella	Double Leg Symmetric	Choose the runners to finish first and second in either order for two designated races with an exchange to the second race.
QD	Quinella Double	Double Leg Symmetric Early Scan	Choose runners to finish first and second in either order, in each of two designated races.
DE	Double Exacta	Double Leg Early Scan	Choose runners to finish first and second in exact order, in each of two designated races.
ITE	Italian Double Exacta	Double Leg Pool / Late Scan	(Duplice Accoppiata in order) Choose the runners to finish first and second in exact order with an exchange to a designated second race, which could have either an ITE or an ITQ .
ITQ	Italian Double Quinella	Double Leg Symmetric Pool / Late Scan	(Duplice Accoppiata not in order) Choose the runners to finish first and second in either order with an exchange to a designated second race, which could have either an ITE or an ITQ .
TT	Twin Trifecta	Triple Leg	Choose the runners to finish first, second, and third in exact order for two designated races with an exchange to the second race.
TS	Tri Super	Triple Leg/Late Scan	Choose the runners to finish first, second, and third in exact order for the first of two designated races with an exchange to the second race in which the runners are chosen to finish first, second, third, and fourth in exact order.
SS	Twin Superfecta	Late Scan/Late Scan	Choose the runners to finish first, second, third, and fourth in exact order for two designated races with an exchange to the second race.
P03	Pick Three	Triple Leg	Choose the runners to finish first in three designated races.
Pnn	Pick-N	Early Scan	Choose the runners to finish in a designated finish for nn designated races. The supported values for nn are from 4 through 12.
ETS	Ex-Tri-Spr	Double/Triple/Late Scan	Choose the runners to finish first and second in exact order in the first of three designated races with an exchange to the second race to choose the runners to finish first, second, and third in exact order with another exchange to the third race to choose the runners to finish first, second, third, and fourth in exact order.
OMN	Omni	Double Leg Symmetric	Choose two runners to finish first, second, or third in any order in one race.

Inter-Tote System Protocol

Enn	Exact-N	Late Scan	Choose the runners to finish first through <i>nnth</i> in exact order in one race. The supported values for <i>nn</i> are 5 through 12.
PPT	Perfecta-Perfecta- Trifecta	Late Scan	Choose runners to finish first and second in exact order in each of two designated races, and to finish first, second and third in exact order in a third race.
***	All Pools	Pending Pool Final	Indicates that the remote has all pools available for transfer as final.

The **<pool code>** *No Pool* should be used in all messages which are not pool specific.

A **<pool code>** of *All Pools* may only occur in a message type of *Pool Pending Final* from a remote system. See the *Pools* and *Scans* message descriptions for details about final pool and scan transfers.

8. The valid **<reason>** values for all message types are:

- No Reason
- b - Begin
- e - End
- f - Final
- h - Host
- r - Remote

If these reason codes occur in a message type of *acknowledge* they indicate that the data was accepted by the receiving system.

9. In addition to the above reason codes, the following reason codes are valid only for a message type of *acknowledge*:

- | | |
|--------------------|------------------------|
| H - Invalid Header | I - Inappropriate |
| D - Data Checksum | A - Not Available |
| R - Invalid Race | T - Invalid Event |
| P - Invalid Pool | X - Terminate sequence |
| C - Race Closed | ? - Format Error |
| S - Bad Total | |

All of these reason codes indicate a 'negative' acknowledge and that the message being acknowledged was rejected by the receiving system. The reason code *Invalid Header* is reserved for all header errors for which there is not a specific error code which do not involve a header checksum.

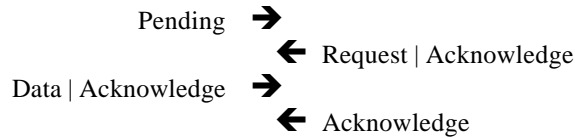
10. **<time>** is the current universal time, sometimes referred to as GMT or Zulu time.

11. **<length>** is the total length of the data section of the message including the checksum. A length of zero indicates that no data or data checksum is present.

12. The **<data type>**, **<event code>**, **<race number>** and **<pool code>** fields of a message of message type *acknowledge* must be the same as those in the header of the received message that is being acknowledged if these fields are not in error.

13. A message of message type *acknowledge* with a reason code indicating message acceptance has no **<data>** section. A message of message type *acknowledge* with a reason code indicating message rejection may have an optional **<data>** section of type **<text>**. The *text* section may be used to provide additional clarifying information concerning the error condition being reported. In general, negative acknowledgements during message sequences, such as the initialization sequence, should only be use on errors which should terminate the sequence.

14. A full transfer sequence for sending data to a system consists of the following message types:

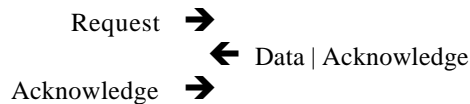


An 'negative' Acknowledge message causes the sequence to be terminated.

15. A short transfer sequence for sending data to a system consists of the following message types:



16. A request sequence for requesting data from a system consists of the following message types:



A 'negative' Acknowledge message causes the sequence to be terminated. An unsolicited request sequence may be initiated only by the host. A remote system may send the *request* message type only as part of the full transfer sequence shown above.

17. A sequence which does not involve a collision may only be terminated by a negative acknowledge.

Syntax

<link>	:= <identifier> [<text>]
<identifier>	:= <document> <version>
<document>	:= { <alpha> *4}
<version>	:= <version number> <period> <revision number>
<version number>	:= { <digit> *2}
<revision number>	:= { <digit> *2}
<text>	:= { <alpha> *l}

Semantics

1. The **<identifier>** field is the implementation ID for the sending system and may be used by the receiving system to check compatibility of sent and received messages.
2. The **<document>** is the name of the document used for implementation. This document is represented by the name "ITSP".
3. The **<version>** field specifies the version number and revision number of the implemented document. The current version of this document is "05.18". Where "05" is the **<version number>** and "18" is the **<revision number>**. Any edit revision codes appended to the revision number are not included in the **<revision number>** field.
4. All **<link>** messages will contain a reason code set to the system operating mode of the sending system. Acknowledgements of link messages shall contain a reason code set to the acknowledging system's operating mode, except for the link end message and corresponding ack. Upon receiving a link message the receiving system must terminate any single and/or extended message sequences which are in progress, acknowledge the message and enter its link connect state.
5. A link acknowledge message will have a data section containing the **<identifier>** field of the acknowledging system and optionally a **<text>** field.
6. On startup and after a link failure a system must enter a link connect state in which it issues a **<link>** message on a periodic basis until it either receives a positive acknowledge or sends a positive acknowledge to a **<link>** message from the other system. All other messages received while a system is in this link connect state should be disregarded by the receiving system.
7. The host system will send initialization information after either receiving an acknowledge to its **<link>** message or sending an acknowledge to a remote system initiated **<link>** message. The initialization information will consist of the following messages:
 - 7.1 A **<configuration>** message exchange.
 - 7.2 Race status message exchanges for each race which the remote system is betting as indicated by the **<race list>** field in the remote system's configuration which must be a subset of the host's **<race list>**.
 - 7.3 A payoff sequence for each race for which the host has official prices, and the remote does not according to the **<status>** in the race status messages received from the remote.

Inter-Tote System Protocol

- 7.4 Merged pools for the current race if the pools are final, and the race is not official according to the **<status>** in the race status messages received from the remote.
- 7.5 A **<will pay>** message for the next-to-last race of a pool type, if appropriate. See the **<will pay>** message description.
- 7.6 A **<link>** message with **<reason>** of *End* to terminate initialization which indicates that the link is ready.
8. The host must guarantee that the remote system's data base is current when initialization is complete. Thus, if host system data base updates occur while initialization is in progress and the modified information has already been transmitted, it must be retransmitted before terminating initialization.
9. In order to support communications testing before a system is ready to enter the link connect state, a pre-connect state is defined. A system in pre-connect state may only send a link message, or respond to one. In these messages, the reason code will be the operational mode of the sending system, with an event code of **<no event>**. A system that is in connect state upon receipt of a pre-connect link message will respond with an acknowledge with the proper event code. While in a pre-connect state, a host system will satisfy the normal protocol time out requirements on message transfers.
10. A system must not enter a link connect state until its data base for the event is completely setup and verified to prevent initialization from occurring with incorrect, incomplete or inconsistent information.
11. The host or remote system may issue a **<link>** message with **<reason>** of *Final* to indicate that it is going offline. The receiving system will acknowledge the **<link>** message with **<reason>** of *Final*. After receiving a shutdown message the systems will cease communicating until a **<link>** message is received from the system initiating the shutdown. All non final pool data from a shutdown remote system, including data received prior to shutdown will not be included in the merged pools by the host. Therefore, a shutdown message should not be issued by a remote system unless its non-final pools will not be used for the current race.
12. The *l* in the **<text>** definition is the number of characters contained in the text and may range from 1 to 80. The value of *l* is obtained from the **<length>** field less the size of the **<identifier>** and **<checksum>** fields. The message may consist of any ASCII text, but may not include any control characters.

Configuration

Syntax

<configuration>	:= <mode><calculation><date><time><performance> <close bet delay><close cancel delay><race list><pools>{<pool configuration>*p}
<mode>	:= <H> <R>
<calculation>	:= <S> <N>
<date>	:= <month><day><year>
<month>	:= {<digit>*2}
<day>	:= {<digit>*2}
<year>	:= {<digit>*4}
<time>	:= <hours><minutes><seconds>
<hours>	:= {<digit>*2}
<minutes>	:= {<digit>*2}
<seconds>	:= {<digit>*2}
<performance>	:= {<digit>*4}
<close bet delay>	:= {<digit>*2}
<close cancel delay>	:= {<digit>*2}
<race list>	:= <racess>{<race>*r}
<racess>	:= {<digit>*2}
<race>	:= {<digit>*2}
<pools>	:= {<digit>*2}
<pool configuration>	:= <pool code><exchange part><scan mode> <send gross pool><receive gross pool> <send net pool><receive net pool><send total><receive total><xfer will pay> <pool unit><min payoff><break><commissions>{commission rate}*c}
<pool code>	:= {<alpha>*3}
<exchange part>	:= <alpha>
<scan mode>	:= <E> <L> <S> <X> <Q> <A> <C> <K> <P>
<send gross pool>	:= <when>
<receive gross pool>	:= <when>
<send net pool>	:= <when>
<receive net pool>	:= <when>
<send total>	:= <when>
<receive total>	:= <when>
<xfer will pay>	:= <when>
<when>	:= <N> <F> <A>
<pool unit>	:= <amount>
<min payoff>	:= <amount>
<break>	:= <amount>
<commissions>	:= <digit>
<commission rate>	:= <rate><rounding>
<rate>	:= <amount>
<rounding>	:= <U> <D> <R>

Semantics

1. The **<mode>** field indicates the operational mode of the sending system. The valid **<mode>** values are:

H - Host
R - Remote

2. The **<calculation>** field indicates the type of price calculation being used. The valid **<calculation>** values are:

S - Standard single commission calculations
N - 'Net price' multi-commission calculations

The **<calculation>** value sent by a remote system must match the host's value.

3. The **<date>** field is the date on which the performance starts based on universal time.
4. The **<time>** field is the performance start time in universal time. Times in the data portion of other messages which are "less" than the performance start time refer to the day after the performance start date. The host performance start time is a constant value that is earlier than the earliest connect time for the host.
5. The **<performance>** field is the performance number. It should be noted that this field is informational only, and the host and remote system performances numbers may not be the same.
6. The **<close bet delay>** field is the delay in seconds of the sending system from the steward's key initiation (or the remote system's receipt of the post time event) to the time that the pool is closed for betting. The **<close cancel delay>** field is the delay in seconds from the close of betting to the time that the pool is also closed for cancelling.
7. The *r* in the **<race list>** field is the number of races in the list determined by the **<races>** field.
8. The **<race list>** field is the list of races in the performance. The host will send all races in the racecard that the remote may participate. The remote will send a subset of the host's races that define the races the remote will participate. If a remote participates in a pool, the remote must participate in all races of the pool. Any races missing by the remote from the host's list will cause the host to exclude all information for that race. No messages will be sent or requested for a non-participating race. Messages that are not race specific will be transmitted. Race specific messages include:
 - Pools
 - Scan
 - Pool Totals
 - Payoff
 - Race Status
 - Results
 - Will Pay
9. The *p* in the **<configuration>** message is the number of pools being sent and is determined by the **<pools>** field. The host system must send all valid pools to the remote system. As a minimum this includes all pools present in the race card, plus any pools which may be added during the course of the performance. If a remote receives an unrecognized pool or a pool that is not contained in its configuration, the pool must be accepted but not included in the remote's pool definition. The remote system must send a subset of the host's pools. If a pool is not present in a remote system configuration, the host will not consider the system as a source for that particular pool.
10. The **<send gross pool>**, **<send net pool>**, and **<send total>** fields indicate when this particular gross or net pool, or pool total will be sent by the system.

11. The **<receive gross pool>**, **<receive net pool>**, and **<receive total>** fields indicate when the system expects to receive the gross or net pool, or pool total.
12. The valid **<when>** values are:
 - N - Never
 - F - Final
 - A - All cycles

If a system is not processing a pool, then all **<when>** values must be set to *Never*. Considering these three values to be ordered in increasing order, the following relations must hold for both pools and totals for each pool which a remote system is processing:

$$\text{Host Receive} = \text{Remote Send} \geq \text{Host Send} \geq \text{Remote Receive}$$

For each pool which the host is offering, it must set its value for **<receive ... pool>** to at least *Final*. Remote systems set **<send ... pool>** equal to the host's **<receive ... pool>**; the host will request pools at that rate. The remote systems set **<receive ... pool>** equal to or less than the host's **<send ... pool>**; the host will send at the remote's **<receive ... pool>** rate. The relations also apply to the corresponding total values.

The values of **<send ... pool>** and **<receive ... pool>** for pool types which are transmitted as scan messages must be either *Final* or *Never*.

The **<xfer will pay>** settings imply *send* from the host and *receive* at the remote, since these are one-way transfers. The **<xfer will pay>** value may be only *Final* or *Never*. If both host and remote set the values to *Final*, the host will transmit will-pays as part of the payoff sequence in the next -to-last race of the pool type.

When feasible, the host system should allow the transmission of pools and scans and will pays to the remote system.

13. The **<pool unit>** field specifies the smallest granularity and multiple in which gross pool, total, and scan amounts will be sent for this pool. The remote system must not allow bets in multiples less than the host's amount if it participates in this pool type, and must respond with a **<pool unit>** at least as large in its Configuration message. For instance, if the host's **<pool unit>** is \$1, then the remote cannot allow bets in \$0.50 multiples. If the host's **<pool unit>** is less than the remote's, the remote must be prepared to round any gross pool, total, or scan values received from the host in a reasonable way, or decline to participate in the pool. Such rounding would affect the accuracy of locally formatted displays.
14. The **<min payoff>** field specifies the minimum price per dollar for the pool. For instance, common values would be \$1.05 or \$1.10. *Note: In ITSP levels 05.13 and older, this specified the minimum profit, commonly \$0.05 or \$0.10.*
15. The **<break>** field contains the 'break to' amount per dollar for the pool. For instance, common values would be \$0.10 or \$0.05.
16. The *c* in the **<pool configuration>** definition is the number of commission rates being sent and is determined by the **<commissions>** field with a maximum of four commissions for each pool. If a pool's commission is broken down into more than one recipient, all recipient commissions must be sent for that pool so that each **<rounding>** type can be applied separately. Conversely, if the sum of the recipient commissions must be equal to a specified percent, only that one "total" commission should be sent. If the **<calculation>** field is *S*, the remote must accept the host's commission rates and return the same commissions. If the **<calculation>** is *N*, the host must accept the remote's commission rates. The **<payoff>** commissions will be based on the received remote commission breakdown.
17. The **<rate>** field gives the total commission rate for the pool, and is in the form of a fractional amount that is less than one.

18. The **<rounding>** field indicates the type of rounding to be used in the commission calculation. The valid rounding values are:
 - U - Round up
 - D - Round down
 - R - True rounding
19. After the host sends a **<configuration>** message the host will request the remote system's configuration to update its configuration information for the remote.
20. The host system is controlling in the sense that it defines the event's characteristics and the remote system's characteristics must be a consistent subset of the host's. Independent and possibly inconsistent activity is permitted on a remote system so long as these activities are handled entirely by the remote system totally independent of the protocol.
21. For exchange pool types there will be a **<pool configuration>** section for each part of the exchange. The **<exchange part>** field indicates which part of an exchange pool type is being described. The first part is "a", the second part is "b", etc. For non-exchange pool types **<exchange part>** should be set to "a". Exchange types which are not the first part should set the number of **<commissions>** to zero; all other fields are still relevant.
22. The **<scan mode>** field indicates the organization of scan data. The host informs the remote system of the type of scan that will be required. The valid **<scan mode>** values are described in the *Scans* message section. Unless otherwise noted, a remote system that cannot accommodate the specified **<scan mode>** must not participate in the pool type. In order to accommodate a **<scan mode>**, the remote must be prepared to supply any and all the requested data that the host is allowed to request within that **<scan mode>**. More specifically, in the case of a Remote-to-Intermediate-to-Host linkup, all of the intermediate's remotes must either be able to "accommodate" the **<scan mode>** or decline to participate in pool type otherwise the intermediate will fail to accommodate it's **<scan mode>** obligations to the host.

Race Status

Syntax

<race status>	:= <status><post time><display><runners>{<runner status>*r} <brackets>{<bracket status>*b} <pools>{<pool definition>*p}<scratch pools>{<scratch info>*s}
<status>	:= <O> <C> <X> <P> <F> <U>
<post time>	:= <time> <no post>
<time>	:= <hours><minutes><seconds>
<hours>	:= {<digit>*2}
<minutes>	:= {<digit>*2}
<seconds>	:= {<digit>*2}
<no post>	:= {<period>*6}
<display>	:= <M> <C> <P> <F>
<runners>	:= {<digit>*2} <no runners>
<no runners>	:= {<period>*2}
<runner status>	:= <digit> <scratch>
<scratch>	:= <alpha>
<brackets>	:= {<digit>*2} <no brackets>
<no brackets>	:= {<period>*2}
<bracket status>	:= <numeric range>
<pools>	:= {<digit>*2} <no pools>
<no pools>	:= {<period>*2}
<pool definition>	:= <pool code><pool status><pool race list><min bet><net addin><gross addin>
<pool code>	:= {<alpha>*3}
<pool status>	:= <O> <C> <X> <E>
<pool race list>	:= {<race>*l}
<race>	:= {<digit>*2}
<min bet>	:= <amount>
<net addin>	:= <amount> <no addin>
<gross addin>	:= <amount> <no addin>
<no addin>	:= <period>
<scratch pools>	:= {<digit>*2} <no pools>
<scratch info>	:= <pool code><race><pool runners>
<pool runners>	:= {<numeric range>*n}
<numeric range>	:= [<numeric span>[<,><numeric span>...]]</>
<numeric span>	:= <number> <number><-><number>
<number>	:= <digit> <digit><number>
<amount>	:= {<digit>*d ₁ }<last digit> {<digit>*d ₁ }{<decimal digit>*d ₂ }<last decimal>
<last digit>	:= digit code+10 ₁₆
<decimal digit>	:= digit code+20 ₁₆
<last decimal>	:= decimal digit code+10 ₁₆

Semantics

1. The values for **<status>** are:

- O - Open
- C - Closed
- X - Cancelled
- P - Post Time
- F - Official
- U - Unofficial

The remote system may not wager on races whose status is *Cancelled*, *Post Time*, *Unofficial* or *Official*. If a race status of *Cancelled*, *Post Time*, *Unofficial*, or *Official* is received and the remote system is allowing wagering on the received race, the remote must close all pools for that race. However, if a race status of *Post Time* is received during a normal bet- or cancel-closing delay sequence, the delay may continue to normal completion before closing. *Unofficial* means that the host has made its prices unofficial after previously making them official, and the remote system must make its prices unofficial. The host, upon making its pools official again, must send the entire payoff sequence and the *Official* status to the remote. Other values of the **<status>** field are informational only and may optionally be used by the remote system to control wagering on the race.

During a link sequence the host will detect any races where it has prices official and the remote does not, according to the **<status>** received from the remote. The host will send these prices only. A race status of *Official* from the remote system means that all prices are official or refunded.

2. The **<post time>** field is in universal time. **<no post>** indicates that post time is undefined. To avoid errors due to time differences between systems, this time should only be used by computing an offset from the time in the message header. If the **<status>** is *P* or *F* the **<post time>** field is defined to be the time the terminals were locked by the sending system
3. The **<display>** field indicates the state of the odds and price displays. The values are:
 - M - Morning line odds displayed.
 - C - Computed odds displayed.
 - P - Parade time - Computed odds displayed - Prices removed
 - F - Final Cycle - Final odds are displayed.
4. The *r* in the **<race status>** message is the number of runners for the race and is determined by the **<runners>** field. All runners beyond *r* were 'never live'. If the **<runners>** field is **<no runners>** no runner information being sent. Remote systems are not required to send this information. Host systems should always include this information unless otherwise noted. If any runner information is sent, all runners for the race must be sent.
5. The **<runner status>** values are:
 - 0 - Never Live
 - 1-9 - Live runner.
 - A-Z - Scratched runner.

where:

- 5.1 For live runners the digit indicates the number of live runners coupled as a single wagering interest. An entry of 9 indicates 9 or more runners coupled as a single wagering interest.

- 5.2 For scratched runners the letter is the race number plus 40₁₆ during which the last live runner was scratched. The time period covered by a race is defined by the host rules covering refunding for the pools involved, usually from the previous race's final cycle to this race's final cycle, or from official to official. The time before race 1 is "during" race 1.
- 5.3 Once the **<runner status>** for a runner is live or scratched on the host it may not be set to *Never Live*.
- 5.4 If a remote cards a runner which is Never Live on the host, and there is money bet on the runner the remote system must payout and account for this money locally without any transfer of information regarding the Never Live runner to or from the host (ie to the host the runner must appear to be Never Live).
6. The *b* in the **<race status>** message is the number of groups of bracketed runners in the race and is determined by the **<brackets>** field. If the **<brackets>** field is **<no brackets>** no bracket information is being sent. Remote systems are not required to send this information. When applicable, host systems should always include this information unless otherwise noted.

Each **<bracket status>** field identifies a list of runners that are numbered separately but are grouped (bracketed) together for price calculation purposes in some pool types. Bracketed groups must be mutually exclusive and may include only live runners. The "coupling" method where multiple runners are grouped as a single numbered entry, and the "bracket" method described here, cannot be used together in the same program.

7. The *p* in the **<race status>** message is the number of pools in the race and is determined by the **<pools>** field. If the **<pools>** field is **<no pools>** pool information is not being sent. If the **<pools>** field is zero then no pools are participating in the race.
8. Any exchange portion of an exchange pool is considered to be a pool in the race in which the exchange occurs and always shows a **<pool status>** of *exchange*.
9. The values for **<pool status>** are:
- O - Open
 - C - Closed
 - X - Cancelled
 - E - Exchange

If the host receives a **<pool status>** of *Open* and remote pools on the host are indicated as final, the host should report the remote's pools being incorrectly open and optionally remove the final indicators from the remote's pools if the host race is open. This would allow the subsequent transfers of non-final pools to the host and the final transfer of the new pool values.

10. The *l* in the **<pool race list>** definition is one less than the number of races in the pool. The race list contains a list of the second and subsequent races of the pool. If the pool is an exchange pool, the **<pool race list>** is the race of the next exchange portion of the pool. If there are no subsequent exchanges, the **<pool race list>** field will contain zero.
11. The **<min bet>** field specifies the sending system's minimum bet allowed for this wager. This includes all permutation bets such as box and wheel bets. This field shall be used as informational only, but must be filled in accurately.
12. The **<net addin>** field contains the addin plus carryover for the pool, net of commission. The **<gross addin>** field contains an amount to be added to the pool total before commissions are removed. The fields are not related and both could be present. For an exchange pool with an addin or carry over, the addin amounts will be sent with the non-exchange portion of the pool. An addin field is set to **<no addin>** when an amount is not to be transmitted. A remote system always sends **<no addin>**. The host may send **<no addin>** before the race is made current, and after the pool's final race is official, even when an amount is available.

13. The *s* in the **<race status>** message is the number of **<scratch info>** fields and is determined by the **<scratch pools>** field. If the **<scratch pools>** field is **<no pools>** this information is not being sent. Remote systems are not required to send this information. Host systems should always include this information unless otherwise noted.

A **<scratch info>** field identifies a pool and its first race number, and shows specifically how this pool is affected by live runners. Race status messages not contained in a link sequence will contain **<scratch info>** data for each pool that has any leg in that race. For instance, a race 3 scratch could affect a race 1-2-3 Pick-3, a race 2-3 Double, and several single-race pools. When a race status message is formatted in a link sequence, **<scratch info>** fields will be included for each pool whose first leg is in that race. For instance, the race 1 race status message would contain data for the race 1-2-3 Pick-3, but the race 2 and 3 race status messages would not.

The *n* in the **<pool runners>** is the number of races in the pool. The **<pool runners>** field contains sections with the non-refundable live runners for each race of the pool. Each section includes one or more runners with successive sections, including the last, being delimited by a slash. Eventually, a refund may be made for wagers which contain at least one runner which is not in the **<pool runners>** list for that race. This type of refund will be called an implicit refund. This information will affect the contents of all subsequent **<pool total>** messages. The last implicit refund information sent for a pool should agree with that in the **<payoff>** message when that becomes available. *NOTE: live runners are listed here by race, not by position as in the <live runner> field of the <payoff> message.*

14. If the **<race status>** message is for the current race and the message is not the post time message, the reason code must be set to *Begin* to indicate that this is the current race.
15. A **<reason>** of *End* in the header will indicate post time for the current race and no runner or pool data will be sent. The acknowledge for this message must have a reason code of *End* indicating that the remote system has received the post time message. This message must be sent from the host before any intentional **<close bet delay>** or **<close cancel delay>** by the host. For this reason, the **<status>** might not yet be set to *P*. Upon receipt of this message, the remote must start its bet- or cancel-close delay sequence to close the pools. Each time a remote receives a **<race status>** message with **<reason>** of *End* and the remote's pools are final the remote will send a *PPf All Pools* message.
16. A **<race status>** message with **<reason>** of *End* may be issued by the host at any time, even though another extended sequence may be in progress. Any sequence in progress will be terminated by the occurrence of this message. The host must repeatedly send this message until it either receives a valid acknowledge or normal retry and time-out conditions indicate a link failure.
17. The host must send to the remote system a subset of all the pools in the host's racecard which are contained in the remote's **<configuration>** message. If any pool information is sent, all pools for the race must be sent that represent all the available pools for the specified race. If the host sends status message with **<pools>** set to zero, then there are no pools available for wagering for the specified race. If a pool is present in the host's race card it may not be deleted on subsequent race card transfers (i.e. the host must dispose of the money bet in the pool either through a payoff or refund).
18. A **<race status>** message from the remote system must include a pool section that is a consistent subset of the pools in the host's race card. The pools received by the host from the remote system are used to indicate to the host the pools the remote system is wagering on. All other information in the message will be disregarded by the host and should either be set to the values provided by the host or not transmitted if the data value has the capability. If a remote is participating in a pool, the remote must participate in all races for that pool. For multi-race pools, all races used by the pool must be participated by the remote. The remote may choose not to participate in all of the pools for the races, but the race must be participating.

Inter-Tote System Protocol

If the remote system sends a **<race status>** message with **<pools>** set to zero, the remote does not have any participating pools for the specified race and will not receive any pool specific messages for the race. If a pool is not present in the remote's **<pool definition>** list, the remote is not participating in the missing pool and will not transmit nor expect to receive pool specific messages for that pool or be included in the merged pools for the race (i.e. the host assumes the remote is not wagering this particular pool for this race). The host will only send or request pool specific messages for the remote participating pools. Pool specific messages include:

- Pools
- Scan
- Pool Total
- Payoff
- Will Pay

Syntax

<pool>	:= <pool header> [<pool data>]
<pool header>	:= <pool mode> <segments> <segment>
<pool mode>	:= <G> <N>
<segments>	:= { <digit> *2}
<segment>	:= { <digit> *2}
<pool data>	:= <rows> <columns> { <row data> * r } <segment total> <total> <net total>
<rows>	:= { <digit> *2}
<columns>	:= { <digit> *2}
<row data>	:= { <amount> * c } <row total>
<row total>	:= <amount>
<segment total>	:= <amount>
<total>	:= <amount>
<net total>	:= <amount>
<amount>	:= { <digit> * d ₁ } <last digit> { <digit> * d ₁ }{ <decimal digit> * d ₂ } <last decimal>
<last digit>	:= digit code+10 ₁₆
<decimal digit>	:= digit code+20 ₁₆
<last decimal>	:= decimal digit code+10 ₁₆

Semantics

- The **<pool mode>** field indicates whether a gross pool or net pool is being transferred. The valid **<pool mode>** values are:

- G - The pool is a gross pool.
- N - The pool is a net pool, i.e. commission has been removed from each pool entry.

Net pools are sent with values rounded up to the nearest higher cent.

The **<pool mode>** is determined by the **<calculation>** field of the hosts configuration. Gross pools are transferred by the remote always, and by the host system if the **<calculation>** field is *S*. Net pools, gross pools, or both may be transferred by the host if the **<calculation>** field is *N*, depending on the settings of **<send gross pool>** and **<send net pool>**.

- The *c* in the **<row data>** definition is the number of columns in the row and is determined by the **<columns>** field. If *c* is zero the **<row total>** field must be zero.
- The *r* in the **<pool data>** definition is the number of rows in the matrix and is determined by the **<rows>** field. The **<segment total>** is the total of all **<row total>**'s for the message. If *r* is zero then:
 - The **<columns>** and **<segment total>** fields must be zero
 - The message represents a zero pool or pool segment

4. A triple leg **<pool>** message is actually an extended sequence of s **<pool>** messages where s is determined by the **<segments>** field and **<segment>** indicates which segment in the sequence is being sent. All s **<pool>** messages must be sent in increasing order of **<segment>**. The **<segments>** field must not be zero and must agree with the host value.

The **<pool>** *pending* and *request* messages will have a data section containing only **<pool header>**. A **<pool>** *pending* or *request* message for a multiple segment transmission has **<segments>** and **<segment>** fields indicating, respectively, the total number of segments in the sequence, and the particular segment which is available or being requested. In single segment transmissions, those fields should both be one (1).

5. The **<total>** is the cumulative total of all segments in the sequence up to and including the segment being sent. For single segment pool transfers, the **<total>** will have the same value as the **<segment total>**. Each message in the sequence is independent of all other messages in the sequence in terms of the protocol control and error handling rules. An extended sequence is terminated and considered to have failed if it is interrupted by any other host initiated message.
6. The **<net total>** is used for all pool transfers from the host if the **<pool mode>** is 'N' (NET), and is otherwise zero. The **<net total>** will be zero for all but the last segment of a pool transfer, in which it will represent the live pool total net of commissions, rounded to cents by the host as specified in the Configuration messages sent from the remotes. Note that this total may be different from **<total>** due to rounding methods.
7. If a pool has p pool positions and p is less than four it will be sent in **<pool>** format. If a pool message is not a zero pool or segment, the dimensions in the pool message must not be greater than the number of runners in the host's race card (live or scratched) for the race associated with that position. All data beyond the transmitted rows and columns are zero and must be set by the receiving system to zero. Pool positions are sent in the following order:

	Pool Type	Segments	Segment	Rows	Columns
Single Leg	1	1	1	First Position	
Double Leg	1	1	First Position	Second Position	
Triple Leg	Runners in	First Position	Second Position	Third Position	
		First Position			

Symmetric pools, such as the Quinella, will be sent in upper triangular form with all data below the diagonal sent as zero.

8. Triple leg pools will only be sent on final cycle unless the rules require the display of pool information during the race.
9. The pool data will consist of the sum of the pools for all sources served by the sending system.
10. If a pool has p pool positions and p is greater than three the data will be sent as a scan message, with the specific format, dimensions, and data content being pool and rule dependent (see **<scan>** messages).
11. The acknowledge message for a pool transfer message that is accepted will have a data section containing the total amount of the pool received. For a **<pool>** transfer of a triple leg pool this will be the cumulative total of all the pool segments received.
12. To synchronize the host and remote system displays, the host system will use the same pool data as is sent to the remote systems to generate its local displays. Display data on all systems should remain constant between pool transfers from the host.
13. Upon receipt of a **<race status>** message indicating post time the remote system will close its pools, and when its final pools are ready, will send a *Pool Pending* message with the **<pool code>** set to *All Pools* and the reason set to *Final*.

Inter-Tote System Protocol

The only valid response to any *Pool Pending Final* message from a remote system is a message of type *Acknowledge* with the same **<pool code>** received. The actual pool transfers will be controlled by the host using request message sequences for each of the individual pool types to be transferred. The remote cannot send pools to the host without the host requesting the data.

Upon receipt of a *Pool Pending Final* message, the host should request the corresponding pools and pool totals from the remote as soon as possible.

14. When the host has generated final merged pools, it will initiate a reverse pool transfer on a pool by pool basis using a pool pending message for each pool with the reason set to final.

Scan Request Begin

Syntax

```

<scan request begin> := <scan mode><scan runner list><leg>
<scan mode>         := <E>|<L>|<S>|<X>|<Q>|<A>|<C>|<K>|<P>
<scan runner list> := <combos>{<scan runners>*c}<period><live runners><period>
                    <favorite runners><period><sub runners>
<leg>               := {<digit>*2}
<combos>            := <digit><digit>
<scan runners>     := <runner list>
<live runners>     := <runner list>
<favorite runners> := <runner list>
<sub runners>      := <runner list>
<runner list>      := {<numeric range>}*p
<numeric range>    := [<numeric span>[<,><numeric span>]...]</>
<numeric span>     := <number>|<number>-<number>
<number>           := <digit>|<digit><number>

```

Semantics

1. All scans will be host initiated to permit the use of rule independent scan procedures on remote systems.
2. All pool types with four or more legs will be requested as scans.
3. The **<scan mode>** field indicates the organization of data. The host informs the remote system of the type of scan required. The valid **<scan mode>** values are described in the *Scans* message description, below.
4. The **c** in the **<scan runner list>** field is the number of **<scan runners>** combinations in the list and is determined by the **<combos>** field. If more combinations are required than the host or remote system can support, a manual price calculation procedure based on a complete matrix printout will be used.
5. The **<scan runners>** for a *p* position scan will have *p* sections. For early scans (<scan mode> *E* and *K*), the runners of the races which have not been completed must be set to an empty **<numeric range>**. Each section may have one or more runners with successive sections, including the last, being delimited by a slash.
6. The **<live runners>** for a *p* position scan will have *p* sections indicating which runners are live for each leg of the scan.
7. The **<favorite runners>** for a *p* position scan will have *p* sections indicating which runners are the win odds favorites for each leg of the scan. See the **<scan>** message description for more details.
8. The **<sub runners>** for a *p* position scan will have *p* sections indicating which runners are eligible for ‘alternate runner’ or for ‘win odds favorite’ substitution. See the *Scans* message description for more details.
9. The **<leg>** indicates the runner distribution leg for <scan mode> *E* and *K*, supporting unambiguous intermediate scans when progressively scanning, i.e.: scans after each leg completion with the next leg’s live money distribution. For <scan mode> *E* and *K*, this field indicates which leg’s runners “label” the columns, and valid values are 2 through *N*, where *N* is determined by the pool. In the classic <scan mode> *E*, applying to Pick-*N* pools, <leg> is set to *N*. For <scan mode> *L* and *A*, <leg> can only be set to *N* as intermediate scans are not supported.

Inter-Tote System Protocol

10. A **<scan request begin>** message has a **<reason>** of *Begin* and indicates a *Request* message from the host to generate a scan for the designated **<scan runners>** and **<live runners>**. Upon receiving this message the remote system will send an *Acknowledge* message with **<reason>** of *Begin* and then generate a scan with the designated runners.
11. When the scan generation is completed on the remote system it will send a *Pending Scan* message with the reason set to *Final*. See the *Scans* message description for more details.

Scans

Syntax

<scan>	:= [<scan mode><scan runner list><scan header>[<pool data><scan total><live total>]
<scan mode>	:= <E> <L> <S> <X> <Q> <A> <C> <K> <P>
<scan runner list>	:= <combos>{<scan runners>*c}<period><live runners><period> <favorite runners><period><sub runners>
<combos>	:= <digit><digit>
<scan runners>	:= <runner list>
<favorite runners>	:= <runner list>
<sub runners>	:= <runner list>
<live runners>	:= <runner list>
<runner list>	:= {<numeric range>}*p
<numeric range>	:= [<numeric span>[<,><numeric span>]...]</>
<numeric span>	:= <number> <number>-><number>
<number>	:= <digit> <digit><number>
<scan header>	:= <leg><pool header>
<leg>	:= {<digit>*2}
<pool header>	:= <pool mode><segments><segment>
<pool mode>	:= <G> <N>
<segments>	:= {<digit>*2}
<segment>	:= {<digit>*2}
<pool data>	:= <rows><columns>{<row data>*r}<segment total><total><net total>
<rows>	:= {<digit>*2}
<columns>	:= {<digit>*2}
<row data>	:= {<amount>*c}<row total>
<row total>	:= <amount>
<segment total>	:= <amount>
<total>	:= <amount>
<net total>	:= <amount>
<scan total>	:= <amount>
<live total>	:= <amount>
<amount>	:= {<digit>*d ₁ <last digit>}{<digit>*d ₁ }{<decimal digit>*d ₂ <last decimal>
<last digit>	:= digit code+10 ₁₆
<decimal digit>	:= digit code+20 ₁₆
<last decimal>	:= decimal digit code+10 ₁₆

Semantics

1. The rules governing the use of gross and net scans are identical to those for the use of gross and net pools.
2. If a pool has p pool positions and p is greater than three the data will be sent as a scan message sequence, with the specific format, dimensions and data content being pool and rule dependent.
3. The <scan mode> field indicates the organization of data. This allows future expansion should one pool type have two possible methods of data organization. The host informs the remote system of the type of scan required. The valid <scan mode> values are:

P - Used in a Configuration message only, this means a pool message is used so scans are not applicable.

Inter-Tote System Protocol

- A - Alternate runner Late scan. Used after the winners are known for the last race of Pick-N pool types. Scan for bets matching any x selections out of N , taking into consideration any alternate runners selected in the bets for substitutable scratches. When the host specifies this **<scan mode>** in the Configuration message, the remote system may respond with E in its configuration, indicating that the remote system will not be using alternate runner selections. The host must conform and use E in subsequent **<scan request begin>** and **<scan>** messages.
 - E - Early scan, used after the winners are known for the next to last race of Pick-N pool types; scan for bets matching any x selections out of N .
 - L - Late scan, used after the winners are known for the last race of Pick-N pool types; scan for bets matching any x selections out of N . Late scans will be used only when the rules make early scans impractical.
 - S - Superfecta late scan data, used after the winners are known.
 - X - Exact-N late scan data, used after the winners are known.
 - C - Combination List Late scan. A generic form of scan used after the winners are known, for any appropriate pool type; scan for bets matching exact combinations.
 - K - Combination List Early scan. A generic form of scan used after the winners are known for the next to last race for Pick-N pool types; scan for bets matching exact combinations.
 - Q - Quiniela-Double early scan data, used after the winners are known for the first race of the Quinella Double (or the Double Exacta).
4. The **<scan runners>** for a p position scan will have p sections. For early scans (**<scan mode>** *E* or *K*), the runners of the races which have not been completed must be set an empty **<numeric range>**. Each section may have one or more runners with successive sections, including the last, being delimited by a slash.
 5. The **<live runners>** for a p position scan will have p sections indicating which runners are live (not refunded) for each leg of the scan.
 6. The **<favorite runners>** for a p position scan will have p sections indicating which runners are the win odds favorites for each leg of the scan. If this pool type has no use for win odds favorites, each of the p sections will be empty, containing just the terminating slashes.
 7. The **<sub runners>** for a p position scan will have p sections indicating which runners are eligible for 'alternate runner' or for 'win odds favorite' substitution. For each position where the win odds favorite actually wins (according to host rules), these **<sub runners>** are also included in the **<scan runners>**; this enables remote systems not configured for 'alternate runner' substitution to ignore the **<favorite runners>** and **<sub runners>** fields when collating scan data. If this pool type has no use for substitutions, each of the p sections will be empty, containing just the terminating slashes.
 8. The **<scan total>** indicates the total money bet in the pool including any refund. The **<live total>** is the total of all nonrefundable (i.e. "live") money in the pool. Both of these fields may be zero for all but the last segment of a scan transfer.
 9. The **<net total>** is used for all scan transfers from the host if the **<pool mode>** is 'N' (NET), and is otherwise zero. The **<net total>** will be zero for all but the last segment of a scan transfer, in which it will represent the live pool total net of commissions, rounded to cents by the host as specified in the Configuration messages sent from the remotes. Note that this total may be different from the sum of individual net amounts in the **<scan>** message due to rounding methods.
 10. A late **<scan>** message (**<scan mode>** *L* or *A*) for a Pick-N pool uses a single leg **<pool data>** message with the **<columns>** field equal to $N+1$. The i th entry is the amount of money with $N-i+1$ winners. Normally, one combination is specified in **<combos>**. The **<leg>** field must be set to N as intermediate scans are not supported.
 11. An early **<scan>** message (**<scan mode>** *E*) for an Pick-N pool uses a double leg **<pool data>** message with the **<columns>** field equal to the number of runners in the **<leg>** race of the pool and the **<rows>** field equal to N . The i th row contains the amount of money on each runner in the **<leg>** race with $N-i$ winners in the first **<leg>-1** races. Normally, one combination is specified in **<combos>**.

12. A superfecta late scan (<**scan mode**> S) uses a double leg <**pool data**> message for each <**scan runners**> combination specified (normally one). There will be one row for each permutation of dead-heated runners in the combination with the rows (and permutations) in lexical order (left to right in ascending order) of finishers. Each row will have 16 columns with all exact finishes for the four runners associated with the row. The columns will be in the following order:

1234 123L 12L4 1L34 L234 12LL 1L3L 1LL4 L23L L2L4 LL34 1LLL L2LL LL3L LLL4 LLLL

where:

- 12.1 1 - A runner finishing first
 2 - A runner finishing second
 3 - A runner finishing third
 4 - A runner finishing fourth
 L - All live runners
- 12.2 If less than 4 runners finish the race, the entries in the matrix that correspond to positions in which no runner finished will be zero and those sections of the <**scan runners**> will be empty.
- 12.3 A maximum of 16 rows will be sent. If more than 16 rows are required, a manual price calculation procedure based on a complete matrix printout will be used.
- 12.4 Normally there will be just one <**scan runners**> combination, since dead-heats can be permuted from it. Note however, that other combinations may be present, even seemingly illogical ones with repeated runners. For instance, 5555 could be requested in the second (exchange) part of a Tri-Super, if 5 is a late scratch and the amounts bet on LLL5, LL5L, L5LL, and 5LLL are needed for a consolation calculation.
13. An Exact-N late scan (<**scan mode**> X) uses a double leg <**pool data**> message for each <**scan runners**> combination specified (normally one). There will be one row for each permutation of dead-heated runners in the combination with the rows (and permutations) in lexical order (left to right in ascending order) of finishers.

Each row will have $N+1$ columns. The first column is for an exact match, then the first $N-1$ finishers followed by one *all* (i.e. all live runners), then the first $N-2$ finishers followed by two *alls*, etc., ending with N *alls*. No other scrambling of the finishers with *all* is supported. For example an Exact-6 (or Hexafecta) would use this order:

123456 12345L 1234LL 123LLL 12LLLL 1LLLLL LLLLLL

where:

- 13.1 1 - A runner finishing first
 ...
 6 - A runner finishing sixth
 L - All live runners
- 13.2 If less than N runners finish the race, the entries in the matrix that correspond to positions in which no runner finished will be zero and those sections of the <**scan runners**> will be empty.
- 13.3 A maximum of 16 rows will be sent. If more than 16 rows are required, a manual price calculation procedure based on a complete scan printout will be used.
14. A Quiniela-Double scan (<**scan mode**> Q) is used for the pool types Quinella-Double (QD), the Double-Exacta (DE), the Italian Double Quinella (ITQ), and the Italian Double Exacta (ITE). For the QD and DE, this is an early scan which will be requested when the finishers for the first race are known. For the ITQ and ITE, which are exchange types, this is a late scan since it cannot be requested until the second race is closed.

This scan includes one **<scan runners>** combination for each double leg **<pool>** matrix required. The first two positions of **<scan runners>** select the first race winners. The last two positions will specify all/all, indicating all of the runners which are required to compute the pool matrix.

Since the QD and ITQ are symmetric pool types, each double leg matrix will be sent in upper triangular form with all data below the diagonal sent as zero, and first-race **<scan runners>** will be in ascending order only. This does not apply to the DE pool type, or the ITE pool type which could be paired with an ITQ and affect either race.

For each matrix required, selected double leg partial pools from the theoretical quadruple leg pool are summed. For instance, if a single **<scan runners>** specifies 1/2,3/all/all then the requested matrix must contain the sum of the 1/2/all/all, and 1/3/all/all partial pool matrices (and their reverses for a Quinella-Double). If two **<scan runners>** specify 1/2/all/all and 1/3/all/all, then two separate partial pool matrices are required.

NOTE: The host will normally make one scan request containing a list of every combination of first race winners that may be needed. Example 1: For ARCI QD rules and a normal 1/2/3/4 finish, there would be two partial pool matrices, 1/2/all/all and all/all/all/all. Example 2: For ARCI QD rules and a dead heat for 2nd in the first race, there would be the three partial pool matrices 1/2/all/all, 1/3/all/all, and all/all/all/all. Example 3: For some other hypothetical QD rules, and a dead heat for 2nd in the first race, there might be all of those matrices plus the four matrices 2/3/all/all, 1/all/all/all, 2/all/all/all and 3/all/all/all.

15. A Combination List Late scan (**<scan mode>** C) uses one message for each **<scan runners>** combination specified. When the amount bet on several combinations are required, each combination is listed explicitly. Each message uses a single leg **<pool data>** message formatted as one row with one column containing the amount bet on the combination specified.

NOTE: Any of these combinations that will eventually appear as winners or refunds in a **<payoff>** message must be mutually exclusive, i.e. must not overlap. The remote system must be able to exactly match each winning combination in the **<payoff>** message with one in the scan data to calculate its own liabilities, with no duplication. For example, 1 / 2 / 3 / 4 and 1 / 2 / 3 / 1-8 overlap but 1 / 2 / 3 / 4 and 1 / 2 / 3 / 5-8 do not. Other scan combinations needed by the host have no such restrictions, so if a bet matches two scan combinations, its amount must be included in both.

16. A Combination List Early scan (**<scan mode>** K) for a Pick-N pool uses one message for each **<scan runners>** combination specified. When the amount bet on several combinations are required, each combination is listed explicitly. Each message uses a single leg **<pool data>** message formatted as one row, with the **<columns>** field equal to the number of runners in the **<leg>** race of the pool. The row contains the amount of money on each runner in the **<leg>** race with **<leg>-1** winners in the first **<leg>-1** races.
17. The *acknowledge* **<scan>** message for a scan pool transfer that is accepted will have a data section containing the **<segment total>**. The **<segment total>** is the total of all **<row total>**'s for the received *data* message.
18. The *pending* and *request* **<scan>** messages will have a data section containing only **<scan header>**. A **<scan>** *pending* or *request* message for a multiple segment transmission has **<segments>** and **<segment>** fields indicating, respectively, the total number of segments in the sequence, and the particular segment which is available or being requested. In single segment transmissions, those fields should both be one (1). The **<leg>** field indicates which leg of the scan is specified.
19. Multiple segment transmissions containing any of the above **<scan mode>** formats are sent in the same fashion as triple leg **<pool>** messages. The **<segments>** and **<segment>** field in the **<pool>** message control the message transfers. The number in **<segments>** redundantly matches the number in **<combos>**, and the current **<segment>** number specifies which **<scan runners>** combination matches the data in the message. For simplicity, every **<scan runners>** combination is included in each segment.

20. Scan pool data will only be sent to the host if the host requests the scan pool data or as a result of a **<scan request begin>** message. When a scan generation is completed on the remote system it will send a *Pending Scan* message with the reason set to *Final*.

The only valid response to any *Scan Pending Final* message from a remote system is a message of type *Acknowledge* with the same **<pool code>** received. The actual scan transfers will be controlled by the host using a request message sequence for the individual pool type to be transferred. The remote cannot send a scan to the host without the host requesting the data.

Upon receipt of a *Scan Pending Final* message, the host should request the corresponding scans from the remote as soon as possible.

Totals

Syntax

<pool total>	:= <live total><net total>
<live total>	:= <amount>
<net total>	:= <amount>
<amount>	:= {<digit>*d_1><last digit> {<digit>*d_1>{<decimal digit>*d_2><last decimal>
<last digit>	:= digit code+10₁₆
<decimal digit>	:= digit code+20₁₆
<last decimal>	:= decimal digit code+10₁₆

Semantics

1. The **<live total>** will consist of the sum of all nonrefundable (i.e. "live") money in the pool for all sources served by the sending system, regardless of commission.
2. The **<net total>** will consist of the sum of all non-refundable money in the pool for all sources served by the sending system net of commissions. This number will only be used if **<calculation>** in the Configuration message is *N*, otherwise it will be zero.
3. Pool totals must be requested by the host system for all pools which are being transferred by scans during the race in which the pool is being wagered as a cross check on the integrity of the scan.
4. The **<live total>** for pools which are being transferred by scans will be sent as the total money bet in the pool including any refund, always assuming that the refund amount is not yet known. The **<net total>** is zero.

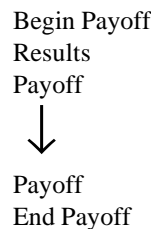
Payoffs

Syntax

<payoff>	:= <live runners><refund><carry over><commissions>{<commission>*c} <prices>{<price definition>*p}
<live runners>	:= <runner list>
<refund>	:= <liability>
<carry over>	:= <amount>
<commissions>	:= <digit>
<commission>	:= <amount>
<prices>	:= {<digit>*2}
<price definition>	:= <status><winners><runner list><price><winning><liability><break amount>
<status>	:= <R> <E> <\$> <P> <A> <C> <Q> <r>
<winners>	:= {<digit>*2}
<runner list>	:= {<numeric range>}*r
<numeric range>	:= [<numeric span>[{ <, ><numeric span>...}]</>
<numeric span>	:= <number> <number><-><number>
<number>	:= <digit> <digit><number>
<price>	:= <signed amount>
<winning>	:= <amount>
<liability>	:= <signed amount>
<break amount>	:= <signed amount>
<signed amount>	:= <sign><amount>
<sign>	:= <+> <->
<amount>	:= {<digit>*d ₁ }<last digit>{ <digit>*d ₁ }{ <decimal digit>*d ₂ }<last decimal>
<last digit>	:= digit code+10 ₁₆
<decimal digit>	:= digit code+20 ₁₆
<last decimal>	:= decimal digit code+10 ₁₆

Semantics

1. A payoff sequence consists of the following messages:



where:

- 1.1 The Begin/End Payoff messages are <payoff> messages with the reason set to Begin/End respectively and no data.
- 1.2 The <reason> in each <payoff> message, except for the Begin and End Payoff is set to *Final* if the prices are official.
- 1.3 The <reason> in the <results> message is set to *Final* if the race is official.

Inter-Tote System Protocol

- 1.4 The **<results>** message must include at least the minimum number of finishers necessary to compute all pools being paid in the race or all finishers if less than this many runners finished the race.
- 1.5 There is one **<payoff>** message for each pool being paid, including pools not yet computed.
- 1.6 The race number in the header of each **<payoff>** message must be set to the race for which prices are being sent.
2. The **<live runners>** field indicates the non-refundable runners for each pool position. It should be noted that for pools which have consolation payoffs or favorite substitutions this list will contain scratched runners. The **<refund>** field indicates the total amount of the refund based on refunding all wagers which contain at least one runner for some pool position which is not in the **<live runners>** list for that position. This type of refund will be called an implicit refund.
3. The **<carry over>** field indicates the new outstanding "jackpot" amount for the pool. A "jackpot" is the amount of today's net addin plus the unpaid amount for today's pool. If the jackpot is paid, the **<carry over>** will be zero. For exchange pools in the non-exchange race, the **<carry over>** will be the outstanding "jackpot" plus the first half carry over amount representing the *new* jackpot if no winning ticket were paid in the exchange race.
4. The *c* in the **<payoff>** message is the number of commissions being sent and is determined by the **<commissions>** field. The number of commissions must agree with the number of commissions in the **<pool configuration>** received from the remote for the pool. Each **<commission>** field is the total commission due the remote for the corresponding commission rates in the pool configuration message for the pool (i.e. the order in which commissions are sent is the same as the order of the commissions in the pool configuration message).
5. The *p* in the **<payoff>** message is the number of prices for the pool, and is determined by the **<prices>** field. If *p* is zero there is no payoff for the pool. A second half exchange that is not paid out will be sent with **<prices>** set to zero.
6. For any pool, when a price is not available, the price transfer will have the **<prices>** in the **<payoff>** message set to zero, a **<reason>** of *No Reason*, a **<live runners>** field of all defined runners for the host, a **<refund>** set to zero, a **<carry over>** set to zero, and a **<commissions>** set to zero. This type of **<payoff>** message indicates that the host has not yet computed the official price and it will be sent at a later time. The remote may not pay any tickets associated with the pool until the official price is received.
7. The **<status>** field defines the type of price being defined and the appropriate action required by the receiving system.

<u>Status</u>	<u>Meaning</u>	<u>Action</u>	<u>Liability</u>
R	Refund	Explicitly refund matching tickets	Total amount of refund
E	Exchange	Generate exchange balance for matching tickets. This status may be sent only when exchanging is actually allowed.	Total amount of exchange
\$	Normal	Pay matching tickets <price>	Total winning money including payout and surcharge
P	Pay Exchange	Pay first half exchange ticket not yet exchanged <price>	Total winning money including payout and surcharge (based on total possible exchanges)
A	Alternate	Pay matching tickets <price> after taking into consideration any substitutable scratches and 'alternate runner' specified in the	Total winning money including payout and surcharge

Inter-Tote System Protocol

ticket. In remote systems not configured for 'alternate runner' substitution, treat status 'A' like '\$'.

C	ITQ Consolation	Italian Double Quinella (ITQ) consolation price. Same as Normal (\$) price except this price has already been rounded by the host with a <break> factor different from that used for normal prices, and must not be rounded again.	Total winning money including payout and surcharge
Q	ITQ Pay Exch.	Italian Double Quinella (ITQ), similar to consolation (C) and Pay Exchange (P) price. Pay <price> to first half exchange ticket not already exchanged, except this price has already been rounded by the host with a <break> factor different from that used for normal prices, and must not be rounded again.	Total winning money including payout and surcharge (based on total possible exchanges)
r	Partial Refund	Explicitly refund matching tickets but the <price> to be refunded is less than the original \$1 bet unit, and must not be rounded or surcharged by the remote system.	Total amount of refund

8. A <price> with <status> value of P or Q is a second half exchange price that is paid to all matching second half tickets and causes all first half \$1.00 exchange prices to be replaced with the second half P price. Otherwise, all first half \$1.00 exchange prices are voided and only the first half non-exchange prices are paid to first half tickets not yet exchanged.
9. The <winners> field is the number of pool positions required to match received <runner list> to receive the payoff. For non-pick and pick three pools it is the number of pool positions.
10. The r in the <runner list> definition is the number of pool positions for the pool. Each pool position may have multiple runners, with successive pool positions, including the last being delimited by a slash.
11. The <price> field in the <price definition> is the \$1 unbroken price, rounded down to the nearest lower cent. Exception: for C and Q prices, the price is already rounded to a broken price by the host and must not be rounded again.
12. The <winning> field is the remote system's winning amount eligible for <price>. For E and R prices, the <winning> and <liability> values will be the same.
13. For an exchange price the <liability> field will be the amount of exchange money and the <price> will be one dollar.
14. For an explicit refund price the <liability> field will be the amount of the refund and the <price> will be one dollar. All and only refunds which are not included in the implicit refund must be sent using an explicit refund.

15. Pools which are completely refunded for any reason such as a no race, cancelled pool or no winning ticket situation will be sent as an explicit refund with no implicit refunds designated. The **<live runners>** should then contain all originally livened runners.
16. The **<liability>** field is defined as the remote system's winning money times the price broken according to host rules. When the remote system might do pre-breakage surcharges a la Illinois, this "broken" price must be identical to the unbroken "penny" **<price>**, and **<break amount>** is computed to the "penny" regardless of the **<break>** value in the *Configuration* message.
17. The **<break amount>** field is the remote system's share of the total breakage, computed and distributed among sources according to host rules. Any fraction of cents will be omitted from the remote system's share of the breakage. See Design Considerations #3 and #4.

Design Considerations

1. Using the standard price transfer for Pick-N pools permits transmission of the appropriate number of prices depending on the rules and/or payoff situation.
2. Early refunding of a pool can be initiated by the remote system based on a pool status of *cancel* in a **<race status>** message.
3. If a remote system does not remove surcharges from payoffs, then the total **<break amount>** is its total share of the breakage. With surcharging, breakage on a remote system can be computed by the system, by removing any surcharges and the system payout amount calculated using broken surcharged prices, from its total **<liability>** and **<break amount>**.

This approach permits the distribution of breakage using ratios of either winning bet or pool total amounts, and guarantees exact network wide balancing.

It should be noted that in the presence of multiple remote system surcharges it is impossible for the host to separately determine the proper total payout, breakage and surcharge amounts for a remote system. The only meaningful number that the host system can calculate is total of the remote system's pre-surcharge liability and breakage.

4. Settlement figures between host and remote must be in whole cents. This is assured by rounding each remote's **<break amount>** down to the cent. Accumulated fractions of cents are retained in the host's breakage. The combined sum of these fractions will be from zero to less than one cent from each payoff, from each remote source, and will always sum to a whole number of cents.

Results

Syntax

<results>	:= <finishers>{<finish>*f}<favorite list>
<finishers>	:= {<digit>*2}
<finish>	:= <runner><entry><position>
<runner>	:= {<digit>*2}
<entry>	:= <alpha>
<position>	:= <digit>*2
<favorite list>	:= {<numeric range>}

Semantics

1. The *f* in the <results> message is the number of finishers being sent and is determined by the <finishers> field. If the race is cancelled <finishers> is zero.
2. The <entry> field indicates which portion of an entry finished in this position. If the runner is not part of an entry the <entry> field will be blank.
3. The <position> field is the position the runner finished in.
4. If any runner for a position is sent all runners for all positions up to and including the given position must be sent. Only the first position involved in a dead heat is sent.
5. If more than one part of an entry finishes in a position it must be included once for each occurrence in that position.
6. All runners for a given position must occur together with positions being sent in increasing order.
7. The <favorite list> is a list of all runners that are win odds favorites and is determined by the runner or runners with the highest amount wagered in the host's merged win pool. This field is informational only and may not be used for calculation purposes or substitution on the remote system.
8. If the race for which results are being sent is official the <reason> in the <results> message will be set to *Final*.

Alert

Syntax

<alert> := [**<alert type>****<text>**]
<alert type> := **<alpha>**
<text> := {**<alpha>****l*}

Semantics

1. The *l* in the **<text>** definition is the number of characters contained in the text and may range from 1 to 256. The value of *l* is determined by the **<length>** field less the size of the **<alert type>** and **<checksum>** fields. The message may consist of any ASCII text, but may not include any control characters.
2. If the **<length>** field of the header is zero, there is no alert data and the message serves as a test message. A test message should be treated as a transparent message initiated by the host for communications testing.
3. The host may issue an **<alert>** message with **<reason>** of *Terminate* to terminate an extended message sequence. Upon receiving an **<alert>** message with **<reason>** of *Terminate* the remote system will acknowledge with **<reason>** of *Terminate* and enter an idle state.
4. An **<alert type>** value "O" indicates that the message is to be directed to the system operator. An **<alert type>** value "T" is for internal informational data, for future application enhancements that are agreed upon by all participants in the network. Other **<alert type>** values may be reserved for private use by network participants, so an **<alert>** message with any unrecognized **<alert type>** value should be acknowledged and ignored. Currently reserved **<alert type>** values are:

A for AmTote International
 U for United Tote
 Z for Autotote

Will Pay

Syntax

<will pay>	:=	<rows><columns>[<will pay row>*<i>r</i>]
<rows>	:=	{<digit>*2}
<columns>	:=	{<digit>*2}
<will pay row>	:=	<runner list>{<will pay item>*<i>c</i>}
<runner list>	:=	{<numeric range>*<i>p-1</i>}
<numeric range>	:=	[<numeric span>[<,><numeric span>...]]</>
<numeric span>	:=	<number> <number><-><number>
<number>	:=	<digit> <digit><number>
<will pay item>	:=	<winners><price><winning>
<winners>	:=	{<digit>*2}
<price>	:=	<signed amount>
<winning>	:=	<amount>

Semantics

1. The *r* and *c* in the **<will pay>** message are the number of rows and columns, determined from the **<rows>** and **<columns>** fields.

The number of rows will equal the number of combinations required by dead heats in the first *p-1* positions, where *p* is the number of positions in the pool type. A maximum of 16 rows will be supported. The number of columns will be the maximum runner number of the originally live runners in the last position of the pool type.

The rows will be in lexical order (left to right in ascending order) of the finishers in **<runner list>**. The *i*th column **<will pay item>** is the data for finisher *i*, which would be in the last position of the combination if all positions were present.

2. The **<runner list>** field contains the finishers in the first *p-1* positions for the items in that row, where *p* is the number of positions in the pool type.
3. The **<winners>** field is the number of pool positions required to match **<runner list>** plus the last position finisher, to receive the price. For non-pick pools it is the number of pool positions.

Note especially for pick three pools, there is no way to format three separate *all* combinations for a single last-position runner, so if a 2-of-3 or 1-of-3 case occurs for some last-position runner, it must be formatted in the Pick-N fashion, or else formatted as 3-of-3 paying nothing. The remote system must accept will pay messages in either format, but may modify the former into the latter for display and report purposes.

4. Individual will pay **<price>** entries will contain the \$1 unbroken price. To avoid ambiguity with a zero price (see below), any unbroken will pay price less than the minimum price specified in the **<configuration>** will be sent as that minimum value.
5. If there is a zero in a pool entry, the corresponding will pay price will be zero. Refunded entries will also have a zero price. Such entries will not be omitted.

Inter-Tote System Protocol

6. The **<winning>** field is the entire network's winning amount eligible for **<price>**. It will be filled in with the amount matching **<winners>** and **<runner list>** even when the last-position runner is scratched, to enable the remote system to display the network amount bet.
7. If **<rows>** and **<columns>** are zero, this is an empty message and the displays should be cleared. This form of the message should be sent if will pays are meant to be transmitted, but unusual conditions prohibit will pay calculations. The host should not clear the will pays for the remote system for reasons of local host display scheduling. The remote system normally will clear the will pay displays on its own schedule.
8. The host will transmit **<will pay>** message when it is deemed final, such as when the next to last race of the pool type is made official, and also whenever the will pays change as a result of a late scratch. The race number in the message header will be the pricing race for that pool type. The **<will pay>** message will also be sent during the link sequence if the current race is the pricing race for this pool type, and prices have not yet been calculated.

Ascii File Transfer

Syntax

<file transfer>	:= <file header> <file header><file data>
<file header>	:= <destination><filename><modification date><file size><file segments> <current segment><current segment size>
<destination>	:= <source> .H .R.
<source>	:= {<alpha>*3}
<filename>	:= {<alpha>*32}
<modification date>	:= <date><time>
<file size>	:= <whole amount>
<file segments>	:= <whole amount>
<current segment>	:= <whole amount>
<current segment size>	:= <whole amount>
<whole amount>	:= <amount> with restriction that amount may not have fractional portion
<file data>	:= {<ascii>*css}
<running_checksum>	:= {<octal>*5}
<ascii>	:= Standard ASCII codes from 20 ₁₆ - 7F ₁₆ , 0A ₁₆ (new line) for line termination, and 0C ₁₆ (form feed)

Semantics

1. Only Ascii files may be sent.
2. Files may only be sent before the first race is current for a program, or after all races in which the remote participates are priced and official.
3. Free file requests are not allowed, a system may only send a file. A file transfer sequence from the host would use the full message sequence of *Pending, Request, Data, Acknowledge*. A file transfer sequence from the remote would begin with *Pending* and host response would be *Acknowledge*. From that point, the host would initiate the message sequence *Request, Data, Acknowledge* for each segment required. Each <message type> (except a positive *Acknowledge*) must have a <data> portion in the message, containing at least the <file header>.
4. The <destination> is the ultimate destination of the file to be transmitted. Allowed entries are either the source name of the system that is directly connected, the reserved name “**H**” meaning send the file to the host for the program within the header, and “**R**” meaning broadcast the file to all remote systems. “**H**” may only be sent from a remote system, “**R**” may only be sent by systems acting as host.
5. The <filename> contains the name of the file to be transferred. Since this field is fixed size, it must be (trailing) blank filled for file names < 32 characters. The name should contain only alphanumeric characters or the underscore (“_”) or period (“.”) and should be considered case-insensitive.
6. The <modification date> field specifies the date and time which should be stored on the receiving system for when the file was last modified. It is intended that this should be used by system operators to compare to see if they have the correct version. If possible, do not change this modification date when copying. Note that this date should not be adjusted due to differences in system times.
7. The <file size> field specifies the total number of bytes contained in the file.
8. The <file segments> field is the total number of segments to be sent. The segment size is defined as 1024 characters. A system sends a n byte file by sending n / 1024 whole segments of 1024 characters, potentially followed by a last segment containing any remaining bytes.

9. The **<current segment>** field is the segment number being sent or requested, out of a total **<file segments>**. This field's range is 1 to **<file segments>**.
10. The **<current segment size>** field is the length of the data section, $0 < \text{<current segment size>} \leq 1024$.
11. The **<file data>** field is only included in the data portion of the File Transfer message. The **<file data>** field contains **<current segment size>** (css) bytes of text. Note that the end of line delimiter is defined here as newline only, and not carriage return/line feed.
12. The **<running_checksum>** is the sum of all **<file data>** characters within previously acknowledged segments, plus the sum of the **<file data>** portion of this message, modulo 100000_8 .

Appendix A - Variations for Italy

This describes the variations to and interpretations of ITSP useful for Italy:

1. Quarte scan messages

The Quarte involved adding (starting in ITSP 05.14) a new type of scan message to the chapters describing **Scan Request** and **Scan** messages. The following is a description of how this new format is used for the Quarte.

- 1.1 When the system operator at the host site enters the winners for the Quarte race, a message is immediately sent to the remote sites, requesting that they begin to collate the scan data for the Quarte. When a remote system is ready with this data, it responds with a message informing the host that scan data is pending. The host requests the scan data. When all remote sites have completed the transmission, the host system immediately calculates the Quarte prices. After the operations staff reviews the price reports, they enter a command to make the prices official, at which time all prices for the race are sent to the remote sites.
- 1.2 The Italian Quarte uses the Combination List Late scan (<scan mode> C) format, with <pool code> *SPR* (Superfecta). A maximum of 50 combinations are supported. (With a two-runner dead heat and two or more scratches, there would be 18 combinations.)

The following combinations may be present:

- Two normal winning combinations, more if there is a dead heat:

1st / 2nd / 3rd / 4th

1st / 2nd / 3rd / all live except 1st, 2nd, 3rd, and 4th

(the above combinations do not overlap, so no winner will match both.)

- If there are any scratches, four consolation winning combinations matching one scratch with 1st, 2nd, 3rd in any order:

all scratches / 1st,2nd,3rd / 1st,2nd,3rd / 1st,2nd,3rd

1st,2nd,3rd / all scratches / 1st,2nd,3rd / 1st,2nd,3rd

1st,2nd,3rd / 1st,2nd,3rd / all scratches / 1st,2nd,3rd

1st,2nd,3rd / 1st,2nd,3rd / 1st,2nd,3rd / all scratches

- Also for consolations, four combinations matching one scratch with three live, in any order, to get the consolation pool:

all scratches / all live / all live / all live

all live / all scratches / all live / all live

all live / all live / all scratches / all live

all live / all live / all live / all scratches

(The scratch-all combinations do overlap with the scratch-winner combinations, which is permissible since the scratch-all combinations will not appear in a payoff message.)

- If there are two or more scratches, six refund combinations that match two or more scratches with any other runners, in any order:

all scratches / all scratches / all scratches and live / all scratches and live
(the above includes: four scratches; three scratches followed by one live;
two scratches followed by two live)

all scratches / all live / all scratches / all scratches and live
(the above includes: all scratches / all live / all scratches / all live;
all scratches / all live / all scratches / all scratches)

all scratches / all live / all live / all scratches
all live / all scratches / all scratches / all scratches and live
(the above includes: all live / all scratches / all scratches / all scratches;
all live / all scratches / all scratches / all live)

all live / all scratches / all live / all scratches
all live / all live / all scratches / all scratches

If any of 1st and 2nd or 1st, 2nd, and 3rd are all members of the same bracket, the numbers will be repeated within each position, without adding more combinations. Example: There are 10 runners, and winners are 1 / 2 / 3 / 4, and both 1 and 2 are in a bracket. Winning combinations are (1,2 / 1,2 / 3 / 4) and (1,2 / 1,2 / 3 / 5-10).

- 1.3 The Quarte scan data works well, but has a relatively large amount of redundant data when many combinations are needed. Each message is required to contain the complete list of all combinations, while containing the data for just one of them. This is more an esthetic than an operational problem, but will be obvious when interpreting monitor logs of scan data transmissions. Changes could be made for a future ITSP version.
- 1.4 Quarte scan message change between ITSP 05.14 and ITSP 05.15: Due to the addition of the new field **<live total>** to each scan message, the Combination List Late scan (**<scan mode> C**) format has been changed. The total live amount must no longer be included as the amount in the second column of the matrix. See the description in the **Scan** message chapter.
2. Duplice Accoppiata pool and scan messages

The Duplice Accoppiata, which is an exchange Double Quinella / Double Exacta, uses an adaptation of the type of scan message proposed for the U.S. Quinella-Double. To accommodate this new scan mode, a new **<live total>** field has been added to the end of every scan message. Since such a change is not backward compatible with older levels, this enhancement is not available in levels older than ITSP 05.15. The new scan message format is explained in the chapters describing **Scan Request** and **Scan** messages. The following is a description of how the Duplice Accoppiata transmissions work.

- 2.1 During the first Duplice Accoppiata race a double leg pool message is used, using the same pool message format as a Quinella or Exacta would use. During the betting period for the second race, pool total messages may be transmitted, but no pool or scan messages are appropriate. After betting closes for the second race, nominally when the system operator enters the second race winners, a message is sent to the remote sites requesting that they begin to collate the scan data for the Duplice Accoppiata. When a remote system is ready with this data, it responds with a message informing the host that scan data is pending. The host requests the scan data. When all remote sites have completed the transmission, the host system immediately calculates the second race Duplice Accoppiata prices. After the operations staff reviews the price reports, they enter a command to make the prices official, at which time all prices for the race are sent to the remote sites.
- 2.2 The first Duplice Accoppiata race uses the double leg pool message with either **<pool code> ITQ** or **ITE**. The second Duplice Accoppiata race uses the Quinella Double scan (**<scan mode> Q**) format, with either **<pool code> ITQ** or **ITE**. The first and second race pool codes do not need to match.

In the scan messages, the winners of the first race are specified in the first two positions of the scan combinations and determine the number of combinations. All originally live second race runners are specified in the last two positions of each combination. If the first race pool code is *ITQ*, then the winners in the first two positions will be specified in ascending order.

When the **<payoff>** message is eventually received, each winning combination will contain four positions. The first two positions of each winning combination will exactly match a combination in the scan message so that the remote system will be able to find its own winning amount.

3. Pool Scaling

The **<pool scaling factor>**, added to ITSP 05.14 specifically for Italy, is not yet implemented. Therefore, the field has been eliminated. Instead, this level of ITSP implements the same method of encoding monetary values that is already used in ITSP 05.13. All data defined as **<amount>** fields are represented in thousands of Lire. For instance, L12000 is represented as "12" and L12345 is represented as "12.345". Thus, wherever the document mentions "dollars", it should read for Italy "Thousands of Lire".

In some future level of the protocol, **<pool scaling factor>** should be added again. It should then be defined for ease of implementation, just enabling the method described above. When Italian systems need to communicate with foreign systems (such as Canada or Hong Kong) this parameter should be in place so the foreign systems do not need special adaptations. At that time, additional new parameters such as currency identification will be equally important.

4. Price ratios

Prices in the **<payoff>** message are values in thousands of Lire as described above. However, it must also be noted that they represent the ratio per thousand Lire bet unit for all pool types except the second half Duplice Accoppiata, where the price is the ratio per 500 Lire unit. The protocol has no provision for explicitly communicating the denominator of the ratio.

Similarly, the **<min payoff>** and **<break>** fields in the **<configuration>** message are amounts that relate to the agreed upon price ratio.

Another problem in the second race of the Duplice Accoppiata is solved by two new **<status>** codes 'C' and 'Q' in the **<payoff>** message. When there is a scratched runner, holders of second half exchange tickets including the scratch are paid a consolation price equal to half of the first race "Will-Pay" price. Since the "Will-Pay" price was originally rounded down to 100 Lira per 1000 Lira bet, the second race consolation price is, in effect, rounded down to 50 Lira per 500 Lira unit bet. The "Will-Pay" price is also used when the second half exchange pool is cancelled, requiring that the price must be paid to all holders of second half tickets, and also to holders of first half winning tickets who failed to exchange them. The **<status>** codes 'C' and 'Q' indicate that the host already rounded the price, and the remote system must not round it again.

5. Commission rate percents

The **<commission rate>** values transmitted in the **<configuration>** message must be ignored, since the number of runners in each race determines the commission percent for certain pools. The remote system must receive the proper live runners and brackets so it can pick the correct commission rate percents for its own use.