INTERNATIONAL STANDARD

ISO/IEC 24730-1

Second edition
2014-02-15

Information technology — Real-time locating systems (RTLS) —

Part 1: Application programming interface (API)

Technologies de l'information — Systèmes de localisation en temps réel (RTLS) —

Partie 1: Interface de programmation d'application (API)

Reference number
ISO/IEC 24730-1:2014(E)

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 24730-1 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 31, *Automatic identification and data capture techniques*.

This second edition cancels and replaces the first edition (ISO/IEC 24730-1:2006), which has been technically revised.

ISO/IEC 24730 consists of the following parts, under the general title *Information technology — Real time locating systems (RTLS)*:

— *Part 1: Application programming interface (API)*

— *Part 2: Direct Sequence Spread Spectrum (DSSS) 2,4 GHz air interface protocol*

— *Part 21: Direct Sequence Spread Spectrum (DSSS) 2,4 GHz air interface protocol: Transmitters operating with a single spread code and employing a DBPSK data encoding and BPSK spreading scheme*

— *Part 22: Direct Sequence Spread Spectrum (DSSS) 2,4 GHz air interface protocol: Transmitters operating with multiple spread codes and employing a QPSK data encoding and Walsh offset QPSK (WOQPSK) spreading scheme*

— *Part 5: Chirp spread spectrum (CSS) at 2,4 GHz air interface*

— *Part 61: Low rate pulse repetition frequency Ultra Wide Band (UWB) air interface*

— *Part 62: High rate pulse repetition frequency Ultra Wide Band (UWB) air interface*

Introduction

ISO/IEC 24730 defines several air interface protocols and a single Application Programming Interface (API) for Real Time Locating Systems (RTLS) for use in asset management and is intended to allow for compatibility and to encourage interoperability of products for the growing RTLS market.

This part of ISO/IEC 24730, the RTLS Application Programming Interface, establishes a technical standard for Real Time Locating Systems. To be fully compliant with ISO/IEC 24730, Real Time Locating Systems (RTLS) shall comply with this part of ISO/IEC 24730 and at least one air interface protocol defined in ISO/IEC 24730.

Real Time Locating Systems are wireless systems with the ability to locate the position of an item anywhere in a defined space (local/campus, wide area/regional, global) at a point in time that is, or is close to, real time. Position is derived by measurements of the physical properties of the radio link.

Conceptually there are four classifications of RTLS:

— Locating an asset via satellite - requires line-of-sight - accuracy to 10 meters

— Locating an asset in a controlled area, e.g., warehouse, campus, airport - area of interest is instrumented - accuracy to 3 meters

— Locating an asset in a more confined area - area of interest is instrumented - accuracy to tens of centimetres

— Locating an asset over a terrestrial area using terrestrial mounted receivers over a wide area, cell phone towers for example – accuracy 200 meters

With a further two methods of locating an object which are really RFID rather than RTLS:

— Locating an asset by virtue of the fact that the asset has passed point A at a certain time and has not passed point B

— Locating an asset by virtue of providing a homing signal whereby a person with a handheld can find an asset

Method of location is through identification and location, generally through multi-lateration, of various types

— Time of Flight Ranging Systems

— Amplitude Triangulation

— Time Difference of Arrival (TDOA)

— Cellular Triangulation

— Satellite Multi-lateration

— Angle of Arrival

The location information of an asset may further be enhanced with information on its spatial orientation.

This part of ISO/IEC 24730 defines an application programming interface (API) needed for utilizing an RTLS system.

An API is a boundary across which application software uses facilities of programming languages to invoke services. These facilities may include procedures or operations, shared data objects and resolution of identifiers. A wide range of services may be required at an API to support applications. Different methods may be appropriate for documenting API specifications for different types of services.

The information flow across the API boundary is defined by the syntax and semantics of a particular programming language, such that the user of that language may access the services provided by the application platform on the other side of the boundary. This implies the specification of a mapping of the functions being made available by the application platform into the syntax and semantics of the programming language. An API specification documents a service and/or service access method that is available at an interface between the application and an application platform.

This API describes the RTLS service and its access methods, to enable client applications to interface with the RTLS system. This RTLS service is the minimum service that shall be provided by a RTLS system to be API compatible with this standard.

This part of ISO/IEC 24730 uses a "full stop" as the decimal point separator since an API file is being created with an output in a .csv file format which uses the comma to separate values.

Information technology — Real-time locating systems (RTLS) —

Part 1: Application programming interface (API)

1 Scope

This part of ISO/IEC 24730 enables software applications to utilize an RTLS infrastructure to locate assets with RTLS transmitters attached to them. It defines a boundary across which application software uses facilities of programming languages to collect information contained in RTLS tag blinks received by the RTLS infrastructure.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 15963, *Information technology — Radio frequency identification for item management — Unique identification for RF tags*

ISO/IEC 19762 (all parts), *Information technology — Automatic identification and data capture (AIDC) techniques — Harmonized vocabulary*

IEEE Guidelines for use of a 48-bit Extended Unique Identifier (EUI-48™)

IEEE Guidelines for 64-bit Global Identifier (EUI-64™) Registration Authority

Extensible Markup Language (XML) 1.0, (Third Edition), W3C Recommendation, World Wide Web Consortium (W3C), 4 February 2004[1)]

SOAP Version 1.2 Part 1: Messaging Framework (Second Edition), W3C Recommendation, World Wide Web Consortium (W3C), 27 April 2007[2)]

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 19762 (all parts), and the following apply

3.1
field
element of a data record in which information is stored, which may contain one or more properties of a tag blink

3.2
XML tag
marker that qualifies content in a XML document

1) http://www.w3.org/TR/REC-xml/

2) http://www.w3.org/TR/2007/REC-soap12-part1-20070427/

3.3
persistent connection
network connection between a server and a client that is kept open for several application level message exchanges, or request call, even after sending application level error responses

3.4
tag status
mandatory fields within a Locate message not including the <source> and <format> fields

4 Symbols and abbreviated terms

For the purposes of this document, the symbols and abbreviated terms given in ISO/IEC 19762 (all parts) and the following apply.

API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
CR	ASCII Carriage Return
EUI	Extended Unique Identifier
JMS	Java Messaging Service
HTTP	HyperText Transfer Protocol
HTTPS	HTTP Secure Protocol
LF	ASCII Line Feed
OUI	Organizationally Unique Identifier
REST	Representational State Transfer
RTLS	Real Time Locating System
S-HTTP	Secure HTTP Protocol
SLMF	Simple Location Message Format
SLMP	Simple Location Message Protocol
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
TDOA	Time Difference Of Arrival
TCP/IP	Transmission Control Protocol / Internet Protocol
XML	eXtensible Markup Language
XSD	XML Schema Definition

5 The service

5.1 Purpose

The purpose of the RTLS software API is to provide a simple minimal interface that facilitates adoption and can be implemented easily on both the RTLS system and the application. The purpose is also to allow fast transfer of messages to any client that connects to the RTLS system. In addition, the message flow should be human-readable and easy to interpret.

The API is to be supported by a device of minimal functionality, which is a ‘collection and forwarding device’, with no persistency or database required.

The device can provide rate filtering and location smoothing capabilities, but these functions are not required by the proposed API because the API can operate before or after this type of pre-processing.

5.2 Specification summary

The RTLS software service shall be a ‘Text over Socket’ connection, such that a TCP/IP client can connect and receive a real-time stream of RTLS messages.

The messages are separated by <CR><LF> to facilitate console display. The field format is comma-separated.

The ‘Text over Socket’ protocol is the minimal mandatory compliance. If additional transport protocols are implemented such as HTTP and JMS, then either Text or XML[3] formats shall be used. If REST or SOAP[4] is implemented, then an XML format is used. The Text format includes comma-separated fields, while the XML format includes ‘abbreviated’ XML tags. Both formats are described within this part of ISO/IEC 24730.

REST, SOAP and higher functionality services are not mandatory because they tend to limit the message rate.

A goal of this part of ISO/IEC 24730 is to easily support 3,000 messages per second or more. Although in many applications a rate of 3,000 messages per second may not be needed, the RTLS minimal API shall support it because an actual tag deployment with 3,000 tags blinking at 1 Hz may easily produce that message rate. When rate filtering is involved, and/or the IT systems and software are tuned to handle high rates, REST, SOAP and other transports may provide additional functionality. As indicated, the comma-separated Text format shall be considered when supporting high data rates because the format is non-verbose and allows transports to operate at medium to high rates.

This API or protocol is referred to as ‘Simple Location Message Protocol’, or ‘SLMP’. The mandatory comma-separated format is referred to as ‘Simple Location Message Format’, or ‘SLMF’. For specific transports, ‘SLMF-Sockets’ is the TCP/IP compatible mandatory interface/API of SLMP, while, for example, SLMF-HTTP is an optional interface/API when supporting HTTP as a simple RTLS REST service. As indicated above, an SLMP implementation may optionally include an XML format.

A client application connects to the RTLS using a TCP/IP connection. The RTLS system responds with a stream of messages that stops only when the client connection is closed.

The RTLS system shall send keep-alive massages if the line is silent for long periods of time. This part of ISO/IEC 24730 does not prescribe a particular keep-alive interval, but rather leaves this decision to the RTLS system vendor (see clause 5.7.3). The Client App shall attempt a reconnect periodically if the Socket connection to the RTLS system is lost.

The RTLS system provides the API via a minimal device that collects message from readers, calculates locations, and forwards messages. This device is not required to have persistency and does not need to keep historical data or state of last tag during the active session. Thus, this API does not provide tag status, but rather only tag events. Tag status is left to the application to handle in the context of a business aware database, or for a higher level API not in this scope.

The API herein defined shall support multiple concurrent client connections.

3) *Extensible Markup Language (XML) 1.0, (Third Edition)*, W3C Recommendation, World Wide Web Consortium (W3C), 4 February 2004. (http://www.w3.org/TR/REC-xml/)

4) *SOAP Version 1.2 Part0: Primer,* W3C Recommendation, World Wide Web Consortium (W3C), 24 June 2003, (http://www.w3.org/TR/2003/REC-soap12-part0-20030624/)

5.3 Message stream configuration

This part of ISO/IEC 24730 does not prescribe specific methods for filtering the output message stream produced by the RTLS system; however, an RTLS system vendor may implement filtering on a data collection and forwarding device if desired while still complying with this part of ISO/IEC 24730. For example, the vendor could implement a client-side subscription method that passes filter definitions to the RTLS system. The RTLS system could then use the filter definitions to limit message stream output to clients. Alternatively, a vendor could implement server-side configuration of filter definitions that apply to all client connections.

5.4 Security

Security protocols regarding RTLS message exchange are intentionally excluded from this part of ISO/IEC 24730 because security can be addressed using existing security standards and technologies at the communication layer based on preferences and policies of individual customers. For example, in the case of a TCP/IP connection SSH is a security protocol that is easy to implement and is considered compliant with this standard. Likewise, security protocols such as HTTPS and S-HTTP are security protocols that may be implemented on top of this part of ISO/IEC 24730.

5.5 Purpose

The API herein defined provides a standard mechanism for client application to access location-enriched tag blinks from an RTLS system.

5.6 Language independence

The API herein defined specifies a software language-independent interface to the RTLS Service. It does so by using an industry standard protocol, Text over Socket (TCP/IP), to communicate to the RTLS service.

5.7 Architecture

Figure 1 describes the API message exchanges between a client application and the RTLS System. The 24730-1 API shall allow multiple client connections, thus it keeps TCP/IP connection state per client.

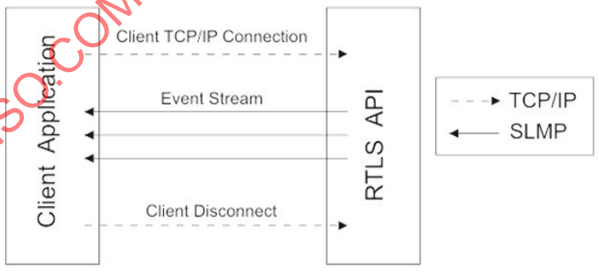

Figure 1 — Architecture of SLMP

5.8 SLMP messages

This clause describes messages that are used within this part of ISO/IEC 24730. Each message type includes a set of fields, which the RTLS vendor shall implement in accordance with the definitions provided herein. All field values and XML tag names that are explicitly defined in this part of ISO/IEC 24730 are case sensitive.

5.8.1 Data types

Data types described in this clause pertain to fields that are associated with messages defined within this part of ISO/IEC 24730. For the Locate message, an RTLS vendor may optionally include extension fields that are not described within this part of ISO/IEC 24730. For such fields, the vendor may choose data types of their choice.

DateTime

This data type represents a date time format as defined in ISO 8601: YYYY-MM-DDThh:mm:ss-hh:mm

Year in the form **YYYY**-MM-DD
Month in the form YYYY-**MM**-DD
Day in the form YYYY-MM-**DD**

T indicates "**T**ime will follow"

Hours in the form **hh**:mm:ss
Minutes hh:**mm**:ss
Seconds hh:mm:**ss**

Plus or minus UTC offset in hours and minutes **(-hh:mm or +hh:mm)**

Example: 2010-11-24T09:07:04-08:00   //for PST time zone

Note that a fraction with up to one-tenth millisecond (.0001 seconds) accuracy may be added to the lowest order time element in the representation. For example, to denote "14 hours, 30, minutes, and 12.359 seconds", represent it as "14:30:12.359".

Double

This data type represents a floating-point format that includes an encoded optional decimal point, and may be expressed with or without the exponent and mantissa. Examples include: 2345.334, -98.7, 1.0, 4, 0.0, 0.5, 9.87+E8.

The range for a field of type 'Double' is 1.7E–308 to 1.7E+308, and the maximum string length is 256 characters.

HexBinary

This data type represents structured or unstructured data that can be expressed in hexadecimal format, where each byte is a binary octet. The high order nibble is expressed as the first (leftmost) nibble within an octet, and each HexBinary string shall contain an even number of nibbles.

The maximum field length for a field of type 'HexBinary' is 256 bytes.

Integer

This data type represents numbers that can be written without a fractional or decimal component, and fall within the set {..., −2, −1, 0, 1, 2, ...}.

The range for a field of type 'Integer' is – 2,147,483,648 to 2,147,483,647.

String

This data type represents a set of ASCII characters, limited to the following characters:

A, B, C, D, E, F, G, H, I, J, K, L, M, N ,O, P, Q, R, S, T, U, V, W, X, Y, Z, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, space, !, (,), [,], *, #, $, %, &, +, -, _, ., /, ?, =

The maximum field length for a field of type 'String' is 256 characters.

5.8.2 Header Message

The RTLS system sends a single Header message when a client application establishes a connection to it.

Fields Sequence:

<Appliance_ID>,SLMF,<SLMF_version>,<SLMF_vendor_version>,<Greeting><CR><LF>

Fields:

Appliance_ID (Mandatory)

XML Tag:	<aid>
Data Type:	String
Restriction:	1 to 10 characters

Represents the RTLS system or appliance that produces the Locate messages.

SLMF_version (Mandatory)

XML Tag:	<ver>
Data Type:	String
Restriction:	1 to 10 characters

Version of the SLMF implementation, assigned by this part of ISO/IEC 24730. The SLMF version for this version of this part of ISO/IEC 24730 is **1.0**.

SLMF_vendor_version (Mandatory)

XML Tag:	<vver>
Data Type:	String
Restriction:	1 to 10 characters

Vendor version of the SLMF implementation, which is used when a vendor provides one or more non-default formats associated with Locate messages. If vendor version is not used, this field shall remain empty.

Greeting (Mandatory)

XML Tag:	<greet>
Data Type:	String
Restriction:	1 to 256 characters

Message is defined by the RTLS vendor, which can represent a greeting.

Example:

MyAppl,SLMF,1.0,1.3,Welcome to the RTLS Text Stream interface.<CR><LF>

5.8.3 Field Definition Message

Field Definition messages are sent immediately after the Header message and represent fields that are used within Locate messages sent by the RTLS system. Locate messages include mandatory and optional fields with definitions that shall be adhered to when these fields are implemented by the RTLS system vendor.

Fields Sequence:

FieldDefinition,<Name>,<Type><CR><LF>

Fields:

Name (Required)

XML Tag:	<nam>
Data Type:	String
Restriction:	1 to 256 characters

The name of the data field that is associated with one or more Locate messages. If the field is explicitly defined within clause 5.8, then the value for <Name> within the Field Definition message shall match the name of the field defined within clause 5.8. When XML is used, the field name shall match the XML tag specified within clause 5.8.

Type (Required)

XML Tag:	<typ>
Data Type:	String
Restriction:	1 to 256 characters

The data type associated with the field. If the field is defined within clause 5.8, then the value for <Type> within the Field Definition message shall match the data type associated with the related field within clause 5.8.

Example:

FieldDefinition,Source,String<CR><LF>

5.8.4 Locate Message Definition Message

Locate Message Definition messages are sent immediately after the Field Definition messages. One Locate Message Definition message shall be defined for each unique <Source>,<Format> pair.

Fields Sequence:

LocateMessageDefinition,<Source>,<Format>,<Field1>,<Field2>,<Field3>,...<CR><LF>

Fields:

Source

XML Tag:	<src>
Data Type:	String
Restriction:	1 to 64 characters

Field Definition: FieldDefinition,Source,String<CR><LF>

Source typically represents a particular location technology or product line. For example, if an RTLS system produces Locate messages that are derived from Product Line A and Product Line B, the RTLS system vendor may specify a source value for each of the two product lines. Source values are defined by RTLS system vendors; however, the RTLS system vendor may provide a method that allows customers to optionally map alternative source values of their choice. This part of ISO/IEC 24730 does not prescribe a specific source value mapping method; this is left to each RTLS system vendor to decide.

Format (Mandatory)

XML Tag: <fmt>
Data Type: String
Restriction: 1 to 64 characters

Field Definition: FieldDefinition,Format,String<CR><LF>

Format represents the set of fields contained in the Locate message. If a Locate message contains non-mandatory fields, the format and source field combination shall be used by the client application to determine how to parse the message; otherwise, the Format value shall be "DFT" indicating that only mandatory fields are contained within the Locate message. Format values other than "DFT" are defined by RTLS system vendors.

Field Names (Mandatory)

See clause 5.8.6 for names of fields that pertain to Locate messages. An RTLS vendor may specify their own field names for extension fields that are not explicitly defined within clause 5.8.6.

LocateMessageDefinition,<Source>,<Format>,Tag_ID_Format,Tag_ID,X,Y,Z,Battery,Timestamp,Ext1,Ext2, Ext3,...<CR><LF>

Examples:

LocateMessageDefinition,MySourceA,DFT,Tag_ID_Format,Tag_ID,X,Y,Z,Battery,Timestamp<CR><LF>

LocateMessageDefinition,MySourceB,DFT,Tag_ID_Format,Tag_ID,X,Y,Z,Battery,Timestamp<CR><LF>

LocateMessageDefinition,MySourceB,S,Tag_ID_Format,Tag_ID,X,Y,Z,Battery,Timestamp,Algorithm<CR><LF>

LocateMessageDefinition,MySourceB,T,Tag_ID_Format,Tag_ID,X,Y,Z,Battery,Timestamp,Data<CR><LF>

5.8.5 Keep-Alive Message

The RTLS system may optionally include configuration that causes a keep-alive message to be sent if the elapsed time since the previous Locate message exceeds a specified duration. The Keep-Alive message is also sent each time the API client establishes a connection with the RTLS system.

Fields Sequence:

KeepAlive,<Period><CR><LF>

Fields:

Period (Required)

XML Tag: <per>
Data Type: Integer
Restriction: 1 to 3600 characters

Time duration, in seconds, that triggers a Keep-Alive message when the line is silent. This value is set within the RTLS system.

Example:

KeepAlive,60<CR><LF>

5.8.6 Locate Message

An RTLS tag event is typically expressed as a Locate message containing mandatory fields, plus any number of optional fields that are also referred to as “extensions”. RTLS system vendors may define their own extension fields and data types if they do not exist within this clause. If the RTLS system is unable to determine the value of <x>, <y>, or <z>, the unknown field(s) shall remain empty.

The maximum length of a Locate message shall be 2^14 (or 16,384) characters.

Fields Sequence:

<Source>,<Format>,<Tag_ID_Format>,<Tag_ID>,<X>,<Y>,<Z>,<Battery>,<Timestamp>,<Ext1>,<Ext2>,<Ext3>...<CR><LF>

Fields:

Source (Mandatory)

See clause 5.8.4 for field definition. For this message, the Source / Format pair shall match the Source / Format pair of a Locate Message Definition message.

Format (Mandatory)

See clause 5.8.4 for field definition. For this message, the Source / Format pair shall match the Source / Format pair of a Locate Message Definition message.

Tag_ID_Format (Mandatory)

XML Tag:	<idfmt>
Data Type:	HexBinary
Restriction:	1 byte

Field Definition: FieldDefinition,Tag_ID_Format,HexBinary<CR><LF>

Indicates the format used for the serial number portion of the Tag ID. The Tag_ID_Format specified in this part of ISO/IEC 24730 shall be one of the values listed in Table 1.

Table 1 — Tag ID formats

Tag_ID_Format	Format
0x01	ISO/IEC 15963
0x02	IEEE EUI-48
0x03	IEEE EUI-64

Tag_ID (Mandatory)

XML Tag:	<tid>
Data Type:	HexBinary
Restriction:	Variable Length

Field Definition: FieldDefinition,Tag_ID,HexBinary<CR><LF>

The unique identifier of the tag. The Tag ID is variable length, depending on the Tag_ID_Format. See below for examples of Tag_ID examples based on Tag_ID_Format.

ISO/IEC 15963

Represents the 48-bit ISO/IEC 15963 format, which includes an allocation class, manufacturer ID, and serial number. The allocation class and manufacturer ID are defined in ISO/IEC 15963 and represent the first 16 bits of the Tag ID. The serial number is 32 bits. Suppose that the Tag_ID_Format is 0x01. If the tag's allocation class is 0x00, manufacturer Id is 0x01, and serial number is 0x00BC614E, then the Tag_ID field shall be 0x000100BC614E and expressed within the message as *000100BC614E*.

IEEE EUI

Represents the 48-bit or 64-bit IEEE EUI format, which includes an OUI and serial number. OUIs are 24 bits long and administered by IEEE. The serial number is 24 bits long for IEEE EUI-48, and 40 bits long for IEEE EUI-64. Suppose that the Tag_ID_Format is 0x02. If the tag's OUI is 0x00003A and serial number is 0x01E64A, then the Tag_ID field shall be represented as 0x00003A01E64A and expressed within the message as *00003A01E64A*.

X, Y, Z (Mandatory)

XML Tag:	<x>, <y>, <z>
Data Type:	Double

Field Definition:	FieldDefinition,X,Double<CR><LF>
Field Definition:	FieldDefinition,Y,Double<CR><LF>
Field Definition:	FieldDefinition,Z,Double<CR><LF>

Location of the tag is relative to a known point. The location units are expressed as Cartesian coordinates, in meters or fractions of meters and may be negative or positive values.

In the event that a tag is detected but its Cartesian coordinates cannot be determined, then the values for <X>, <Y>, or <Z> shall be omitted from the Locate message; however, the Locate message shall still include the comma delimiters for the coordinates.

Battery (Mandatory)

XML Tag:	<bat>
Data Type:	Integer
Restriction:	0, 1, 2, or 3

Field Definition: FieldDefinition,Battery,Integer<CR><LF>

The integer values are as indicated in Table 2.

Table 2 — Battery States

Value	State
0	Good
1	In need of replacement
2	Reserved for use by this standard
3	Battery status unknown

Timestamp (Mandatory)

XML Tag:	<t>
Data Type:	DateTime

Field Definition: FieldDefinition,Timestamp,DateTime<CR><LF>

Timestamp represents the date and time of when the location of the tag was determined.

Classification (Optional)

XML Tag:	<cls>
Data Type:	String
Restriction:	1 to 64 characters

Field Definition: FieldDefinition,Classification,String<CR><LF>

Classification is a unique representation of a tag population based on a common set of attributes associated with the function of the tags or the assets they are attached to. This field is useful when an application needs to treat asset classes differently, or when the data collection device needs to apply special logic to a tag based on its type before publishing through the API. The data collection device may include a method for assigning tags to classifications, whose implementation would be decided by the RTLS system vendor.

For example, a classification could be 'Asset', and another classification could be 'Visitor'. Additional classification examples include: 'Trailer', 'Tractor', 'Container', 'Pallet', and 'Forklift'.

The Classification field is optional because in some applications the tag classifications are connected to business logic. An example of this is a marine terminal application where a vehicle that handles equipment has multiple tags, and the classifications are connected to a specific role the tag performs on that particular vehicle type. Thus, the application deals with vehicle types, and tags are just part of the vehicle instrumentation.

Zone (Optional)

XML Tag:	<zon>
Data Type:	String
Restriction:	1 to 256 characters

Field Definition: FieldDefinition,Zone,String<CR><LF>

The Name of a Zone, such as a parking spot, a building, or a room. Zone may also be expressed in the form of a path, such as Facility/Building/Room.

Typically a Zone is calculated from x,y,z by checking which zone includes the point x,y,z. The list of Zones is a list of named polygons.

Zone is optional, because in some applications the zone definitions are connected to business logic, and zone assignment and zone look up are derived at the application level. An example of this is a marine terminal operation system, where the container locations are defined within each stack by row-bay-tier, and thus zones become very specialized.

Exciter_ID (Optional)

XML Tag:	<exc>
Data Type:	String
Restriction:	1 to 64 characters

Field Definition: FieldDefinition,Exciter_ID,String<CR><LF>

An exciter is a device that makes a tag nearby blink. The tag's blink message may include Exciter_ID in the message payload. The Exciter_ID is typically an integer, but may also be an IP address, or a DNS name.

Antenna_ID (Optional)

XML Tag:	<ant>
Data Type:	Integer
Restriction:	0 to 255

Field Definition: FieldDefinition,Antenna_ID,Integer<CR><LF>

Antenna_ID typically applies to passive RFID and represents a physical read point within a location coverage area. When RFID antennas are used, the tag's location may be inferred by the known location of the fixed antenna.

Data (Optional)

XML Tag:	<dat>
Data Type:	HexBinary
Restriction:	1 to 123 bytes

Field Definition: FieldDefinition,Data,HexBinary<CR><LF>

This field contains unstructured data transmitted by a tag that can be decoded by an API client. For example, sensors that are connected to a tag might pass temperature, fuel level, and engine status wirelessly to the RTLS system. Typically an RTLS system will decode unstructured data before sending to the client, but there may be some instances where it is appropriate for decoding to occur at the application level.

Algorithm (Optional)

XML Tag:	<alg>
Data Type:	String
Restriction:	1 to 64 characters

Field Definition: FieldDefinition,Algorithm,String<CR><LF>

Vendor-defined formulas used to arrive at the X,Y,Z location. For example, a value of 'P' may indicate 'Presence' when the tag location is derived from a single location sensor.

A, B, C (Optional)

XML Tag:	<a>, <b>, <c>
Data Type:	Double
Restriction:	A and C are radians in the interval 0 to 2π, and B is a radian in the interval of 0 to π.

Field Definition: FieldDefinition,A,Double<CR><LF>

Field Definition: FieldDefinition,B,Double<CR><LF>

Field Definition: FieldDefinition,C,Double<CR><LF>

Spatial three-dimensional orientation of the tag. The orientation is represented by Euler angles as an intrinsic rotation relative to a frame of reference in a 3-dimensional Euclidean space. The three Euler angles A, B, and C unambiguously define a composition of three rotations. This specification requires the use of the Tait–Bryan convention, decomposing the rotation into three subsequent intrinsic rotations about the axes X (first rotation, about the original X-axis), Y' (second rotation, about the new orientation Y' of the moved Y-axis), and Z'' (third rotation, about the new orientation Z'' of the twice moved Z-axis). In other words, the X-Y'-Z'' convention is to be used, where the angles A, B, and C describe the exact Euler angles of the target frame relative to the reference frame.

With respect to the API, A and C are modulo 2π radians, and B is a radian within range [0, π].

Examples of non-extended Locate messages with location:

MySourceA,DFT,01,000100BC614E,100,150,8,0,2010-11-24T09:07:04-08:00<CR><LF>

MySourceB,DFT,02,000040E60A11,53,-40.3,1.5,0,2010-11-24T09:07:04-08:00<CR><LF>

Example of non-extended Locate messages without location:

MySourceB,DFT,02,000040E60A11,,,,0,2010-11-24T09:07:04-08:00<CR><LF>

Examples of extended Locate messages:

MySourceA,**S**,01,000100BC614E,24,903,8,0,2010-11-24T09:07:04-08:00,**3D**<CR><LF>

MySourceA,**T**,01,000100BC614E,100,150,8,0,2010-11-24T09:07:04-08:00,**2D**,**0FE321AB**<CR><LF>

The format field may be used to indicate a particular field extension or set of field extensions. In the examples above, the message with format = ‘S’ represents an extended message with a single optional field, while the message with format = ‘T’ represents an extended message with two optional fields. Both format values are defined by the RTLS vendor since they describe messages that include extension fields.

5.8.7 Message Sequence Example

The message sequence below represents an example of messages sent by the RTLS system after a client application establishes a connection.

```
MyAppl,SLMF,1.0,Welcome to the RTLS Text Stream Interface.<CR><LF>
FieldDefinition,Source,String<CR><LF>
FieldDefinition,Format,String<CR><LF>
FieldDefinition,Tag_ID_Format,HexBinary<CR><LF>
FieldDefinition,Tag_ID,HexBinary<CR><LF>
FieldDefinition,X,Double<CR><LF>
FieldDefinition,Y,Double<CR><LF>
FieldDefinition,Z,Double<CR><LF>
FieldDefinition,Battery,HexBinary<CR><LF>
FieldDefinition,Timestamp,DateTime<CR><LF>
FieldDefinition,Algorithm,String<CR><LF>
FieldDefinition,Data,HexBinary<CR><LF>
FieldDefinition,A,Double<CR><LF>
FieldDefinition,B,Double<CR><LF>
FieldDefinition,C,Double<CR><LF>
LocateMessageDefinition,MySourceA,DFT,Tag_ID_Format,Tag_ID,X,Y,Z,Battery,Timestamp<CR><LF>
LocateMessageDefinition,MySourceA,S,Tag_ID_Format,Tag_ID,X,Y,Z,Battery,Timestamp,Algorithm<
CR><LF>
LocateMessageDefinition,MySourceA,T,Tag_ID_Format,Tag_ID,X,Y,Z,Battery,Timestamp,Algorithm,
Data<CR><LF>
LocateMessageDefinition,MySourceC,SP,Tag_ID_Format,Tag_ID,X,Y,Z,Battery,Timestamp,Algorithm
,Data,A,B,C<CR><LF>
KeepAlive,30<CR><LF>
MySourceA,DFT,01,000100BC614E,100,150,8,1,2010-11-24T09:07:04-08:00<CR><LF>
MySourceA,S,01,000100BC614E,100,150,8,0,2010-11-24T09:07:04-08:00,2D<CR><LF>
MySourceA,S,01,000100BC614E,100,150,8,0,2010-11-24T09:07:04-08:00,L<CR><LF>
MySourceA,T,01,000100BC614E,100,150,8,0,2010-11-24T09:07:04-08:00,2D,0A46E137<CR><LF>
MySourceA,S,01,000100BC614E,100,150,8,0,2010-11-24T09:07:04-08:00,P<CR><LF>
MySourceC,SP,01,000100BC614E,100,150,8,0,2010-11-24T09:07:04-08:00,3D,B7F349,3.14,0.785,0.
0<CR><LF>
```

5.9 Simple Location Message Format over HTTP

5.9.1 Purpose

The objective of SLMF-HTTP is to provide a simple and fast message stream API that is easy to integrate with enterprise systems, and with cloud applications via HTTP. The intent is to preserve the field definitions of SLMF and a high data rate to the extent possible.