INTERNATIONAL STANDARD

ISO/IEC 24730-1

First edition
2006-02-15

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

Part 1: Application program 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 24730-1:2006(E)

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

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 and IEC shall not be held responsible for identifying any or all such patent rights.

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

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

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

— *Part 2: 2,4 GHz air interface protocol*

The following part is under preparation:

— *Part 3: 433 MHz air interface protocol*

Introduction

ISO/IEC 24730 defines two air interface protocols and a single application program 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 program interface, establishes a technical standard for RTLS. To be fully compliant with this standard, RTLS must 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 m.
- Locating an asset in a controlled area, e.g., warehouse, campus, airport (area of interest is instrumented) - accuracy to 3 m.
- 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 a terrestrial mounted receiver over a wide area, cell phone towers for example – accuracy 200 m.

There are 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 beacon whereby a person with a handheld can find an asset.

The method of location is through identification and location, generally through multi-lateration. The different types are

- Time of Flight Ranging Systems,
- Amplitude Triangulation,
- Time Difference of Arrival (TDOA),
- Cellular Triangulation,
- Satellite Multi-lateration,
- Angle of Arrival.

This part of ISO/IEC 24730 defines an API needed for utilizing an RTLS.

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. This RTLS service is the minimum service that must be provided by an RTLS to be API compatible with this standard.

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

Part 1:
Application program interface (API)

1 Scope

This part of ISO/IEC 24730 enables software applications to utilize a real-time locating system (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 referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 19762-1, *Information technology — Automatic identification and data capture (AIDC) techniques — Harmonized vocabulary — Part 1: General terms relating to AIDC*

ISO/IEC 19762-3, *Information technology — Automatic identification and data capture (AIDC) techniques — Harmonized vocabulary — Part 3: Radio frequency identification (RFID)*

ISO/IEC 9075:2003 (all parts), *Information technology — Database languages — SQL*

IETF RFC 2616: June 1999, *Hypertext Transfer Protocol — HTTP/1.1* (http://www.ietf.org/rfc/rfc2616.txt)

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/)

XML Schema Part 1: Structures, W3C Recommendation, World Wide Web Consortium (W3C), Cambridge Massachusetts, 2 May 2001. (http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/)

XML Schema Part 2: Datatypes, W3C Recommendation, World Wide Web Consortium (W3C), Cambridge Massachusetts, 2 May 2001. (http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/)

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/)

SOAP Version 1.2 Part1: Messaging Framework, W3C Recommendation, World Wide Web Consortium (W3C), 24 June 2003. (http://www.w3.org/TR/2003/REC-soap12-part1-20030624/)

SOAP Version 1.2 Part2: Adjuncts, W3C Recommendation, World Wide Web Consortium (W3C), 24 June 2003. (http://www.w3.org/TR/2003/REC-soap12-part2-20030624/)

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 19762-1, ISO/IEC 19762-3 and the following apply.

3.1
RTLS transmitters
active (battery powered) radio transmitters that are attached to assets (items, people, etc.)

NOTE RTLS transmitters send messages containing a unique ID for the asset, and may provide certain status information about the RTLS transmitter (e.g. battery level) and the asset (e.g. temperature).

3.2
tag blink
series of transmissions communicating a single asset movement or status change

NOTE Within the API, a tag blink consists of a "TagID" property that uniquely identifies the RTLS transmitter, as well as numerous other properties. Within the API, each property is also referred to as a field.

3.3
field
element of a data record in which information is stored, which may contain one or more properties of a tag blink, and which has a unique XML tag associated with it

NOTE Each tag blink property has a single, corresponding field in each TagBlink *XML* message. Conversely, within this API, each field may have one or more properties.

3.4
XML tag
marker that qualifies content in an XML document

3.5
SOAP
lightweight protocol intended for exchanging structured information in a decentralized, distributed environment

NOTE See references to SOAP Version 1.2 in the bibliography for additional information.

3.6
service
software program provides responses to requests from other software programs, which are frequently on other remotely connected computers

3.7
persistent connection
network connection between a webserver and a client that is kept open even after sending error responses

3.8
real-time
actively monitored with sub-minute updates for asset movement or status change

4 Symbols and abbreviated terms

For the purposes of this document, the symbols and abbreviated terms given in ISO/IEC 19762-1, ISO/IEC 19762-3 and the following apply.

HTTP	HyperText Transfer Protocol
RPC	Remote Procedure Call
RTLS	Real Time Locating System
XML	eXtensible Markup Language

5 The service

5.1 Purpose

The RTLS software service shall be a Web Service. It receives tag blinks from the RTLS infrastructure, and delivers those blinks as the response to client requests, over standard Internet protocols. In addition, it allows filtering and field selection on the contents of those blinks.

5.2 Specification summary

- An RTLS service shall support message exchange using the SOAP Version 1.2 encoding.

- An RTLS service should listen and respond to client requests using the HTTP 1.1 network protocol. If this network protocol is used, then the service shall conform to the IETF RFC 2616. It should listen on port 80, or port 443 (for secure communication).

- An RTLS service may take advantage of the persistent connections feature, which is available in HTTP 1.1. This would yield significant performance benefits for repeated calls to the RTLS Service, as described in Clause 7.4.

- An RTLS service shall maintain state of the last blink of all tags in the RTLS infrastructure, since the service was started. It may maintain state, from before its start time, by using a repository.

- An RTLS service shall support filtering and field selection, as described in Clause 7.2.5 and Clause 7.2.6. An RTLS service shall eliminate extraneous spaces in the content of query parameters, such as filters, as described in Clause 7.2.5.

- An RTLS service shall be able to create a session, on request from the client, as described in Clause 7.3.

- Each newly, created session shall keep state of the query parameters, such as the filter criteria, for utilization in subsequent requests to that session.

- Each request to that session shall return all tag blinks that match the filter criteria, and which occurred subsequent to the last request, as described in the Clause 7.4.

- If no tag blinks match the criteria, an RTLS service may delay response to session-based requests, until such tag blinks are received, or some pre-determined period has passed.

- An RTLS service shall remove the session state, when the CloseSession request is received, as described in Clause 7.5.

- An RTLS service may close the session, when a QuerySession request is not received after some predetermined period of time. This optional, predetermined period (ExpirationSeconds) may be defined by each vendor, in the SessionResponse structure, as described in Clause 8.3.

- An RTLS Service shall limit the length of all string values to a 1000 characters.

5.3 Security

The API uses SOAP Version 1.2, which typically assumes standard Internetworking protocols (such as HTTP 1.1 and TCP/IP). Security issues regarding RTLS message exchange can be addressed using existing, security standards and technologies at the communication layer. Therefore, security of the RTLS message exchange is beyond the scope of this standard.

6 The Application Program Interface (API)

6.1 Purpose

The Application Program Interface (API) provides a standard mechanism for accessing location-enriched, tag blinks from the RTLS Service from a client application.

6.2 Language Independence

This API specifies a language independent interface to the RTLS Service. It does so by using an industry standard protocol, SOAP Version 1.2, to communicate to the RTLS service.

6.3 Architecture

Figure 1 describes all the API message exchanges between a client application and the RTLS Service. The RTLS service shall not keep state regarding a stateless RPC request (Query), but it shall keep state of a session-based RPC request (OpenSession).

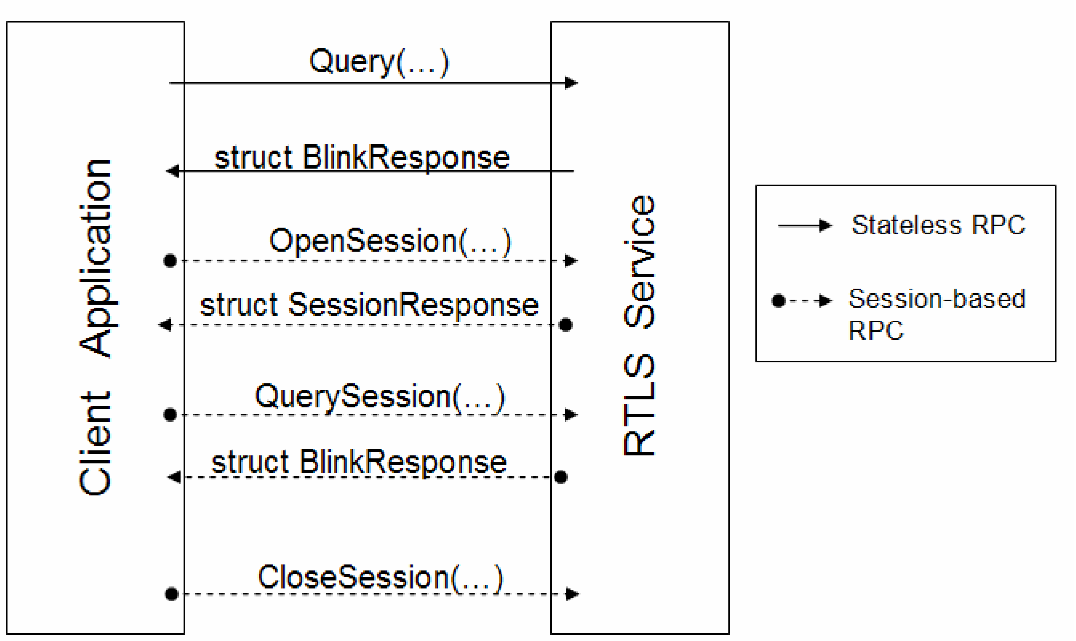

Figure 1 — Architecture of SOAP-RPC

6.4 Nomenclature and conventions

The API standard declares namespaces for RTLS in its XML Schema. All fields in the message exchange fall under one of the namespaces described in those schemas. See the Annex A for the schemas.

7 Subroutine calls

7.1 Overview of SOAP-RPC

a) struct QueryResponse Query(char* queryName, struct FilterType filters, struct FieldType fields, struct SortType SortBy)

b) struct SessionResponse OpenSession(char* queryName, struct FilterType filters, struct FieldType fields)

c) struct QueryResponse QuerySession (char* SessionID)

d) void CloseSession(char* SessionID)

“SOAP-RPC” refers to the remote methods provided to client applications by the RTLS Service.

The function signatures shown above are in C language syntax, merely to describe corresponding SOAP-RPC methods concisely. The SOAP-RPC may be bound to any programming language, with corresponding subroutines. The client implementation of each subroutine shall create a well-formatted *SOAP* message, and should make an *HTTP-POST* request to the RTLS Service.

Responses to a request may be a *SOAP* fault structure, a QueryResponse structure, or a SessionResponse structure. These structures may be deserialized, only if no error occurred. If an error occurs, the server shall return a *SOAP* fault structure, and the client may deserialize that fault structure.

7.2 Remote Procedure Call: Query

7.2.1 Purpose

This client request retrieves the last blink received by the RTLS Service for all tags that match the criteria of the query. At most 1 blink per tag is returned. This is a stateless query. No state about the query is kept in the RTLS service.

7.2.2 Synopsis

The RPC function signature in C language syntax would look like:

struct QueryResponse Query(char* queryName, struct FilterType filters, struct FieldType fields, struct SortType SortBy)

Input parameters:

- QueryName
- Filters
- Fields
- SortBy

Output parameters:

— QueryResponse structure

An underlying requirement is that parameters follow the Query method *SOAP* schema, as described in the Annex A, Clause A.2.

7.2.3 Description

The client application shall construct a *SOAP* request message, which will be deserialized at the service to invoke the function above. The *SOAP* request schema for Query RPC is described in the Annex A.

The client application is responsible for opening a connection (typically, using HTTP 1.1), and sending the *SOAP* request message - Query, with its associated parameters. The response will consist of either a QueryResponse structure or a *SOAP* fault structure. The client application, or a library that implements client side proxy functions, shall parse the response accordingly.

All the parameters discussed below refer to fields of a tag blink. These fields are described in the QueryResponse data structure, in Clause 8.2, and in the XML Schema defined in Annex A.

7.2.4 Parameter: queryName

This is the simplest parameter. This parameter opens up the opportunity for creating other queries to the RTLS system, in the future.

Requirements:

1) The RTLS Service shall accept "RTLS_Blinks" as the value for the query name.

EXAMPLE

<QueryName>RTLS_Blinks</QueryName>

7.2.5 Parameter: filters structure

This is the most complex parameter.

Requirements:

1) The service shall return only those tag blinks that match the filters criteria, in the QueryResponse structure.

2) If no filters are needed, the filter parameter shall be passed as follows: <FilterBy/>.

3) The filters shall be expressed as a sum-of-products Boolean expression, with Boolean operators acting on relational expressions, which in turn contain TagBlink fields and their values.

4) The filters parameter shall use a subset of Boolean and relational operators, as standardized in: ISO/IEC 9075:2003 (all parts), *Information technology — Database languages — SQL*. See Table 1 below.

5) Filters shall not include fields with no values, such as parent fields. Examples of parent fields are <VendorSection>, <States/> and <TagBlink/>.

6) The service shall ignore extraneous blank spaces in non-string field values.

7) The right-hand side of all relational expressions, except those containing '=' only, shall be inside a CDATA section. E.g. <TagID><![CDATA[<> 600]]></TagID>.