S60 Platform: Landmarks Exchange Format Specification

Version 1.0
September 1, 2005

Legal notice

Contents

Change history

September 1, 2005	Version 1.0	Initial document release. Revision on August 29, 2006: minor editorial changes

1. Introduction

1.1 Scope of this document

This document specifies how mobile applications can exchange landmark information over cellular and local connectivity. The exchange of information may happen between two mobile terminal applications and/or terminal and network applications. This document uses reference document [19] as a baseline for the definition of a landmark object using XML schema representation. Nevertheless, the use of landmarks as outlined in this document is not restricted to Java-enabled devices. XML provides a platform-independent way of representing a landmark. However, compatibility with reference document [19] should be enforced.

1.2 Landmarks

Landmarks are the key concept in the managing and consuming of personal location information with location-aware services and applications.

A landmark is an object that contains coordinates of a certain location in the WGS84 datum. A landmark database contains a collection of landmarks the user has created, received, downloaded, etc. Landmarks allow saving location information in a manageable way (not in coordinates, but with user-defined names of locations).

The mobile phone user can use landmarks e.g. to navigate to a certain location. The user fetches a landmark from the landmark database and uses a navigation application to find the way to the destination. Another use case for landmarks is that they can be used with a “reminder” application. In this case, the user is provided with a reminder when the user gets close to the location defined by the landmark (e.g. when the user gets close to a Mall, the user will get the shopping list as remainder).

The user must be able to create a landmark for the user’s current location (**Create a landmark**) and also to be able to create a landmark by manually inserting the coordinates for the landmark. The user must also be able to edit and delete landmarks. The user should be able to categorize the landmarks, e.g. Hotel, Pub, Golf course name, etc. The user should also be able to send and receive landmarks over various communication mechanisms such as SMS, MMS, e-mail, HTTP, OBEX, etc.

1.3 Key requirements in landmarks sharing

1.3.1 Messaging of landmarks

The user should be able to send, receive and forward landmarks. This would increase the usage of landmarks and many location applications significantly.

An example use case:

User A is organizing an important meeting with customers. The meeting will take place in London but many participants come to the meeting from abroad. User A sends a landmark (location) of the meeting facility to the meeting participants who can use the landmark with their navigation application to find their way to the meeting.

1.3.2 Downloading of landmarks

The phone users should be able to download landmarks using their browser.

An example use case:

User A is traveling to Barcelona. To ensure that the user will find the way to the hotel, the user browses the Web pages of the hotel and downloads the landmark for the hotel. After arriving at Barcelona, User A starts the navigation application and uses the downloaded landmark to find the way to the hotel.

1.3.3 Interoperability

Landmark messaging should work across different manufacturers' devices. This specification is open and license-free and allows incorporating the functionality to different software platforms and interoperable Java™ MIDP landmark messaging applications for the devices of other manufacturers.

1.4 Key use cases for landmarks sharing

Four use cases are supported:

1. The user sends landmark(s) data to another user via a messaging bearer
2. The user receives a landmark(s) from another user via a messaging bearer
3. The user forwards a landmark(s) to another user.
4. The user downloads a landmark(s) from a Web site.

The basic use cases of landmarks (adding, editing, removing, etc.) are out of the scope of this specification.

1.5 Technical use cases

1.5.1 Selecting a landmark

Participants:

Mobile phone user

Brief description:

This use case describes the flow of events when the mobile phone user specifies landmarks by selecting one or several items from landmarks storage.

Basic flow:

1. The user opens a landmark application.
2. The user selects a landmark item.

1.5.2 Sending a landmark

Participants:

Mobile phone user

Brief description:

This use case describes the flow of events when the mobile phone user sends a landmark message to one or several remote parties.

Basic flow:

1. The user selects a landmark to be sent.
2. The user selects the messaging media (Send via > SMS, MMS, email, Bluetooth, Infrared).
3. The user specifies a recipient/recipients, enters message text and adds other attachments (if applicable) and may delete or change the sending options (up to the message editor).
4. The user selects **Send**.

1.5.3 Receiving a landmark

Participants:

Mobile phone user

Brief description:

This use case describes the flow of events when the mobile phone user receives a landmark.

Basic flow:

1. The mobile phone receives a message.
2. The incoming message appears to the *Inbox*.
3. The user is notified about the new message with the **Message received** notification.

1.5.4 Opening and viewing a landmark message

Participants:

Mobile phone user

Brief description:

This use case describes the flow of events when the mobile phone user opens a message with a landmark object.

Basic flow:

1. The user selects and opens a message (a new (Inbox) or already read (All message folders) message).
2. The landmark information with details is shown to the user. In case of an MMS or email message, the landmark document is shown as an attachment, which must be opened separately.

3. The user is able to save the landmark information.

Direct landmark forwarding can be supported or the user has to store the landmark first and then send the message to another recipient.

1.5.5 Downloading a landmark

Participants:

Mobile phone user

Brief description:

This use case describes the flow of events when the mobile phone user browses to a Web site and clicks a landmark for downloading.

Basic flow:

1. The user opens a mobile browser and browses to a Web/xHTML page.
2. The user clicks a hyperlink that is associated with a landmark document.
3. The terminal receives the landmarks document (over HTTP) and opens a viewer for viewing the landmark document.
4. The user is able to save the landmark information.

2. Representation of landmarks

A landmark object is used to deliver waypoint or Point-of-Interest information. It may contain the following information:

- Name

  The name of landmark.

- Description

  A textual description of the landmark.

- Category

  The name of the category, e.g. Restaurants, Golf courses, etc. A landmark may belong to multiple categories. Categories may be global or user-defined.

- Coordinates data

  The coordinates of the landmark (e.g. latitude and longitude) with coordinates accuracy information.

- Coverage radius

  The radius of the area covered by the landmark.

- Address information

  The textual address information of the landmark.

- Media links

  A set of URIs to a media data associated with the place.

Each landmark may belong to a certain category and multiple landmarks may be grouped in a landmark collection, i.e. a landmark collection contains the following information:

- Landmarks: A plurality of landmarks as defined above.

The content of the data types is based on the landmark related objects defined in the landmark class of Location API for J2ME, see reference document [19].

The landmark related objects are represented in XML schema (see reference documents [20] and [21]), which is an XML language (see reference document [9]). XML schema provides additional power compared to plain XML or DTD style representation of structured data, e.g. it provides strong data typing.

2.1 Landmarks XML schema

A landmark object represents a landmark, i.e. a known location with a name, and the landmark object is the fundamental object in the context of landmark sharing. The definition of the landmark follows that of reference document [19].

Note: The XML standard requires all conformant XML processors to support UTF-8 and UTF-16 character encoding. UTF-8 and UTF-16 are both conforming encoding forms of the Unicode standard or ISO/IEC 10646 (see reference document [16]).

2.2 Landmark exchange XML schema

An implementation can serialize several landmarks together. This facilitates bulk transfer of landmarks from one device to another without the need for attaching the landmarks as individual entities in the multipart structure of a message. The Landmark Collection object is defined by the schema for these purposes.

In JSR-179 (see reference document [19]), the uniqueness of a category is guaranteed within the same `LandmarkStore` but the uniqueness of landmark names is not guaranteed. This implies that when a landmark collection is received, a landmark will always be treated as a 'new' landmark. If the category to which the landmark belongs to in the landmark collection already exists, the landmark may be saved as belonging to that category. On the other hand, if the category does not exist yet, then it is created before the landmark is stored.

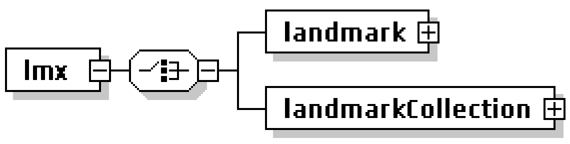
2.2.1 Schema

Schema name	`lmx.xsd`
Namespace name	http://www.nokia.com/schemas/location/landmarks/1/0
Global elements	`lmx`

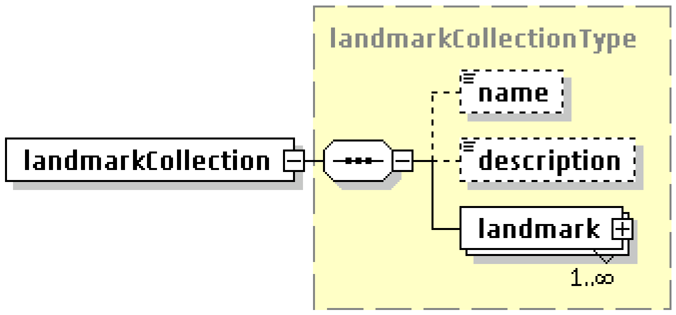
Complex types	`addressInfoType`
	`categoryType`
	`coordinatesType`
	`landmarkCollectionType`
	`landmarkType`
	`mediaLinkType`

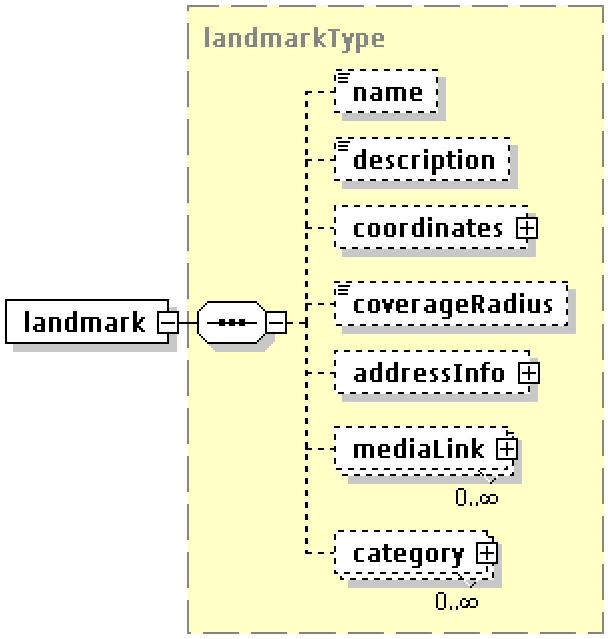
2.2.2 Root element

A Landmarks XML document must start from an `lmx` element.

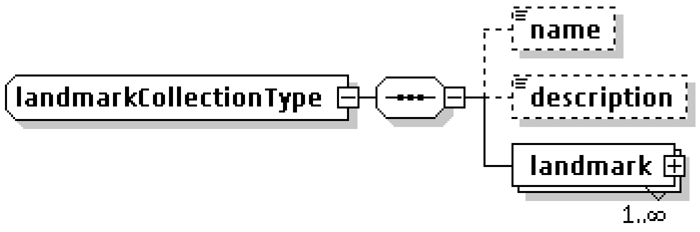
Element	`lmx`
Diagram	
Annotation	The root element of a landmarks XML exchange file. May contain either a single landmark or a collection of landmarks.

Child elements:

Element	`lmx/landmarkCollection`
Diagram	 landmarkCollectionType landmarkCollection name description landmark 1..∞
Type	`landmarkCollectionType`
Annotation	A collection of landmarks.

Element	`lmx/landmark`
Diagram	 landmarkType landmark name description coordinates coverageRadius addressInfo mediaLink 0..∞ category 0..∞
Type	`landmarkType`
Annotation	A single landmark.

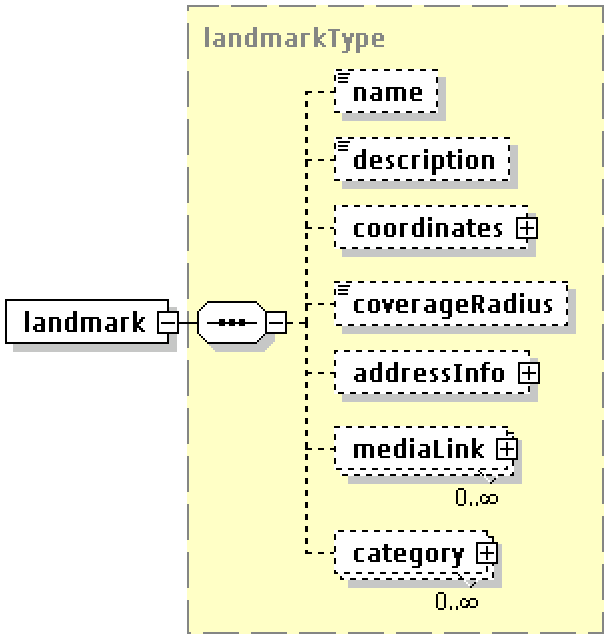
2.2.3 Landmark collection

complexType	`landmarkCollectionType`
Diagram	 landmarkCollectionType name description landmark 1..∞
Used by element	`lmx/landmarkCollection`
Annotation	The landmark collection type. The landmark collection includes a set of landmarks. At least one landmark is required. The landmark collection may have a name and description, which both are mostly intended for displaying purposes.

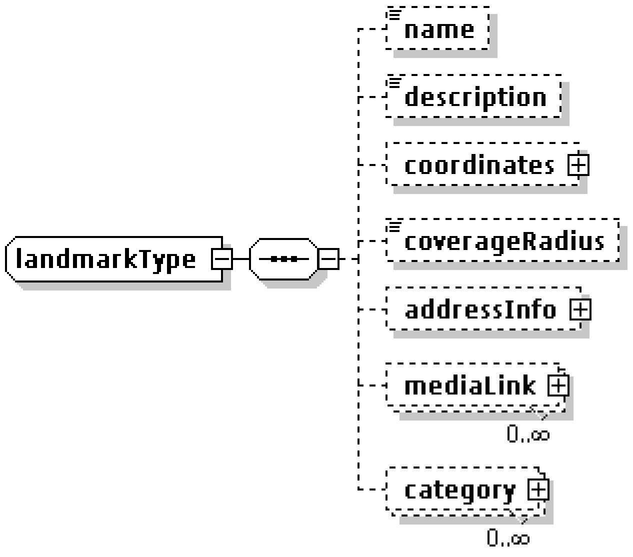
Child elements:

Element	`landmarkCollectionType/name`
Diagram	
Type	`xsd:string`
Annotation	The name of the collection. Optional data, used for displaying purposes.

Element	`landmarkCollectionType/description`
Diagram	
Type	`xsd:string`
Annotation	A description of the collection. Optional data, used for displaying purposes. Usually this is more detailed than the name or provides some additional information about the collection.

Element	`landmarkCollectionType/landmark`
Diagram	 landmarkType landmark name description coordinates coverageRadius addressInfo mediaLink 0..∞ category 0..∞
Type	`landmarkType`
Annotation	A landmark.

2.2.4 Landmark

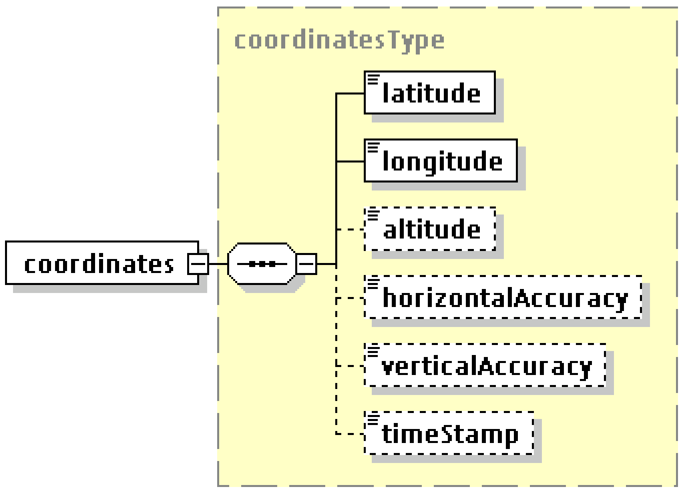
complexType	`landmarkType`
Diagram	 landmarkType name description coordinates coverageRadius addressInfo mediaLink 0..∞ category 0..∞
Used by elements	`lmx/landmark` `landmarkCollectionType/landmark`

Annotation	The landmark type. The landmark is a point on a map or a place in the world the position of which is defined by coordinates. Landmarks may belong to a set of landmark categories. A landmark may also have additional address information of the place the landmark points to. The landmark may also have additional information such as a description, coverage radius and a list of media links.

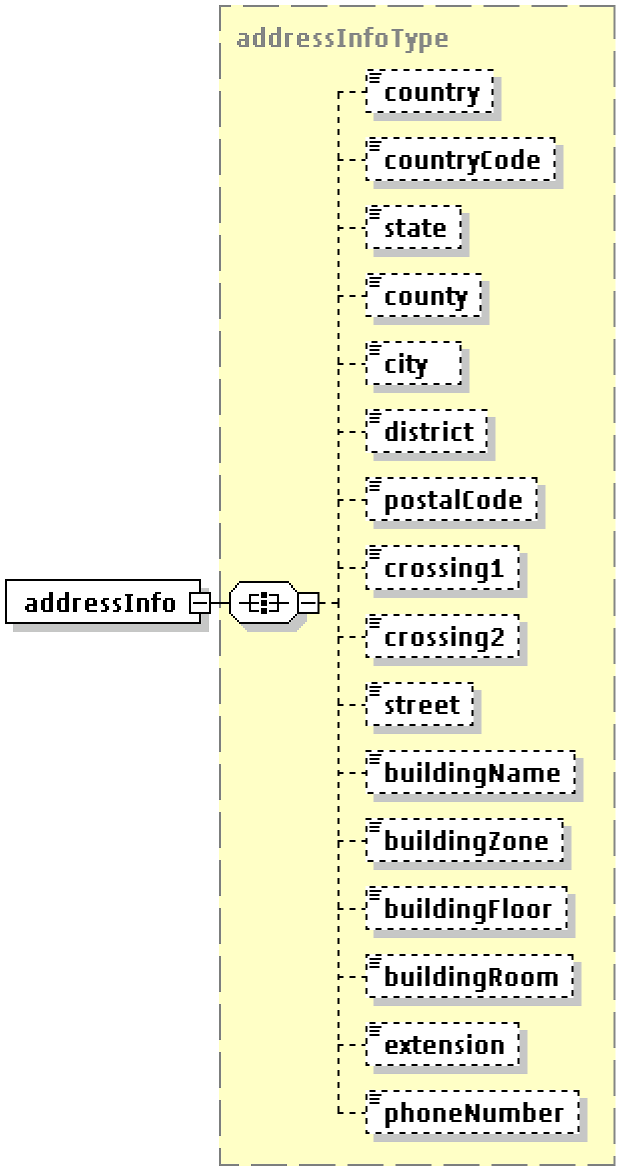
Child elements:

Element	`landmarkType/name`
Diagram	
Type	`xsd:string`
Annotation	The name of the landmark. Optional data, used for displaying purposes.

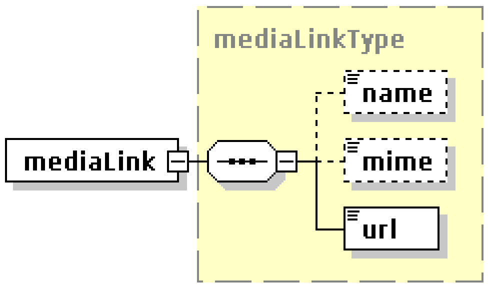
Element	`landmarkType/description`
Diagram	
Type	`xsd:string`
Annotation	A description of the landmark. Optional data, used for displaying purposes. Usually the description is more detailed than the name or provides some additional information about the landmark.

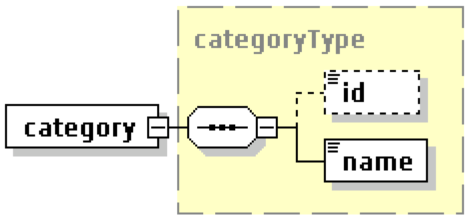
Element	`landmarkType/coordinates`
Diagram	 coordinatesType latitude longitude altitude horizontalAccuracy verticalAccuracy timeStamp coordinates
Type	`coordinatesType`
Annotation	The landmark's coordinates.

Element	`landmarkType/coverageRadius`
Diagram	
Type	Restriction of `xsd:float`
Facets	`minInclusive = 0`
Annotation	The coverage radius of the landmark in meters. The landmark's coordinates are the center of the area this landmark covers.

Element	`landmarkType/addressInfo`
Diagram	 addressInfoType addressInfo country countryCode state county city district postalCode crossing1 crossing2 street buildingName buildingZone buildingFloor buildingRoom extension phoneNumber

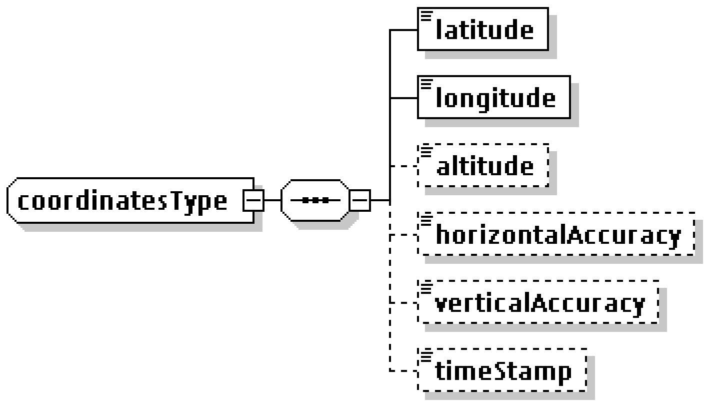
Type	`addressInfoType`
Annotation	The landmark's address information.

Element	`landmarkType/mediaLink`
Diagram	
Type	`mediaLinkType`
Annotation	A media link associated with this place.

Element	`landmarkType/category`
Diagram	
Type	`categoryType`
Annotation	The landmark category.

2.2.5 Coordinates

Coordinates are represented as latitude-longitude-altitude values that are associated with an accuracy value. The latitude and longitude values are expressed in degrees using floating-point values. The degrees are in decimal values (rather than minutes or seconds). The coordinates are given using the WGS84 datum.

complexType	`coordinatesType`
Diagram	 coordinatesType latitude longitude altitude horizontalAccuracy verticalAccuracy timeStamp
Used by element	`landmarkType/coordinates`
Annotation	The landmark coordinates' type. The landmark coordinates must be specified in WGS84 format. Latitude and longitude values are required. Restrictions are applied to the coordinate values.

Child elements:

Element	`coordinatesType/latitude`
Diagram	
Type	Restriction of `xsd:double`

Facets	`minInclusive = -90`
	`maxInclusive = 90`

Annotation	The latitude part of a WGS84 coordinate in degrees. Must be in the range [-90;+90].

Element	`coordinatesType/longitude`
Diagram	
Type	Restriction of xsd:double

Facets	`minInclusive = -180`
	`maxExclusive = +180`

Annotation	The longitude part of a WGS84 coordinate in degrees. Must be in the range [-180;+180).

Element	`coordinatesType/altitude`
Diagram	
Type	`xsd:float`
Annotation	The altitude part of a WGS84 coordinate in meters.

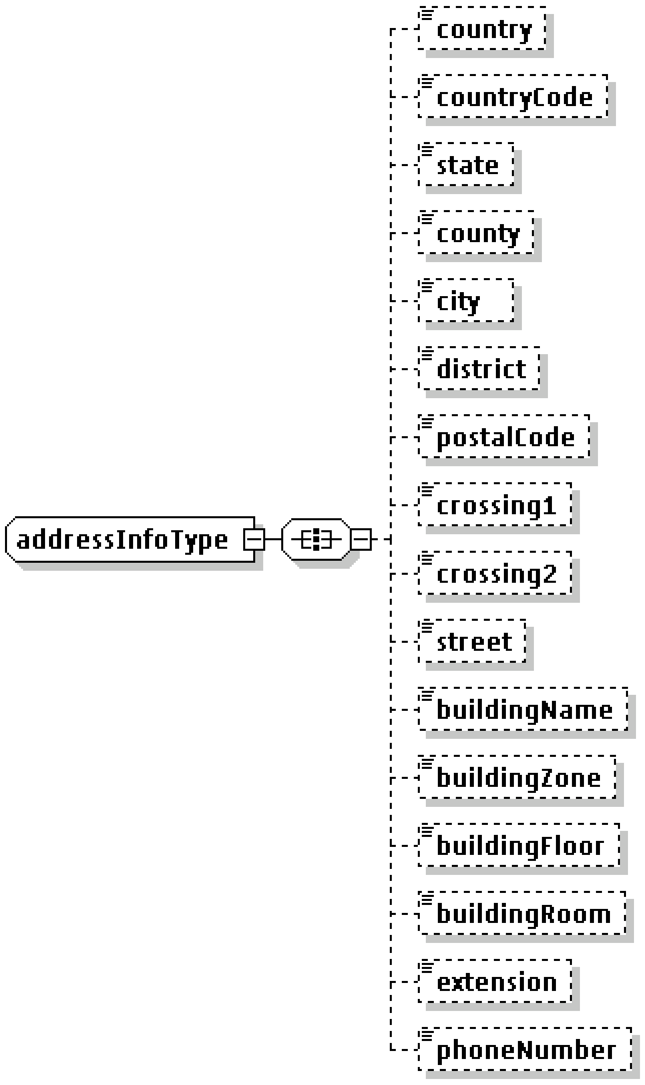
Element	`coordinatesType/horizontalAccuracy`
Diagram	
Type	Restriction of `xsd:float`
Facets	`minInclusive = 0`
Annotation	Horizontal accuracy of the coordinates in meters.

Element	`coordinatesType/verticalAccuracy`
Diagram	
Type	Restriction of `xsd:float`
Facets	`minInclusive = 0`
Annotation	Vertical accuracy of the coordinate in meters.

Element	`coordinatesType/timeStamp`
Diagram	
Type	`xsd:dateTime`
Annotation	The timestamp of the moment when the coordinates were calculated. May contain date, time and time zone information.

2.2.6 Address information

`addressInfoType` holds textual information about a location, e.g. a street address.

complexType	`addressInfoType`
Diagram	 addressInfoType country countryCode state county city district postalCode crossing1 crossing2 street buildingName buildingZone buildingFloor buildingRoom extension phoneNumber
Used by element	`landmarkType/addressInfo`
Annotation	The landmark address information type. An address is divided into fields, each of which represents a part of the address information. The order of the fields is not specified.

Child elements:

Element	`addressInfoType/country`
Diagram	
Type	`xsd:string`
Annotation	The address field denoting country.

Element	`addressInfoType/countryCode`
Diagram	
Type	Restriction of `xsd:token`
Facets	`length = 2`
Annotation	The address field denoting country as a two-letter ISO-3166-1 code.

Element	`addressInfoType/state`
Diagram	
Type	`xsd:string`
Annotation	The address field denoting state or province.

Element	`addressInfoType/county`
Diagram	
Type	`xsd:string`
Annotation	The address field denoting county, an entity between a state and a city.

Element	`addressInfoType/city`
Diagram	
Type	`xsd:string`
Annotation	The address field denoting city or town name.

Element	`addressInfoType/district`
Diagram	
Type	`xsd:string`
Annotation	The address field denoting municipal district.

Element	`addressInfoType/postalCode`
Diagram	
Type	`xsd:string`
Annotation	The address field denoting zip or postal code.

Element	`addressInfoType/crossing1`
Diagram	
Type	`xsd:string`
Annotation	The address field denoting a street in a crossing.

Element	`addressInfoType/crossing2`
Diagram	
Type	`xsd:string`
Annotation	The address field denoting another street in a crossing.

Element	`addressInfoType/street`
Diagram	
Type	`xsd:string`
Annotation	Address field denoting street name and number.

Element	`addressInfoType/buildingName`
Diagram	

Type	`xsd:string`
Annotation	The address field denoting a building name.

Element	`addressInfoType/buildingZone`
Diagram	
Type	`xsd:string`
Annotation	Address field denoting a building zone.

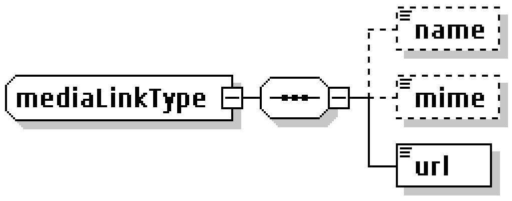
Element	`addressInfoType/buildingFloor`
Diagram	
Type	`xsd:string`
Annotation	The address field denoting a building floor.

Element	`addressInfoType/buildingRoom`
Diagram	
Type	`xsd:string`
Annotation	The address field denoting a building room.

Element	`addressInfoType/extension`
Diagram	
Type	`xsd:string`
Annotation	The address field denoting address extension, e.g. flat number.

Element	`addressInfoType/phoneNumber`
Diagram	
Type	`xsd:string`
Annotation	The address field denoting a phone number for this place.

2.2.7 Media link

complexType	`mediaLinkType`
Diagram	
Used by element	`landmarkType/mediaLink`
Annotation	The media link information type. A media link is a URI and optional displayable name.

Child elements:

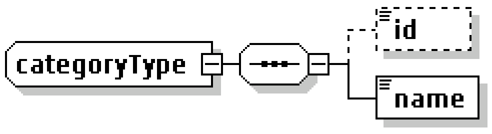
Element	`mediaLinkType/name`
Diagram	
Type	`xsd:string`
Annotation	The displayable name of the media link.

Element	`mediaLinkType/mime`
Diagram	
Type	`xsd:string`
Annotation	The MIME type of the referenced data.

Element	`mediaLinkType/url`
Diagram	
Type	`xsd:anyURI`
Annotation	The URI of the media link. This URI must be absolute.

2.2.8 Landmark category

Category indicates the category a landmark belongs to. The category consists of an ID and a logical name. The ID is only used with global landmark categories. Predefined categories are listed in Appendix C, “Predefined global landmark categories.” Global landmark category IDs are used for localization purposes: if an ID is known on the receiving party, then category may be shown using current language and the name specified can be ignored.

complexType	`categoryType`
Diagram	
Used by element	`landmarkType/category`
Annotation	The landmark category type. A category is a name and an optional global landmark category ID.

Child elements:

Element	`categoryType/id`
Diagram	
Type	`xsd:unsignedShort`
Annotation	The ID of a global landmark category.

Element	`categoryType/name`
Diagram	
Type	restriction of `xsd:string`
Annotation	The name of the landmark category. Must be non-empty.

2.2.9 XML data types

This schema utilizes the data types defined in reference document [21]. Data in an instance XML document must conform to these definitions. Data validity can be checked either using XML validators, when those are available, or directly by parsers of landmark data. The validation rules defined in reference document [21] should also be followed by encoders to guarantee that the data they generate is compatible with this format and is read without errors by a compatible parser at the receiving side.

The schema defined in this document utilizes a small set of XML data types. The following is a short description of these XML data types for parser and encoder implementations. For more details, see reference document [21].

Type name	`xsd:string`
Definition	The string data type represents character strings in XML. The value space of string is a set of finite-length sequences of characters (as defined in reference document [30]) that match the Char production from reference document [30].

Example	`<lm:landmark>`
	`  <lm:name>A landmark</lm:name>`
	`  <lm:description>Example string datatype`
	`usage</lm:description>`
	`</lm:landmark>`

Type name	`xsd:float`
Definition	`float` corresponds to the IEEE single-precision 32-bit floating-point type (see reference document [31]). The basic value space of `float` consists of the values `m × 2^e`, where `m` is an integer whose absolute value is less than `2^24` and `e` is an integer between `-149` and `104`, inclusive.
Representation	`float` values have a lexical representation consisting of a mantissa followed (optionally) by the character "E" or "e", followed by an exponent. The exponent must be an integer. The mantissa must be a decimal number. The representations for exponent and mantissa must follow the lexical rules for integer and decimal. If the "E" or "e" and the following exponent are omitted, an exponent value of 0 is assumed. This allows to have up to 7 decimal digits (8 digits for some values). The special values positive and negative zero, positive and negative infinity and not-a-number have the lexical representations `0`, `-0`, `INF`, `-INF` and `NaN` respectively. For example, `-1E4, 1267.43233E12, 12.78e-2, 12` and `INF` are all legal literals for float. **Note:** Empty strings are not valid `float` numbers. **Note:** The Coordinates tag requires both latitude and longitude. If one of these parameters is not defined, they still cannot be omitted. Instead, use a `NaN` value as in the following example.

Example	`<lm:coordinates>`
	`  <lm:latitude>40.25</lm:latitude>`
	`  <lm:longitude>NaN</lm:longitude>`
	`</lm:coordinates>`

Type name	`xsd:double`
Definition	The `double` data type corresponds to the IEEE double-precision 64-bit floating-point type (see reference document [31]). The basic value space of double consists of the values `m × 2^e`, where `m` is an integer whose absolute value is less than `2^53`,

	and `e` is an integer between `-1075` and `970`, inclusive. This allows to have up to 15 decimal digits (16 digits for some values).
Representation	`double` has the same representation rules as `float`.

Example	`<lm:landmark>`
	`  <lm:coverageRadius>INF</lm:coverageRadius>`
	`</lm:landmark>`

Type name	`xsd:unsignedShort`
Definition	`unsignedShort` is derived from decimal by setting the following restrictions: • `fractionDigits = 0` • `minInclusive = 0` • `maxInclusive = 65535`
Representation	`unsignedShort` has a lexical representation consisting of a finite-length sequence of decimal digits (#x30-#x39). For example: 0, 12678, 10000

Example	`<lm:category>`
	`  <lm:id>9000</lm:id>`
	`  <lm:name>Restaurants</lm:name>`
	`</lm:category>`

Type name	`xsd:anyURI`
Definition	`anyURI` represents a Uniform Resource Identifier Reference (URI). An `anyURI` value can be absolute or relative and may have an optional fragment identifier (i.e., it may be a URI Reference). This type is used to specify the intention that the value fulfills the role of a URI as defined by RFC 2396 (see reference document [32]) as amended by RFC 2732 (see reference document [33]). **Note:** It is required that the `mediaLinkType/url` element contains an absolute URI. This is because relative URIs have no meaning without a base URI and there is no place in the present schema to specify it.
Representation	The lexical space of `anyURI` is a finite-length character sequences which, when the algorithm defined in reference document [34] is applied to them, result in strings that are legal URIs according to RFC 2396 (see reference document [32]) as amended by RFC 2732 (see reference document [33]). **Note:** In principle, spaces are allowed in the lexical space of `anyURI`. However, their use is highly discouraged (unless they are encoded by %20).

Example	`<lm:mediaLink>`
	`  <lm:url>http://www.w3.org/TR/xmlschema-`
	`2/</lm:url>`
	`</lm:mediaLink>`

Type name	`xsd:dateTime`
Definition	`dateTime` represents a specific instant of time. The value space of `dateTime` is the space of combinations of date and time of day values as defined in reference document [15].
Representation	A single lexical representation, which is a subset of the lexical representations allowed by ISO 8601 (see reference document [15]), is allowed for `dateTime`. This lexical representation is the ISO 8601 (see reference document [15]) extended format `CCYY-MM-DDThh:mm:ss` where `CC` represents the century, `YY` the year, `MM` the month and `DD` the day, preceded by an optional leading "-" sign to indicate a negative number. If the sign is omitted, "+" is assumed. The letter "T" is the date/time separator and `hh`, `mm`, `ss` represent the hour, minute and second respectively. Additional digits can be used to increase the precision of fractional seconds if desired i.e. the format `ss.ss`... with any number of digits after the decimal point is supported. The fractional seconds part is optional; other parts of the lexical form are not optional. To accommodate year values greater than 9999, additional digits can be added to the left of this representation. Leading zeros are required if the year value would otherwise have fewer than four digits; otherwise they are forbidden. The year 0000 is prohibited. The `CCYY` field must have at least four digits, the `MM`, `DD`, `SS`, `hh`, `mm` and `ss` fields exactly two digits each (not counting fractional seconds); leading zeroes must be used if the field would otherwise have too few digits. This representation may be immediately followed by a "Z" to indicate Coordinated Universal Time (UTC) or, to indicate the time zone, i.e. the difference between the local time and Coordinated Universal Time, immediately followed by a sign, + or -, followed by the difference from UTC represented as `hh:mm` (**Note:** The minutes part is required). See reference document [21] for details about the legal values in the various fields. If the time zone is included, both hours and minutes must be present. For example, to indicate 1:20 pm on May the 31st, 1999 for Eastern Standard Time which is five hours behind Coordinated Universal Time (UTC), one would write: `1999-05-31T13:20:00-05:00`.

Example	`<lm:coordinates>`
	`  <lm:timeStamp>2004-06-`
	`24T08:58:05Z</lm:timeStamp>`
	`</lm:coordinates>`

3. Backward and forward compatibility

There is a potential risk that the schema needs to be changed in the future. These changes could be due to bugs found or general development in the landmark concept, the evolution of JSR 179, etc.

In general, the following requirements are given for forward and backward compatibility:

- A terminal supporting subsequent releases of this specification must support the schemas of the previous releases of this specification to ensure backwards compatibility. This means that an implementation supporting a subsequent release of this specification must be able to parse and handle messages containing instance documents serialized according to previous versions of the schemas. The parser may support validation of the schema according to the `schemaLocation` attribute given in the instance document. Deprecated elements and attributes with respect to the highest supported schemas must be ignored by the implementation as the implementation does not necessarily support these elements or attributes in its internal storage structure anymore.

- An implementation conformant to this specification must be able to parse instance documents serialized according to a subsequent schema with respect to this release of the specification to ensure forward compatibility. The parser may support validation of the schema according to the `schemaLocation` attribute given in the instance document. Elements and attributes not understood by the implementation are ignored.

- The version of the schemas used for any instance document is given by the `schemaLocation` attribute either as a part of the namespace URI or the schema itself.

- The semantics of any element or attribute must not be changed over subsequent versions of this specification. Where the semantics of a particular element change, the original element or attribute should be deprecated and a new element or attribute introduced with the intended semantics and with a different naming. This is done to avoid implementations over subsequent releases of this specification treating the same element differently.

4. Compact binary representation of landmark collection XML documents

The Wireless Binary XML (WBXML) content format (see reference document [4]) was designed to reduce the transmission size of XML documents, allowing more effective use of XML data on narrow-band communication channels. The binary format allows for compact transmission with no loss of functionality or semantic information. The format is designed to preserve the element structure of XML, allowing a browser to skip unknown elements or attributes. The binary format encodes the parsed physical form of an XML document; i.e., the structure and content of the document entities. Meta-information, including the document type definition and conditional sections, is removed when the document is converted to the binary format.

Landmark XML documents can be encoded using a compact binary representation before transmission. Any implementation conforming to this specification must support both the compact binary format as well as the verbose text-based XML format. The compact binary representation is based on the WAP Binary XML content format (see reference document [4]).

4.1 Extension tokens

4.1.1 Tag tokens

This specification defines a set of single-byte tokens corresponding to the tags defined in the XML schema. These tokens are defined within the code page zero.

Tag tokens are single `u_int8s`. Their structure is described in the following section.

4.1.1.1 Tag format

See reference document [4] for details.

Bit(s)	Description
7 (most significant)	Indicates whether attributes follow the tag code. If this bit is zero, the tag contains no attributes. If this bit is one, the tag is followed immediately by one or more attributes. The attribute list is terminated by an END token.
6	Indicates whether this tag begins an element containing content. If this bit is zero, the tag contains no content and no end tag. If this bit is one, the tag is followed by any content it contains and is terminated by an END token.
5-0	Indicates the tag identity.

4.1.2 Attribute tokens

This specification defines a set of single-byte tokens corresponding to the attribute names and values defined in the XML schema. These tokens are defined within the code page zero.

4.1.2.1 *Attribute code space (ATTRSTART and ATTRVALUE)*

Attribute tokens are encoded as single u_int8s. The attribute code space is split into the following two ranges (in addition to the global range present in all code spaces):

- Attribute start

  Tokens with a value less than `128` indicate the start of an attribute. The attribute start token fully identifies the attribute name, e.g., `URL=`, and may optionally specify the beginning of the attribute value, e.g., `PUBLIC="TRUE"`. Unknown attribute names are encoded with the globally unique code `LITERAL` (see reference document [4]). `LITERAL` must not be used to encode any portion of an attribute value.

- Attribute value

  Tokens with a value of `128` or greater represent a well-known string present in an attribute value. These tokens may only be used to represent attribute values. Unknown attribute values are encoded with string, entity or extension codes.

4.2 Encoding semantics

4.2.1 Document validation

XML document validation (see reference document [10]) should occur during the process of tokenizing a landmark collection document and, if done, it should be based on the `schemaLocation` declared in the document. However, reference document [10] does not require the processor to obtain or use the cited schema documents and the processor is free to use other schemas obtained by any suitable means, or to use no schema at all. In other words, the schema can be locally stored or the processor may not use validation at all. The tokenization process must check that the source document is well formed XML and that it is valid, and it should notify the originator of any errors detected in the source document.

Note: The tokenization process removes the element namespace qualification since tag tokens are fixed whereas the namespace qualification is dynamic and may vary from one instance document to another. However, all landmark schema elements belong to the same namespace, so actually there is no need for namespace qualification. Nevertheless, when the WBXML encoded XML document instance is decoded, the tokenizer must reconstruct the namespace qualification. The reason is that a validating parser conforming to reference documents [20] and [21] would otherwise conclude that the instance document is ill-formed and not valid as the parser is not instructed that the elements belong to the same namespace.

Reconstructing the namespace qualification is a straightforward process:

1. All elements are prefixed with a namespace prefix, e.g. lm:.

2. The target namespace, i.e. `xmlns="http://www.nokia.com/schemas/location/landmarks/1/0/"` in the instance document is postfixed with the same namespace qualifier, e.g. `:lm`, giving `xmlns:lm=http://www.nokia.com/schemas/location/landmarks/1/0/`

Even though the namespace qualifier is dynamic and determined by the context, the use of the lm: prefix is recommended.

4.3 Numeric constants

4.3.1 Document identifier

Originally, WBXML was specified for DTD-based XML and hence there are no provisions regarding the use of XML schema in reference document [4]. In WBXML, there are a number of global tokens specified which are unique over all code spaces. The document identifier or the public identifier was specified as being a global token. The public identifier for DTDs corresponds to the `schemaLocation` element in XML schema. Since there are no provisions for XML schema in reference document [4], the `schemaLocation` token is specified in the attribute token space instead of the global token space.

The public document identifier for this landmarks exchange format is registered in OMNA as follows:

Public Identifier	Value	WBXML representation
`-//NOKIA//DTD LANDMARKS 1.0//EN`	`0x1204`	`A4 04`

4.3.2 Tag tokens, code page 0

The following token codes represent tags in the code page zero (0). All the numbers are in hexadecimal.

Element name	Base token	With content	Element name	Base token	With content
`lmx`	05	C5*	`state`	17	57
`landmarkCollection`	06	46	`county`	18	58
`landmark`	07	47	`city`	19	59
`name`	08	48	`district`	1A	5A
`description`	09	49	`postalCode`	1B	5B
`coordinates`	0A	4A	`crossing1`	1C	5C
`latitude`	0B	4B	`crossing2`	1D	5D
`longitude`	0C	4C	`street`	1E	5E
`altitude`	0D	4D	`buildingName`	1F	5F
`horizontalAccuracy`	0E	4E	`buildingFloor`	20	60
`verticalAccuracy`	0F	4F	`buildingZone`	21	61
`timeStamp`	10	50	`buildingRoom`	22	62
`coverageRadius`	11	51	`extension`	23	63

Element name	Base token	With content	Element name	Base token	With content
category	12	52	phoneNumber	24	64
id	13	53	mediaLink	25	65
addressInfo	14	54	mime	26	66
country	15	55	url	27	67
countryCode	16	56			

*) Note that `lmx` always has attributes and content and thus bits 7 and 6 will always be set, i.e. token C5 is used. All the other elements do not have any attributes but have content, thus bit 6 is always set. See Appendix B, “Landmarks WBXML tag codes,” for the complete table.

Note that floating-point values are encoded as strings.

4.3.3 Attribute start tokens

The following token codes represent the start of an attribute in the code page zero (0). All the numbers are in hexadecimal.

Token	Attribute name	Attribute value prefix
05	xmlns	http://www.nokia.com/schemas/location/landmarks/
06	xmlns:xsi	
07	xsi:schemaLocation	http://www.nokia.com/schemas/location/landmarks/

4.3.4 Attribute value tokens

The following token codes represent the values of attributes in the code page zero (0). All the numbers are in hexadecimal.

Token	Attribute value	Description
85	1/0/	The schema version.
86	http://www.w3.org/2001/XMLSchema-instance	The W3C schema instance namespace.
87	‘ ’	A whitespace character.
88	lmx.xsd	The schema file name.
89	http://www.nokia.com/schemas/location/landmarks/	The remote schema file path prefix.

Target namespace schema location may be specified in an XML document in two ways:

Schema location	XML	WBXML
Remote	`xsi:schemaLocation=”http://www.nokia.com/schemas/location/landmarks/1/0/ http://www.nokia.com/schemas/location/landmarks/1/0/lmx.xsd”`	07 85 87 89 88
Local	`xsi:schemaLocation=”http://www.nokia.com/schemas/location/landmarks/1/0/ lmx.xsd”`	07 85 87 88

Note: Usage of the attribute value token 0x87 (whitespace) is introduced to reduce the data needed to be transferred and to correspond to the reference document [4] recommendation to use tokens instead of strings whenever possible.

4.4 Example

The example below illustrates how a landmark can be tokenized.

```
<?xml version="1.0" encoding="UTF-8"?>
<lm:lmx xmlns:lm="http://www.nokia.com/schemas/location/landmarks/1/0/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.nokia.com/schemas/location/landmarks/1/0/
lmx.xsd">
  <lm:landmark>
    <lm:name>Nice restaurant</lm:name>
    <lm:coordinates>
      <lm:latitude>65.4321</lm:latitude>
      <lm:longitude>43.2198</lm:longitude>
    </lm:coordinates>
    <lm:addressInfo>
      <lm:city>Helsinki</lm:city>
      <lm:phoneNumber>+35891234567</lm:phoneNumber>
    </lm:addressInfo>
    <lm:category>
      <lm:id>9000</lm:id>
      <lm:name>Restaurants</lm:name>
    </lm:category>
  </lm:landmark>
</lm:lmx>
```

Note: This example specifies that a schema file is located locally (see the `xsi:schemaLocation` attribute value).

The tokenized form of the example above (numbers in hexadecimal) using the WBXML encoding defined in this chapter is found below. This example assumes UTF-8 character encoding and `NULL` terminated strings. In this example, the textual landmark document consists of 600 octets, while the encoded form consists of 116 octets.

```
03 A4 04 6A 00 C5 05 85 06 86 07 85 87 88 01 47 48 03 'N' 'i' 'c' 'e'
' ' 'r' 'e' 's' 't' 'a' 'u' 'r' 'a' 'n' 't' 00 01 4A 4B 03 '6' '5'
'.' '4' '3' '2' '1' 00 01 4C 03 '4' '3' '.' '2' '1' '9' '8' 00 01 01
54 59 03 'H' 'e' 'l' 's' 'i' 'n' 'k' 'i' 00 01 64 03 '+' '3' '5' '8'
'9' '1' '2' '3' '4' '5' '6' '7' 00 01 01 52 53 03 '9' '0' '0' '0' 00
01 48 03 'R' 'e' 's' 't' 'a' 'u' 'r' 'a' 'n' 't' 's' 00 01 01 01 01
```

Token stream description:

Below is the same example in an expanded and annotated form:

Token stream	Description
`03`	Version number, WBXML version 1.3.
`A4 04`	Public Identifier `"-//NOKIA//DTD LANDMARKS 1.0//EN"`
`6A`	`Charset=UTF-8 (MIBEnum 106)`
`00`	String table length.
`C5`	`lmx`, with attributes and content.
`05`	`xmlns`, Attribute value prefix `http://www.nokia.com/schemas/location/landmarks/`.
`85`	Attribute value token `1/0/`
`06`	`xmlns:xsi`, no prefix.
`86`	Attribute value token `http://www.w3.org/2001/XMLSchema-instance`.
`07`	`xsi:schemaLocation`, Attribute value prefix `http://www.nokia.com/schemas/location/landmarks/`
`85`	Attribute value token `1/0/`
`87`	Attribute value token ‘ ’
`88`	Attribute value token ‘`lmx.xsd`’.
`01`	End of attribute list.
`47`	`landmark` with content.
`48`	`name` with content.
`03`	Inline string follows.
`'N' 'i' 'c' 'e' ' ' 'r' 'e' 's' 't' 'a' 'u' 'r' 'a' 'n' 't' 00`	The string ‘`Nice restaurant`’.
`01`	End of `name`.
`4A`	`coordinates` with content.
`4B`	`latitude` with content.
`03`	Inline string follows.
`'6' '5' '.' '4' '3' '2' '1' 00`	The string ’`65.4321`’.

Token stream	Description
`01`	End of `latitude`.
`4C`	`longitude` with content.
`03`	Inline string follows.
`'4' '3' '.' '2' '1' '9' '8' 00`	The string ’`43.2198`’.
`01`	End of `longitude`.
`01`	End of `coordinates`.
`54`	`addressInfo` with content.
`59`	`city` with content.
`03`	Inline string follows.
`'H' 'e' 'l' 's' 'i' 'n' 'k' 'i' 00`	The string ‘`Helsinki`’.
`01`	End of `city`.
`64`	`phoneNumber` with content.
`03`	Inline string follows.
`'+' '3' '5' '8' '9' '1' '2' '3' '4' '5' '6' '7' 00`	The string ‘`+35891234567`’.
`01`	End of `phoneNumber`.
`01`	End of `addressInfo`.
`52`	`category` with content.
`53`	`id` with content.
`03`	Inline string follows.
`'9' '0' '0' '0' 00`	The string ‘`9000`’.
`01`	End of `id`.
`48`	`name` with content.
`03`	Inline string follows.
`'R' 'e' 's' 't' 'a' 'u' 'r' 'a' 'n' 't' 's' 00`	The string ‘`Restaurants`’.
`01`	End of `name`.
`01`	End of `category`.

Token stream	Description
`01`	End of `landmark`.
`01`	End of `lmx`.

Utilizing a string table:

Reference document [4] allows reducing the size of an encoded document having strings values in several places of the document by using string tables. A string table starts immediately after the public identifier with the table's length, see reference document [4]. If the length is other than zero, one or more strings follow.

In the landmarks exchange format, landmark categories are specified for every landmark and if there are several landmarks belonging to the same categories in the document, they can share a category name string. This makes moving the category names into a string table a good idea. It is illustrated in the example below. (Note that in this particular example, creating a string table does not reduce the document size (actually, the document size increases slightly) because there is only one landmark and the category name is not reused.) For the purpose of illustration, the city name "Helsinki" is also moved to the string table.

```
03 A4 04 6A 21 'R' 'e' 's' 't' 'a' 'u' 'r' 'a' 'n' 't' 's' 00 'H' 'e'
'l' 's' 'i' 'n' 'k' 'i' 00 C5 05 85 05 86 07 85 87 88 01 47 48 03 'N'
'i' 'c' 'e' ' ' 'r' 'e' 's' 't' 'a' 'u' 'r' 'a' 'n' 't' 00 01 4A 4B
03 '6' '5' '.' '4' '3' '2' '1' 00 01 4C 03 '4' '3' '.' '2' '1' '9'
'8' 00 01 01 54 59 83 12 01 64 03 '+' '3' '5' '8' '9' '1' '2' '3' '4'
'5' '6' '7' 00 01 01 52 53 03 '9' '0' '0' '0' 00 01 48 83 00 01 01 01
```

Below is the same example in annotated form:

Token stream	Description
`03`	Version number - WBXML version 1.3.
`A4 04`	Public Identifier `"-//NOKIA//DTD LANDMARKS 1.0//EN"`
`6A`	`Charset=UTF-8 (MIBEnum 106)`
`21`	String table length.
`'R' 'e' 's' 't' 'a' 'u' 'r' 'a' 'n' 't' 's' 00`	The string '`Restaurants`'.
`'H' 'e' 'l' 's' 'i' 'n' 'k' 'i' 00`	The string '`Helsinki`'.
`C5`	`lmx`, with attributes and content.
`05`	`xmlns`, Attribute value prefix `http://www.nokia.com/schemas/location/landmarks/`.
`85`	Attribute value token `1/0/`.

Token stream	Description
06	xmlns:xsi, no prefix.
86	Attribute value token http://www.w3.org/2001/XMLSchema-instance.
07	xsi:schemaLocation, Attribute value prefix http://www.nokia.com/schemas/location/landmarks/.
85	Attribute value token 1/0/.
87	Attribute value token ‘ ‘ (whitespace) .
88	Attribute value token ‘lmx.xsd’.
01	End of attribute list.
47	landmark with content.
48	name with content.
03	Inline string follows.
'N' 'i' 'c' 'e' ' ' 'r' 'e' 's' 't' 'a' 'u' 'r' 'a' 'n' 't' 00	The string ‘Nice restaurant’.
01	End of name.
4A	coordinates with content.
4B	latitude with content.
03	Inline string follows.
'6' '5' '.' '4' '3' '2' '1' 00	The string ’65.4321’.
01	End of latitude.
4C	longitude with content.
03	Inline string follows.
'4' '3' '.' '2' '1' '9' '8' 00	The string ’43.2198’.
01	End of longitude.
01	End of coordinates.
54	addressInfo with content.
59	city with content.
83	String reference follows.
12	Offset of the string ‘Helsinki’.
01	End of city.

Token stream	Description
`64`	`phoneNumber` with content.
`03`	Inline string follows.
`'+' '3' '5' '8' '9' '1' '2' '3' '4' '5' '6' '7' 00`	The string '`+35891234567`'.
`01`	End of `phoneNumber`.
`01`	End of `addressInfo`.
`52`	`category` with content.
`53`	`id` with content.
`03`	Inline string follows.
`'9' '0' '0' '0' 00`	The string '`9000`'.
`01`	End of `id`.
`48`	`name` with content.
`83`	Reference to a string follows.
`00`	Offset of the string '`Restaurants`'.
`01`	End of `name`.
`01`	End of `category`.
`01`	End of `landmark`.
`01`	End of `lmx`.

5. Encapsulation on bearers

The mobile phone may send and receive landmark objects through mobile, Internet and local connectivity bearers. Landmarks may be fetched and stored from a local landmark store on the mobile phone using JSR-179 (see reference document [19]).

The following encapsulation is used on different bearers:

- SMS: WDP
- MMS: encapsulation as a multipart entity
- HTTP: encapsulation as a multipart entity
- E-mail: encapsulation as a multipart entity
- Bluetooth and Infrared: OBEX
- Raw data within a file in a file system

The general mechanism for sending and receiving landmarks is to encapsulate the landmark object as a multipart entity in the multipart structure, e.g. in an MMS message, RFC822/2822 e-mail (see reference document [7]) or in the HTTP response body. For SMS, Bluetooth and IrDA-specific mechanisms are needed.

To be able to encapsulate the landmark object, a special MIME content type is defined. MIME is specified in general in reference documents [22] and [23] and in particular for XML media types in reference document [24].

5.1 Landmark MIME type

MIME allows the inclusion of media other than plain text and the inclusion of several entities in one single message. Originally, MIME was specified to extend standard Internet e-mail as specified in reference document [6].

To be able to include landmarks as a separate entity in the multipart structure of a MIME message, a special media type is needed. This facilitates the appropriate dispatching of the media entity by the MIME handler to the correct application. The table below shows the special media types for the landmark objects.

WDP port 3969 is assigned by IANA for landmark messaging.

Name	**MIME type**	**WDP port number**
Landmark Collection (XML)	`application/vnd.nokia.landmarkcollection+xml`	N/A
Landmark (WBXML)	`application/vnd.nokia.landmark+wbxml`	`3969` (0x0F81)

The `application/vnd.nokia.landmark+wbxml` format is intended for the use of resource critical bearers such as SMS.

5.2 WDP (SMS)

The WDP transfer mapping facilitates landmark sharing also in terminals not supporting e.g. MMS or not implementing a 'complete' TCP/IP stack. Additionally, there may be situations where access to e.g. GPRS service is restricted, thus restricting the use of e.g. MMS as the transfer mechanism for landmarks.

WDP provides a simple connectionless, unreliable datagram service optimized for narrowband data bearers in telecommunication networks. The WDP protocol may be seen as the wireless counterpart to UDP in the TCP/IP protocol suite. Reference document [1] even mandates the use of UDP in situations where the underlying bearer service supports IP.

The reason for choosing WDP as the transport mechanism for an SMS bearer is that WDP provides some level of abstraction from the data bearer itself. In other words, WDP has been mapped over a number of various data bearers such as WDP over GSM including profiles for SMS and USSD as well as WDP over CDMA including a profile for SMS, just to mention a few. Since WDP includes profiles for GSM SMS as well as CDMA SMS, the same landmark transfer mechanism over WDP may be used in both environments without changes to the specification or implementation.

WDP also provides mechanisms for Segmentation and Re-Assembly (SAR) if the underlying data bearer does not provide this. This means that a landmark implementation does not have to worry about whether a landmark needs to be segmented before it is sent or re-assembled when received. In other words, SAR is taken care of by the WDP implementation or by the underlying data bearer.

5.2.1 WDP protocol overview

See reference document [1] for details.

The primitive types used in this specification are:

- Request (`.Req`)

  The Request primitive type is used when a higher layer is requesting a service from the next lower layer.

- Indication (`.Ind`)

  The Indication primitive type is used by a layer providing a service to notify the next higher layer of activities related to the Request primitive type of the peer.

The WDP protocol uses a single service primitive (`T-DUnitdata`) to transport data between the peers. If an error occurs, i.e. the WDP layer cannot execute the requested transmission, the user may receive an error indication (`T-DError`).

Note that there is no confirmation that the datagram has been delivered to the destination. The indication only indicates whether the datagram has been successfully sent.

5.2.1.1 T-DUnitdata

`T-DUnitdata` is the primitive used to transmit data as a datagram. `T-DUnitdata` does not require an existing connection to be established. A `T-DUnitdata.Req` can be sent to the WDP layer at any time.

Parameter \ Primitive	T-DUnitdata	
	.Req	.Ind
Source Address	M	M(=)
Source Port	M	M(=)
Destination Address	M	O(=)
Destination Port	M	O(=)
User Data	M	M(=)

Destination Address:

This is the destination address of the user data submitted to the WDP layer. The destination address may be an MSISDN number, IP address, X.25 address or some other identifier. For the purpose of example, Destination Address in an MSISDN number, i.e. E.164 number.

Destination Port:

This is the application address associated with the destination address for the requested communication instance. For the purpose of example, Destination Port is 3969 (0x0F81).

Source Address:

Source Address is the unique address of the device making a request to the WDP layer. Source Address may be an MSISDN number, IP address, X.25 address or some other identifier. For the purpose of example, Source Address in an MSISDN number, i.e. E.164 number.

Source Port:

This is the application address associated with the source address of the requesting communication instance. For the purpose of example, Source Port is 3969 (0x0F81).

User Data:

This is the user data carried by the WDP protocol. The unit of data submitted to or received from the WDP layer is also referred to as the Service Data Unit. This is the complete unit (message, packet, package) of data, which the higher layer has submitted to the WDP layer for transmission. The WDP layer transmits the Service Data Unit and delivers it to its destination without any manipulation of its content. For the purpose of example, the SDU contains the WBXML encoded landmark collection as outlined in Chapter 4, “Compact binary representation of landmark collection XML documents.”

5.2.1.2 T-DError

The `T-DError` primitive is used to provide information to the higher layer when an error occurs that may impact the requested service. The WDP layer may issue a `T-DError` indication only after the higher layer has made a request to the WDP layer, for example, by issuing a `T-DUnitdata` request. The `T-DError` primitive is used when the WDP layer is unable to complete the requested service due to a local problem. It is not used to inform the upper layer of networking errors external to the device/server.

For example, if the upper layer issues a `D-Unitdata` request containing an SDU, which is larger than the maximum SDU size allowed by the specific WDP implementation, a `T-DError` indication is returned to the upper layer with an error code indicating that the SDU size is too large.

If the SDU is too large to be transmitted by the WDP layer, it is up to the implementation to segment the data and invoke subsequent `T-DUnitdata.Req` to transmit the complete data. For example, this situation may occur when trying to send a large landmark collection. However, this situation is outside the scope of this specification.

Primitive / **Parameter**	**T-DError**	
	.Req	**.Ind**
Source Address		O
Source Port		O
Destination Address		O
Destination Port		O
Error Code		M

Error Code:

This is the error code carried by the `T-DError` primitive to the higher layer. The error codes are of local significance only.

The other parameters have the same definitions as for `T-DUnitdata` (see Section 5.2.1.1, “T-DUnitdata”).

5.2.2 General WDP message structure (GSM SMS)

This section shows the general WDP message structure with respect to landmarks. The following aspects are taken into account:

- GSM Phase 2 SMS is required, i.e. support for binary headers. This specification does not specify any GSM Phase 1 SMS text based headers.
- 8-bit reference and 16-bit addressing scheme is assumed.

The WDP datagram is defined within the UDH framework in reference documents [27] and [28].

The WDP datagram structure is the following:

Length of the total user data header (all information elements)
UDH IE identifier: Port numbers (5)
UDH port number IE length (4)
Destination Port (High)
Destination Port (Low)
Originator Port (High)

Originator Port (Low)
UDH IE identifier: SAR (0)
UDH SAR IE length (3)
Datagram Reference number
Total number of segments in the datagram
Segment count
1 - n bytes of User Data

The landmark WDP datagram structure is the following (the shaded part in the table is the user data being transferred):

Hex Value	Meaning	Description
`0x0B`	User data header length (UDHL) = 11 bytes	WDP layer (start WDP headers)
`0x05`	UDH IE identifier: Port numbers	
`0x04`	UDH port number IE length	
`0xC3`	Destination port (high)	Port number 3969
`0x4B`	Destination port (low)	
`0xC3`	Originating port (high)	Port number 3969
`0x4B`	Originating port (low)	
`0x00`	UDH IE identifier: SAR	Indicates that SAR needed
`0x03`	UDH SAR IE length (3)	
`variable`	Datagram reference number	
`variable`	Total number of segments in the datagram	
`variable`	Segment count	WDP Layer (end WDP headers)
`0x03`	WBXML version	Application layer (Start application layer headers) WBXML 1.3
`0xA4`	Public identifier	`-//NOKIA//DTD LANDMARKS 1.0//EN`
`0x04`		
`0x6A`	Character set	UTF-8
`0x00`	String table length	

Hex Value	Meaning	Description
...	User data	WBXML encoded user data

5.2.3 General WDP message structure (CDMA SMS)

A WDP datagram containing N bytes of data sent over an IS-637 SMS (see reference document [29]) has the following structure:

Field	Length (bits)
SOURCE_PORT	16
DESTINATION_PORT	16
DATA	N * 8

The user data structure is the following:

Field	Length (bits)
MSG_TYPE	8
TOTAL_SEGMENTS	8
SEGMENT_NUMBER	8
DATAGRAM	(NUM_FIELDS – 3) * 8

The landmark WDP datagram structure is the following (the shaded part in the table is the user data being transferred):

Hex Value	Meaning	Description
0xC3	Source port (high)	Port number 3969
0x4B	Source port (low)	
0xC3	Destination port (high)	Port number 3969
0x4B	Destination port (low)	
0x00	MSG_TYPE	Set to ‘00000000’ to indicate that this is a WDP message.
variable	TOTAL_SEGMENTS	
variable	SEGMENT_NUMBER	
0x03	WBXML version	Application layer (Start application layer headers) WBXML 1.3
0xA4	Public identifier	-//NOKIA//DTD LANDMARKS 1.0//EN
0x04		

Hex Value	Meaning	Description
`0x6A`	Character set	UTF-8
`0x00`	String table length	
`...`	User data	WBXML encoded user data

5.3 MMS

An MMS message can be used as a type of container for transporting multiple objects such as pictures, text, as well as landmarks. For MMS, the encapsulation of the media objects is specified in reference document [5] and it follows the basic principles specified for MIME in reference documents [22] and [23] and the principles in particular for XML media types in reference document [24]. For the appropriate dispatching by the MIME handler to the correct application, the landmark entity of the multipart structure is identified by its content type, i.e. `application/vnd.nokia.landmarkcollection+xml`.

A single landmark example:

```
[other MMS headers]
… Content-Type: multipart/mixed; boundary="-- simple boundary"
Content-Type: text/plain; charset=us-ascii
Content-Length: nnnn
Hey! Here’s the landmark to the nice restaurant
I told you about...
    -Sebastian
-- simple boundary
Content-Type: application/vnd.nokia.landmarkcollection+xml; charset=utf-8
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<!-- Example landmark xml file. Single landmark with coordinates, two
categories, address, and URL -->
<lm:lmx xmlns:lm="http://www.nokia.com/schemas/location/landmarks/1/0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.nokia.com/schemas/location/landmarks/1/0
lmx.xsd">
  <lm:landmark>
    <lm:name>Nice Thai restaurant</lm:name>
    <lm:description>Good food with nice interior and karaoke</lm:description>
    <lm:coordinates>
      <lm:latitude>65.4321</lm:latitude>
      <lm:longitude>43.2198</lm:longitude>
    </lm:coordinates>
    <lm:addressInfo>
      <lm:country>Finland</lm:country>
      <lm:countryCode>FI</lm:countryCode>
      <lm:city>Helsinki</lm:city>
      <lm:postalCode>00380</lm:postalCode>
      <lm:street>Valimotie 9</lm:street>
      <lm:phoneNumber>+35891234567</lm:phoneNumber>
    </lm:addressInfo>
    <lm:mediaLink>
      <lm:url>http://www.nicerestaurant.com</lm:url>
    </lm:mediaLink>
    <lm:category>
```

```
      <lm:id>9000</lm:id>
      <lm:name>Restaurants</lm:name>
    </lm:category>
    <lm:category>
      <lm:name>Karaoke</lm:name>
    </lm:category>
  </lm:landmark>
</lm:lmx>
-- simple boundary
```

A landmark collection example:

```
[other MMS headers]
…
Content-Type: multipart/mixed; boundary="-- simple boundary"
Content-Type: text/plain; charset=us-ascii
Content-Length: nnnn
Hey! Here's the landmark to the nice restaurant
I told you about...

    -Sebastian

-- simple boundary
Content-Type: application/vnd.nokia.landmarkcollection+xml;
charset=utf-8
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<!-- Example landmark xml file. Two landmarks with coordinates and
categories -->
<lm:lmx
xmlns:lm="http://www.nokia.com/schemas/location/landmarks/1/0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.nokia.com/schemas/location/landmarks/1
/0 lmx.xsd">
  <lm:landmarkCollection>
    <lm:landmark>
      <lm:name>Nice Thai restaurant</lm:name>
      <lm:coordinates>
        <lm:latitude>65.4321</lm:latitude>
        <lm:longitude>43.2198</lm:longitude>
      </lm:coordinates>
      <lm:category>
        <lm:id>9000</lm:id>
        <lm:name>Restaurants</lm:name>
      </lm:category>
    </lm:landmark>
    <lm:landmark>
      <lm:name>Nice Mexican restaurant</lm:name>
      <lm:description>Very spicy food</lm:description>
      <lm:coordinates>
        <lm:latitude>43.2198</lm:latitude>
        <lm:longitude>65.4321</lm:longitude>
      </lm:coordinates>
      <lm:category>
        <lm:id>9000</lm:id>
        <lm:name>Restaurants</lm:name>
      </lm:category>
    </lm:landmark>
  </lm:landmarkCollection>
</lm:lmx>
-- simple boundary
```

5.4 HTTP

For the user to be able to download landmarks from e.g. a Web site, a landmark object may be encapsulated as a multipart entity in the multipart structure of a standard HTTP response (see reference document [3]).

A single landmark example:

```
[other RFC2616 headers]
…
Content-Type: multipart/mixed; boundary="-- simple boundary"
Content-Type: text/html; charset=us-ascii
Content-Length: nnnn
<html>
...some html page...
</html>


-- simple boundary
Content-Type: application/vnd.nokia.landmarkcollection+xml; charset=utf-8
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<!-- Example landmark xml file. Single landmark with coordinates, two
categories, address, and URL -->
<lm:lmx xmlns:lm="http://www.nokia.com/schemas/location/landmarks/1/0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.nokia.com/schemas/location/landmarks/1/0/
lmx.xsd">
  <lm:landmark>
    <lm:name>Nice Thai restaurant</lm:name>
    <lm:description>Good food with nice interior and
karaoke</lm:description>
    <lm:coordinates>
      <lm:latitude>65.4321</lm:latitude>
      <lm:longitude>43.2198</lm:longitude>
    </lm:coordinates>
    <lm:addressInfo>
      <lm:country>Finland</lm:country>
      <lm:countryCode>FI</lm:countryCode>
      <lm:city>Helsinki</lm:city>
      <lm:postalCode>00380</lm:postalCode>
      <lm:street>Valimotie 9</lm:street>
      <lm:phoneNumber>+35891234567</lm:phoneNumber>
    </lm:addressInfo>
    <lm:mediaLink>
      <lm:url>http://www.nicerestaurant.com</lm:url>
    </lm:mediaLink>
    <lm:category>
      <lm:id>9000</lm:id>
      <lm:name>Restaurants</lm:name>
    </lm:category>
    <lm:category>
      <lm:name>Karaoke</lm:name>
    </lm:category>
  </lm:landmark>
</lm:lmx>
-- simple boundary
```

A landmark collection example:

```
[other RFC2616 headers]
…
Content-Type: multipart/mixed; boundary="-- simple boundary"
Content-Type: text/html; charset=us-ascii
Content-Length: nnnn
<html>
...some html page...
</html>


-- simple boundary
Content-Type: application/vnd.nokia.landmarkcollection+xml;
charset=utf-8
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<!-- Example landmark xml file. Two landmarks with coordinates and
categories -->
<lm:lmx xmlns:lm="http://www.nokia.com/schemas/location/landmarks/1/0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.nokia.com/schemas/location/landmarks/1/0
lmx.xsd">
  <lm:landmarkCollection>
    <lm:landmark>
      <lm:name>Nice Thai restaurant</lm:name>
      <lm:coordinates>
        <lm:latitude>65.4321</lm:latitude>
        <lm:longitude>43.2198</lm:longitude>
      </lm:coordinates>
      <lm:category>
        <lm:id>9000</lm:id>
        <lm:name>Restaurants</lm:name>
      </lm:category>
    </lm:landmark>
    <lm:landmark>
      <lm:name>Nice Mexican restaurant</lm:name>
      <lm:description>Very spicy food</lm:description>
      <lm:coordinates>
        <lm:latitude>43.2198</lm:latitude>
        <lm:longitude>65.4321</lm:longitude>
      </lm:coordinates>
      <lm:category>
        <lm:id>9000</lm:id>
        <lm:name>Restaurants</lm:name>
      </lm:category>
    </lm:landmark>
  </lm:landmarkCollection>
</lm:lmx>
-- simple boundary
```

5.5 Email

The mechanism to encapsulate landmark objects into a standard Internet mail or RFC822/2822 e-mail (see reference document [7]) is similar to that of MMS, see Section 5.3.

A single landmark example:

```
[other RFC822/2822 headers]
…
Content-Type: multipart/mixed; boundary="-- simple boundary"
-- simple boundary
Content-Type: text/plain; charset=us-ascii
Content-Length: nnnn

Hey! Here’s the landmark to the nice restaurant
I told you about...

    - Sebastian

-- simple boundary
Content-Type: application/vnd.nokia.landmarkcollection+xml; charset=utf-8
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<!-- Example landmark xml file. Single landmark with coordinates, two
categories, address, and URL -->
<lm:lmx xmlns:lm="http://www.nokia.com/schemas/location/landmarks/1/0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.nokia.com/schemas/location/landmarks/1/0/
lmx.xsd">
  <lm:landmark>
    <lm:name>Nice Thai restaurant</lm:name>
    <lm:description>Good food with nice interior and
karaoke</lm:description>
    <lm:coordinates>
      <lm:latitude>65.4321</lm:latitude>
      <lm:longitude>43.2198</lm:longitude>
    </lm:coordinates>
    <lm:addressInfo>
      <lm:country>Finland</lm:country>
      <lm:countryCode>FI</lm:countryCode>
      <lm:city>Helsinki</lm:city>
      <lm:postalCode>00380</lm:postalCode>
      <lm:street>Valimotie 9</lm:street>
      <lm:phoneNumber>+35891234567</lm:phoneNumber>
    </lm:addressInfo>
    <lm:mediaLink>
      <lm:url>http://www.nicerestaurant.com</lm:url>
    </lm:mediaLink>
    <lm:category>
      <lm:id>9000</lm:id>
      <lm:name>Restaurants</lm:name>
    </lm:category>
    <lm:category>
      <lm:name>Karaoke</lm:name>
    </lm:category>
  </lm:landmark>
</lm:lmx>
-- simple boundary
```

A landmark collection example:

```
[other RFC822/2822 headers]
…
Content-Type: multipart/mixed; boundary="-- simple boundary"
-- simple boundary
Content-Type: text/plain; charset=us-ascii
Content-Length: nnnn

Hey! Here’s the landmark to the nice restaurant
I told you about...

    -Sebastian

-- simple boundary
Content-Type: application/vnd.nokia.landmarkcollection+xml; charset=utf-8
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<!-- Example landmark xml file. Two landmarks with coordinates and
categories -->
<lm:lmx
xmlns:lm="http://www.nokia.com/schemas/location/landmarks/1/0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.nokia.com/schemas/location/landmarks/1
/0 lmx.xsd">
  <lm:landmarkCollection>
    <lm:landmark>
      <lm:name>Nice Thai restaurant</lm:name>
      <lm:coordinates>
        <lm:latitude>65.4321</lm:latitude>
        <lm:longitude>43.2198</lm:longitude>
      </lm:coordinates>
      <lm:category>
        <lm:id>9000</lm:id>
        <lm:name>Restaurants</lm:name>
      </lm:category>
    </lm:landmark>
    <lm:landmark>
      <lm:name>Nice Mexican restaurant</lm:name>
      <lm:description>Very spicy food</lm:description>
      <lm:coordinates>
        <lm:latitude>43.2198</lm:latitude>
        <lm:longitude>65.4321</lm:longitude>
      </lm:coordinates>
      <lm:category>
        <lm:id>9000</lm:id>
        <lm:name>Restaurants</lm:name>
      </lm:category>
    </lm:landmark>
  </lm:landmarkCollection>
</lm:lmx>
-- simple boundary
```

5.6 OBEX

OBEX is a compact, efficient, binary protocol that enables a wide range of devices to exchange data in a simple and spontaneous manner (see reference document [8]). OBEX provides the same basic functionality as HTTP and uses a client-server model. OBEX has been developed by IrDA, and also Bluetooth SIG has adopted the protocol; i.e. it possible to exchange objects between devices both via an Infrared or Bluetooth connection using OBEX. Consequently, it is possible to specify the exchange of landmarks using OBEX once and then profile it for IrDA and Bluetooth respectively.

In specifying the exchange of landmarks with OBEX, the following rules apply:

- The device sending the landmarks is considered as the client.
- The device receiving the landmarks is considered as the server.
- The exchange of landmarks is realized using the Inbox service in OBEX.
- Only the Push of landmarks from the client to the server is required and supported.
- The landmark data is identified by its MIME type, `application/vnd.nokia.landmarkcollection+xml`.
- There can be only one landmark collection object per PUT operation.
- The minimum OBEX version required is OBEX 1.0.

5.6.1 Connection establishment

Before a landmark can be sent, a connection must be established between the client and the server. This is accomplished by the client sending a `CONNECT` packet to the server and the server responding with a `SUCCESS` packet.

5.6.1.1 CONNECT

General request format:

Byte 0	**Bytes 1 and 2**	**Byte 3**	**Byte 4**	**Bytes 5 and 6**	**Byte 7 to n**
0x80	`CONNECT` packet length	OBEX version number	Flags	Maximum OBEX packet length	Optional headers

Response format:

Byte 0	**Bytes 1 and 2**	**Byte 3**	**Byte 4**	**Bytes 5 and 6**	**Byte 7 to n**
Response code	`CONNECT` response packet length	OBEX version number	Flags	Maximum OBEX packet length	Optional headers

Example:

The example below shows how a connection can be established between the client and server. By sending a CONNECT without any target headers, the Inbox is made the default OBEX Server.

Client request:	**bytes**	**Meaning**
opcode	0x80	`CONNECT`, final bit set.
	0x0007	The length of the packet is 7 bytes.
	0x10	OBEX version 1.0.
	0x00	No connect flags.
	0x400	1K is the maximum packet size.
Server response:		
response code	0xA0	`SUCCESS`, final bit set.
	0x0007	The length of the packet is 7 bytes.
	0x10	OBEX version 1.0.
	0x00	No connect flags.
	0x400	1K is the maximum packet size.

Note: Some specific headers in `CONNECT` can be used, e.g. the description header containing information about the device or service. It is recommended that the information is presented through the user interface on the receiving side.

5.6.2 Landmark collection Push

After the connection is established, the client can push a landmark collection to the server using the `PUT` operation. As the connection was established without any target headers, the default OBEX Server is the Inbox connection.

Note: Reference document [8] highly recommends that implementations assume default parameters for a connection and accept operations such as `PUT` and `GET` without first requiring `CONNECT`.

For the purpose of sharing landmarks, only the `PUT` operation is required to push a landmark collection from the client to the server's Inbox. When the client pushes a landmark collection to the server, the client should set the Type header according to the MIME type specified for landmarks, i.e. `application/vnd.nokia.landmarkcollection+xml`. This enables the server to easily recognize the content type without looking at the data and to dispatch the correct handler without any further processing of the data.

Note: It is not mandatory for an implementation to support multiple objects per `PUT` operation. Therefore, the client should limit the number of landmarks per `PUT` operation to a single landmark.

5.6.2.1 PUT

General request format:

Byte 0	Bytes 1, 2	Bytes 3 to n
0x02 (0x82 when final bit is set)	Packet length	Sequence of headers

General response format:

Byte 0	Bytes 1, 2	Bytes 3 to n
Response code Typical values: • 0x90 for Continue • 0xA0 for Success	Response packet length	Optional response headers

Landmark request format (all data in one packet)

Note that this format assumes that the landmark object fits in one `PUT` packet, i.e. that the Length of Object (LoO) + opcode + all the headers and header values are less than the maximum packet size.

Byte	0	1, 2	3	4,5	6-91	92	93,94	95	96,97	98+ LoO	98 +LoO +1	98 +LoO +3
Value	0x82	0x…	0x42	0x45	0x…[1)]	0xC3	0x…	0x48	0x…[2)]	0x…[3)]	0x49	0x0003
Desc	Opcode Final bit set	Length of the packet	HI[4)] Type	Length	HV[5)]	HI Length	HV LoO	HI Body	HV	HV Data	End of body	HV Length

1. `application/vnd.nokia.landmarkcollection+xml` as a NULL-terminated Unicode string.
2. Length of Object + 3 (HI = 1 byte + Length = 2 bytes).
3. The landmark data itself.
4. HI = Header ID
5. HV = Header Value

Landmark request format (multiple packets):

With this format, the data is “chunked” over several packets as the total size is more than the maximum allowed packet size. Typically, a landmark contains a limited amount of data of less than approximately 1K. Therefore, the preferred war is to always send the landmark in one packet given that both the client and the server support a maximum packet length greater than the total size of the PUT operation and the landmark data. However, there are no size limitations on landmarks or landmark collections, and consequently, the data sent may be 'chunked' over several packets.

Byte	**0**	**1, 2**	**3**	**4,5**	**6-91**	**92**	**93,94**	**95**	**96,97**	**98+ LoC**[1)]
Value	0x02	0x…	0x42	0x45	0x…	0xC3	0x…	0x48	0x… [2)]	0x…
Desc	Opcode (Final bit not set)	Length of the packet	HI (Type)	Length	HV	HI (Length)	HV (Length of the chunk)	HI (Body)	HV	HV (Data)

1. LoC = Length of Chunk.
2. The body length is set to LoC + 3 (HI = 1 byte + Length = 2 bytes).

Landmark response format (CONTINUE, final packet not received)

Byte	**0**	**1, 2**
Value	0x90	0x0003
Desc	opcode (CONTINUE, final bit set)	Length of the packet

Landmark response format (final packet received)

Byte	**0**	**1, 2**
Value	0xA0	0x0003
Desc	opcode (SUCCESS, Final bit sent)	Length of the packet

5.6.3 Closing the connection

5.6.3.1 DISCONNECT

During one session, the client may push several landmarks or landmark collections to the server using multiple `PUT` operations. After the client is finished sending the landmarks, the connection is closed with `DISCONNECT`. It is not mandatory to do `DISCONNECT` even though it is recommended.

Details on `DISCONNECT` can be found in reference document [8].

5.7 Bluetooth

Section 5.6 outlined how landmarks can be sent from one device to another using OBEX. Since OBEX is one of the adopted protocols for Bluetooth, sending landmarks over a Bluetooth link is fairly transparent. Implementation should be fairly straightforward as code developed for sending vCard, vCal, etc. can be re-used. Hence, the information contained herein is only an overview of the basic aspects. Additionally, how the Bluetooth connection is established is beyond the scope of this document.

For more information, see reference documents [8], [25] and [26] as well as other potential implementation specifications utilizing OBEX over Bluetooth.

5.7.1 Bluetooth profile structure

The usage scenario of sending a landmark from one device to another using an Inbox connection as the default OBEX Server fits with the Object Push profile specified in reference document [26] for Bluetooth.

The relation of the Object Push profile to the other Bluetooth profiles is shown in Figure 5.1.

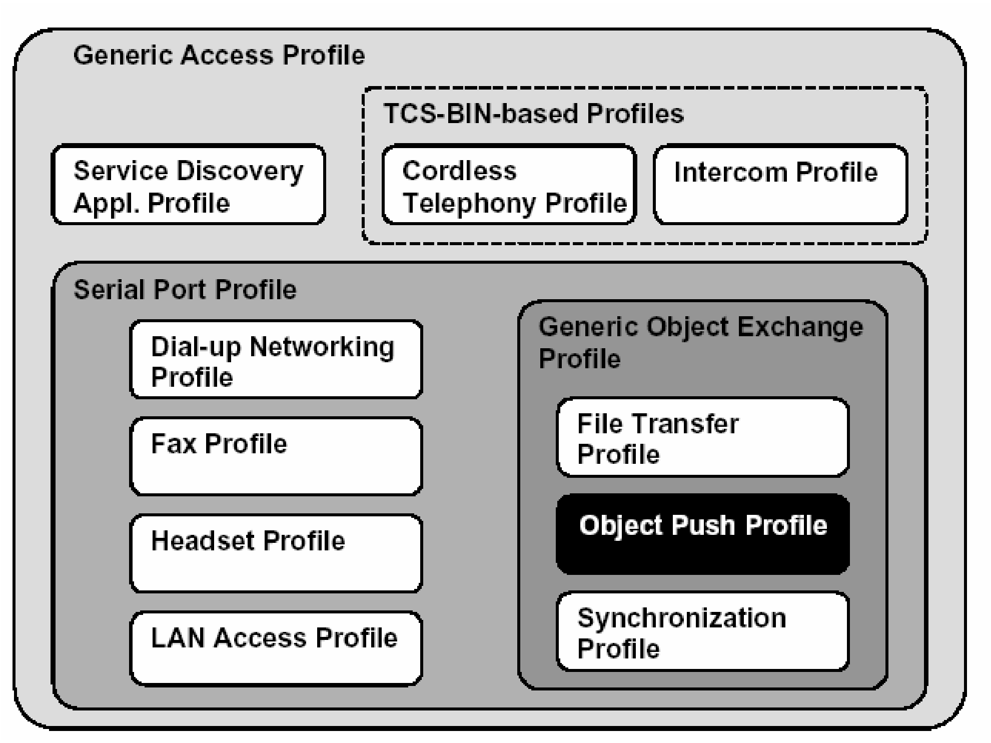


Figure 5.1: Bluetooth profile structure

Figure 5.1 shows that the Object Push profile relies on a number of other profiles, namely Generic Object Exchange profile, Serial Port profile and Generic Access profile.

Information on the various profiles including the interoperability requirements for each profile can be found in reference document [26].

5.7.2 Object Push profile stack

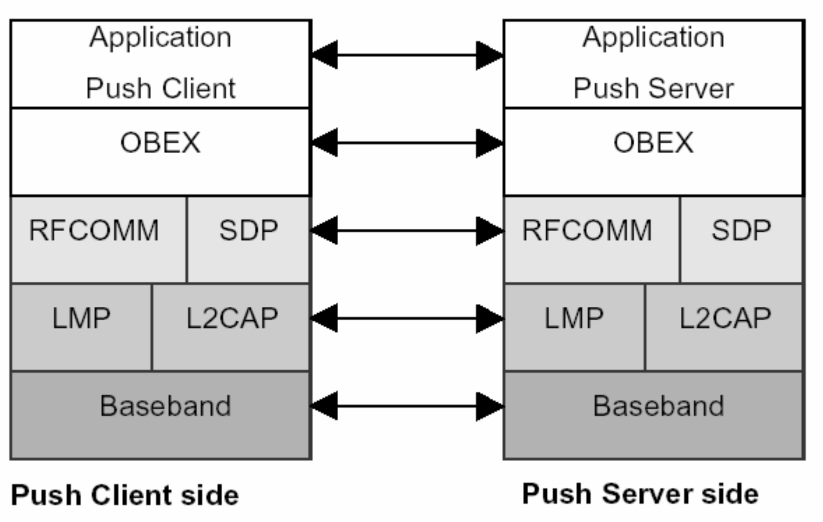


Figure 5.2: Object Push profile stack

5.7.3 Application usage events

Push Client (sending device)	Push Server (receiving device)
	The user enables Bluetooth communication and sets the device into the Object Exchange mode.
The user of the Push Client marks the landmark to be sent and selects the Object Push function on the device.	
	When an object is received in Push Server, it is recommended that the user of Push Server be asked to accept or reject the object.

5.7.4 Application procedure

Push Client (sending device)	Details
`OBEX CONNECT`	Target header must no be used.
One or more `OBEX PUTs` for sending one or more objects.	As an implementation is not required to support multiple objects in one OBEX PUT, the number of landmarks in a PUT is restricted to one.
`OBEX DISCONNECT`	

5.8 Infrared

The landmarks are sent over IrDA Object Exchange Protocol, OBEX, see reference document [8].

Only the provisions forth in Sections 5.6 and 5.7 are given for IrDA.

5.9 File exchange

Landmark data may be also exchanged between terminals in the form of raw landmark data within a file in a file system. For example, landmarks may be stored in the memory card for the purpose of transmitting it to another phone. When the memory card is inserted into another phone or the file is copied into another phone's file system by any means, it may be acquired by the landmark software installed in that phone.

The landmark data must be stored into a file in raw form as defined by the landmarks' MIME types. For the purposes of distinguishing the landmark file from other files, it is recommended to use the LMX file extension (called after the XML schema root element). However, usage of data recognizers is recommended for applications.

Note: The actual format (XML or binary-encoded XML) of landmark data is not specified and thus, an application supporting file-based landmarks exchange needs to be prepared to correctly handle such a file regardless of the actual format used.

6. Terms and abbreviations

Term or abbreviation	Meaning
ABNF	Augmented Backus-Naur Notation
DTD	Document type definition
EBNF	Extended Backus-Naur Form
HTTP	Hypertext transfer protocol
IANA	Internet Assigned Numbers Authority
IE	Information Element
Landmark	An object used to deliver waypoint or Point-of-Interest information.
LCS	Location Service
MIME	Multipurpose Internet mail extensions
MMS	Multimedia messaging service
Point-of-Interest	Point-of-Interests comprise categorized location (destination) information. POI datasets are typically made available by a publisher and they are interesting for a wide audience.
QoP	Quality of Position
SAR	Separation and Re-Assembly
SMS	Short Message Service
UCS-2	Universal multiple-octet Character-Set
UDH	User Data Header
URI	Uniform resource identifier
URL	Uniform resource locator
UTF-8	UCS Transformation Format
Waypoint	A waypoint comprises information about a user-defined and named location. It typically consists of a short name and coordinates. Waypoints are of the user's personal interest.
WBXML	Wireless binary Extensible markup language
WDP	Wireless datagram protocol
WSP	Wireless Session Protocol
WGS-84	World Geodetic Standard 1984
XML	Extensible markup language

7. References

[1] Wireless Datagram Protocol Specification, http://www.openmobilealliance.org/

[2] Wireless Session Protocol Specification, http://www.openmobilealliance.org/

[3] Hypertext Transfer Protocol -- HTTP/1.1, RFC2616, http://www.ietf.org/rfc/rfc2616.txt

[4] WAP Binary XML Content Format, http://www.w3.org/TR/wbxml/

[5] OMA MMS Encapsulation Protocol, http://www.openmobilealliance.org/

[6] Standard for the format of ARPA Internet text Messages, RFC822, http://www.ietf.org/rfc/rfc822.txt

[7] Internet Message Format, RFC2822, http://www.ietf.org/rfc/rfc2822.txt

[8] OBEX, IrDA Object Exchange Protocol, http://www.irda.org/

[9] Extensible Markup Language (XML) 1.0, W3C Recommendation, http://www.w3.org/TR/REC-xml

[10] XML Schema, http://www.w3.org/XML/Schema

[11] MIME Media Types, http://www.iana.org/assignments/media-types/index.html

[12] IANA MIBenum Character Set Registry, http://www.iana.org/assignments/character-sets

[13] IANA Directory of General Assigned Numbers, Port Number Assignments, http://www.iana.org/assignments/port-numbers

[14] WINA Name and Number Database, WSP Content Types, http://www.wapforum.org/wina/wsp-content-type.htm

[15] ISO-8601 Data elements and interchange formats – Information interchange – Representation of dates and times, http://www.pvv.ntnu.no/~nsaa/8601v2000.pdf

[16] ISO/IEC-10646 Universal multiple-octet Character-Set,
http://en.wikipedia.org/wiki/ISO_10646
http://anubis.dkuug.dk/JTC1/SC2/WG2/

[17] DOD World Geodetic System 1984, Definition and Relationships with Local Geodetic Systems, http://earth-info.nga.mil/GandG/publications/index.htm

[18] RFC2279, UTF-8, a transformation format of ISO 10646, http://www.ietf.org/rfc/rfc2279.txt

[19] Location API for J2ME™, Version 1.0 Community Review draft, version 0.6 of the JSR179 Location API, http://www.jcp.org/en/jsr/detail?id=179

[20] XML Schema Part 1: Structures Second Edition, http://www.w3.org/TR/xmlschema-1/

[21] XML Schema Part 2: Datatypes Second Edition, http://www.w3.org/TR/xmlschema-2/

[22] Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, RFC2045, http://www.ietf.org/rfc/rfc2045.txt

[23] Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types, RFC2046, http://www.ietf.org/rfc/rfc2046.txt

[24] XML Media Types, RFC3023, http://www.ietf.org/rfc/rfc3023.txt

[25] Bluetooth V1.1 Specification (Core), http://www.bluetooth.org/spec/

[26] Bluetooth V1.1 Specification (Profiles), http://www.bluetooth.org/spec/

[27] Digital cellular telecommunications system (Phase 2) (GSM); Technical realization of Short Message Service (SMS) Point-to-Point (PP) (GSM 03.40), http://www.etsi.org

[28] Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); Technical realization of Short Message Service (SMS) (3GPP TS 23.040 version 3.10.0 Release 1999), http://www.etsi.org

[29] Short Message Service, http://www.3gpp2.org/Public_html/specs/CS0015-0.pdf

[30] Extensible Markup Language (XML) 1.0, http://www.w3.org/TR/REC-xml

[31] IEEE Standard for Binary Floating-Point Arithmetic, IEEE 754-1985, http://en.wikipedia.org/wiki/IEEE_754 http://standards.ieee.org/

[32] RFC 2396. Uniform Resource Identifiers (URI): Generic Syntax, http://www.ietf.org/rfc/rfc2396.txt

[33] RFC 2732: Format for Literal IPv6 Addresses in URL's, http://www.ietf.org/rfc/rfc2732.txt

[34] XML Linking Language (XLink) Version 1.0, http://www.w3.org/TR/2000/PR-xlink-20001220/

Appendix A. Landmarks exchange format XML schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Landmarks Exchange Format XML Schema. Copyright @ 2004 Nokia.-->
<xsd:schema
targetNamespace="http://www.nokia.com/schemas/location/landmarks/1/0"
elementFormDefault="qualified" attributeFormDefault="qualified"
xmlns="http://www.nokia.com/schemas/location/landmarks/1/0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="lmx">
    <xsd:annotation>
      <xsd:documentation>Root element of landmarks XML exchange file. May
contain either a single landmark or a collection of
landmarks.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:choice>
        <xsd:annotation>
          <xsd:documentation>Choice</xsd:documentation>
        </xsd:annotation>
        <xsd:element name="landmark" type="landmarkType">
          <xsd:annotation>
            <xsd:documentation>A single landmark.</xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element name="landmarkCollection" type="landmarkCollectionType">
          <xsd:annotation>
            <xsd:documentation>Collection of landmarks.</xsd:documentation>
          </xsd:annotation>
        </xsd:element>
      </xsd:choice>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="landmarkCollectionType">
    <xsd:annotation>
      <xsd:documentation>The landmark collection type. The landmark
collection includes a set of landmarks. At least one landmark is required.
The landmark collection may have name and description, which both are mostly
intended for displaying purposes.</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:annotation>
        <xsd:documentation>Sequence</xsd:documentation>
      </xsd:annotation>
      <xsd:element name="name" type="xsd:string" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>The name of the collection. Optional data, used
for displaying purposes.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="description" type="xsd:string" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>The description of the collection. Optional
data, used for displaying purposes. Usually it's more detailed than name or
provides some additional information about the
collection.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="landmark" type="landmarkType" maxOccurs="unbounded">
        <xsd:annotation>
          <xsd:documentation>A landmark.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="landmarkType">
    <xsd:annotation>
      <xsd:documentation>The landmark type. The landmark is a point on map
or place in the world, which position is defined by coordinates. The
landmarks may belong to a set of landmark categories. It also may have
additional address information of the place this landmark points to. The
landmark also may have additional information, like description, coverage
radius and a list of media links.</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:annotation>
        <xsd:documentation>Sequence</xsd:documentation>
      </xsd:annotation>
      <xsd:element name="name" type="xsd:string" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>The name of the landmark. Optional data, used
for displaying purposes.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="description" type="xsd:string" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>The description of the landmark. Optional data,
used for displaying purposes. Usually it's more detailed than name or
provides some additional information about the landmark.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="coordinates" type="coordinatesType" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>The landmark's coordinates.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="coverageRadius" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>The coverage radius of the landmark in meters.
The landmark's coordinates are the center of the area this landmark
covers.</xsd:documentation>
        </xsd:annotation>
        <xsd:simpleType>
          <xsd:restriction base="xsd:float">
            <xsd:minInclusive value="0"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="addressInfo" type="addressInfoType" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>The landmark's address
information.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="mediaLink" type="mediaLinkType" minOccurs="0"
maxOccurs="unbounded">
        <xsd:annotation>
          <xsd:documentation>A media link associated with this
place.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="category" type="categoryType" minOccurs="0"
maxOccurs="unbounded">
        <xsd:annotation>
          <xsd:documentation>Landmark category.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="coordinatesType">
    <xsd:annotation>
      <xsd:documentation>The landmarks coordinates type. Landmark
coordinates must be specified in WGS84 format. Latitude and longitude values
are required. Restrictions applied to coordinate values.</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:annotation>
        <xsd:documentation>Sequence</xsd:documentation>
      </xsd:annotation>
      <xsd:element name="latitude">
        <xsd:annotation>
          <xsd:documentation>The Latitude part of WGS84 coordinate in
degrees. Must be in the range [-90;+90].
</xsd:documentation>
        </xsd:annotation>
        <xsd:simpleType>
          <xsd:restriction base="xsd:double">
            <xsd:minInclusive value="-90"/>
            <xsd:maxInclusive value="90"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="longitude">
        <xsd:annotation>
          <xsd:documentation>The Longitude part of WGS84 coordinate in
degrees. Must be in the range [-180;+180).</xsd:documentation>
        </xsd:annotation>
        <xsd:simpleType>
          <xsd:restriction base="xsd:double">
            <xsd:minInclusive value="-180"/>
            <xsd:maxExclusive value="180"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="altitude" type="xsd:float" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>The Altitude part of WGS84 coordinate in
meters.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="horizontalAccuracy" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>Horizontal accuracy of the coordinates in
meters.</xsd:documentation>
        </xsd:annotation>
        <xsd:simpleType>
          <xsd:restriction base="xsd:float">
            <xsd:minInclusive value="0"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="verticalAccuracy" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>Vertical accuracy of the coordinate in
meters.</xsd:documentation>
        </xsd:annotation>
        <xsd:simpleType>
          <xsd:restriction base="xsd:float">
            <xsd:minInclusive value="0"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="timeStamp" type="xsd:dateTime" minOccurs="0">
        <xsd:annotation>
```

```
          <xsd:documentation>The timestamp of the moment the coordinates
were calculated. May contain date, time and time zone
information.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="categoryType">
    <xsd:annotation>
      <xsd:documentation>The landmark category type. Category is a name and
optional global landmark category ID.</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:annotation>
        <xsd:documentation>Sequence</xsd:documentation>
      </xsd:annotation>
      <xsd:element name="id" type="xsd:unsignedShort" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>The ID of a global landmark
category.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="name" type="xsd:string">
        <xsd:annotation>
          <xsd:documentation>The name of the landmark category. Must be non-
empty.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="addressInfoType">
    <xsd:annotation>
      <xsd:documentation>The landmark address information type. Address is
divided into fields, each of which represent a part of address information.
The order of fields is not specified.</xsd:documentation>
    </xsd:annotation>
    <xsd:all>
      <xsd:annotation>
        <xsd:documentation>All</xsd:documentation>
      </xsd:annotation>
      <xsd:element name="country" type="xsd:string" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>Address field denoting
country.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="countryCode" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>Address field denoting country as two-letter
ISO-3166-1 code.</xsd:documentation>
        </xsd:annotation>
        <xsd:simpleType>
          <xsd:restriction base="xsd:token">
            <xsd:length value="2"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="state" type="xsd:string" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>Address field denoting state or
province.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="county" type="xsd:string" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>Address field denoting county, an entity
between state and city.</xsd:documentation>
```

```
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="city" type="xsd:string" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>Address field denoting city or town
name.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="district" type="xsd:string" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>Address field denoting municipal
district.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="postalCode" type="xsd:string" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>Address field denoting zip or postal
code.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="crossing1" type="xsd:string" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>Address field denoting a street in a
crossing.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="crossing2" type="xsd:string" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>Address field denoting another street in a
crossing.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="street" type="xsd:string" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>Address field denoting street name and
number.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="buildingName" type="xsd:string" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>Address field denoting a building
name.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="buildingZone" type="xsd:string" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>Address field denoting a building
zone.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="buildingFloor" type="xsd:string" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>Address field denoting a building
floor.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="buildingRoom" type="xsd:string" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>Address field denoting a building
room.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="extension" type="xsd:string" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>Address field denoting address extension, e.g.
flat number.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="phoneNumber" type="xsd:string" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>Address field denoting a phone number for this
place.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
    </xsd:all>
  </xsd:complexType>
  <xsd:complexType name="mediaLinkType">
    <xsd:annotation>
      <xsd:documentation>The media link information type. A media link is a
URI and optional displayable name.</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:annotation>
        <xsd:documentation>Sequence</xsd:documentation>
      </xsd:annotation>
      <xsd:element name="name" type="xsd:string" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>Displayable name of the media
link.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="mime" type="xsd:string" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>MIME type of the referenced
data.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="url" type="xsd:anyURI">
        <xsd:annotation>
          <xsd:documentation>URI of the media link.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Appendix B. Landmarks WBXML tag codes

In the table below, the column names mean the following:

- HEX: the element code in hexadecimal format
- A: element contains attribute(s), END token expected
- C: element has contents, END token expected
- AC: element has both attribute(s) and content, END token expected

The shaded area indicates valid tokens. The empty elements should be excluded from a message.

Element	HEX	7	6	5	4	3	3	1	0	A	7	6	5	4	3	2	1	0	C	7	6	5	4	3	2	1	0	AC	7	6	5	4	3	2	1	0
`lmx`	**05**	0	0	0	0	0	1	0	1	**85**	1	0	0	0	0	1	0	1	**45**	0	1	0	0	0	1	0	1	**C5**	1	1	0	0	0	1	0	1
`landmarkCollection`	**06**	0	0	0	0	0	1	1	0	**86**	1	0	0	0	0	1	1	0	**46**	0	1	0	0	0	1	1	0	**C6**	1	1	0	0	0	1	1	0
`landmark`	**07**	0	0	0	0	0	1	1	1	**87**	1	0	0	0	0	1	1	1	**47**	0	1	0	0	0	1	1	1	**C7**	1	1	0	0	0	1	1	1
`name`	**08**	0	0	0	0	1	0	0	0	**88**	1	0	0	0	1	0	0	0	**48**	0	1	0	0	1	0	0	0	**C8**	1	1	0	0	1	0	0	0
`description`	**09**	0	0	0	0	1	0	0	1	**89**	1	0	0	0	1	0	0	1	**49**	0	1	0	0	1	0	0	1	**C9**	1	1	0	0	1	0	0	1
`coordinates`	**0A**	0	0	0	0	1	0	1	0	**8A**	1	0	0	0	1	0	1	0	**4A**	0	1	0	0	1	0	1	0	**CA**	1	1	0	0	1	0	1	0
`latitude`	**0B**	0	0	0	0	1	0	1	1	**8B**	1	0	0	0	1	0	1	1	**4B**	0	1	0	0	1	0	1	1	**CB**	1	1	0	0	1	0	1	1
`longitude`	**0C**	0	0	0	0	1	1	0	0	**8C**	1	0	0	0	1	1	0	0	**4C**	0	1	0	0	1	1	0	0	**CC**	1	1	0	0	1	1	0	0
`altitude`	**0D**	0	0	0	0	1	1	0	1	**8D**	1	0	0	0	1	1	0	1	**4D**	0	1	0	0	1	1	0	1	**CD**	1	1	0	0	1	1	0	1
`horizontalAccuracy`	**0E**	0	0	0	0	1	1	1	0	**8E**	1	0	0	0	1	1	1	0	**4E**	0	1	0	0	1	1	1	0	**CE**	1	1	0	0	1	1	1	0
`verticalAccuracy`	**0F**	0	0	0	0	1	1	1	1	**8F**	1	0	0	0	1	1	1	1	**4F**	0	1	0	0	1	1	1	1	**CF**	1	1	0	0	1	1	1	1
`timeStamp`	**10**	0	0	0	1	0	0	0	0	**90**	1	0	0	1	0	0	0	0	**50**	0	1	0	1	0	0	0	0	**D0**	1	1	0	1	0	0	0	0
`coverageRadius`	**11**	0	0	0	1	0	0	0	1	**91**	1	0	0	1	0	0	0	1	**51**	0	1	0	1	0	0	0	1	**D1**	1	1	0	1	0	0	0	1
`category`	**12**	0	0	0	1	0	0	1	0	**92**	1	0	0	1	0	0	1	0	**52**	0	1	0	1	0	0	1	0	**D2**	1	1	0	1	0	0	1	0
`id`	**13**	0	0	0	1	0	0	1	1	**93**	1	0	0	1	0	0	1	1	**53**	0	1	0	1	0	0	1	1	**D3**	1	1	0	1	0	0	1	1
`addressInfo`	**14**	0	0	0	1	0	1	0	0	**94**	1	0	0	1	0	1	0	0	**54**	0	1	0	1	0	1	0	0	**D4**	1	1	0	1	0	1	0	0
`country`	**15**	0	0	0	1	0	1	0	1	**95**	1	0	0	1	0	1	0	1	**55**	0	1	0	1	0	1	0	1	**D5**	1	1	0	1	0	1	0	1
`countryCode`	**16**	0	0	0	1	0	1	1	0	**96**	1	0	0	1	0	1	1	0	**56**	0	1	0	1	0	1	1	0	**D6**	1	1	0	1	0	1	1	0
`state`	**17**	0	0	0	1	0	1	1	1	**97**	1	0	0	1	0	1	1	1	**57**	0	1	0	1	0	1	1	1	**D7**	1	1	0	1	0	1	1	1
`county`	**18**	0	0	0	1	1	0	0	0	**98**	1	0	0	1	1	0	0	0	**58**	0	1	0	1	1	0	0	0	**D8**	1	1	0	1	1	0	0	0
`city`	**19**	0	0	0	1	1	0	0	1	**99**	1	0	0	1	1	0	0	1	**59**	0	1	0	1	1	0	0	1	**D9**	1	1	0	1	1	0	0	1
`district`	**1A**	0	0	0	1	1	0	1	0	**9A**	1	0	0	1	1	0	1	0	**5A**	0	1	0	1	1	0	1	0	**DA**	1	1	0	1	1	0	1	0

Element	HEX	7	6	5	4	3	3	1	0	A	7	6	5	4	3	2	1	0	C	7	6	5	4	3	2	1	0	AC	7	6	5	4	3	2	1	0
postalCode	1B	0	0	0	1	1	0	1	1	9B	1	0	0	1	1	0	1	1	5B	0	1	0	1	1	0	1	1	DB	1	1	0	1	1	0	1	1
crossing1	1C	0	0	0	1	1	1	0	0	9C	1	0	0	1	1	1	0	0	5C	0	1	0	1	1	1	0	0	DC	1	1	0	1	1	1	0	0
crossing2	1D	0	0	0	1	1	1	0	1	9D	1	0	0	1	1	1	0	1	5D	0	1	0	1	1	1	0	1	DD	1	1	0	1	1	1	0	1
street	1E	0	0	0	1	1	1	1	0	9E	1	0	0	1	1	1	1	0	5E	0	1	0	1	1	1	1	0	DE	1	1	0	1	1	1	1	0
buildingName	1F	0	0	0	1	1	1	1	1	9F	1	0	0	1	1	1	1	1	5F	0	1	0	1	1	1	1	1	DF	1	1	0	1	1	1	1	1
buildingFloor	20	0	0	1	0	0	0	0	0	A0	1	0	1	0	0	0	0	0	60	0	1	1	0	0	0	0	0	E0	1	1	1	0	0	0	0	0
buildingZone	21	0	0	1	0	0	0	0	1	A1	1	0	1	0	0	0	0	1	61	0	1	1	0	0	0	0	1	E1	1	1	1	0	0	0	0	1
buildingRoom	22	0	0	1	0	0	0	1	0	A2	1	0	1	0	0	0	1	0	62	0	1	1	0	0	0	1	0	E2	1	1	1	0	0	0	1	0
extension	23	0	0	1	0	0	0	1	1	A3	1	0	1	0	0	0	1	1	63	0	1	1	0	0	0	1	1	E3	1	1	1	0	0	0	1	1
phoneNumber	24	0	0	1	0	0	1	0	0	A4	1	0	1	0	0	1	0	0	64	0	1	1	0	0	1	0	0	E4	1	1	1	0	0	1	0	0
mediaLink	25	0	0	1	0	0	1	0	1	A5	1	0	1	0	0	1	0	1	65	0	1	1	0	0	1	0	1	E5	1	1	1	0	0	1	0	1
mime	26	0	0	1	0	0	1	1	0	A6	1	0	1	0	0	1	1	0	66	0	1	1	0	0	1	1	0	E6	1	1	1	0	0	1	1	0
url	27	0	0	1	0	0	1	1	1	A7	1	0	1	0	0	1	1	1	67	0	1	1	0	0	1	1	1	E7	1	1	1	0	0	1	1	1

Appendix C. Predefined global landmark categories

The table below shows the predefined global landmark categories. The table has four columns:

1. Number of order: Present only for convenience purposes.
2. Category name. Given in Engineering English, subject to localization.
3. Global Category ID.
4. Examples of landmarks: Here, examples of landmarks that can belong to this category are given. This is mostly for localization people as well as to those who create online helps for users, to clarify the meaning of the category name.

#	Category	ID	Examples of landmarks
1.	Accommodation	3000	Hotel, Camping site
2.	Business	6000	Bank, Factory, Office
3.	Communication	9000	Internet Access Point, Public Telephone, Wireless LAN Hot Spot
4.	Educational institute	12000	School, College
5.	Entertainment	15000	Amusement park, Cinema, Concert hall, Night club
6.	Food & Beverage	18000	Fast food, Restaurant, Café, Bar
7.	Geographical Area	21000	City, City center, Town
8.	Outdoor activities	24000	Camping site, Fishing place, Hunting, National park, Playground
9.	People	27000	My home, My friend’s home, Father’s summer cottage, Child’s school
10.	Public service	30000	Tourist information office, Government office, Library, Post office, Hospital, Police
11.	Religious places	33000	Church, Mosque
12.	Shopping	36000	Market place, Pharmacy, Shop, Shopping center
13.	Sightseeing	39000	Monument, Mountain top, Museum
14.	Sports	42000	Bowling, Golf course, Ice hockey hall, Stadium
15.	Transport	45000	Airport, Bus stop, Harbor, Railway station, Rest area

Values from 48000 to 63000 divisible by 3000 are reserved for possible additional seven categories.

The categories listed here are known as top-level categories and every one of these may be divided into sublevels for finer gradation. The whole value space is reserved for special purposes and future expansion.

Evaluate this resource

Please spare a moment to help us improve documentation quality and recognize the resources you find most valuable, by rating this resource.