TECHNICAL SPECIFICATION

ISO/TS 14827-4

First edition
2022-09

Intelligent transport systems — Data interfaces between centres for transport information and control systems —

Part 4: Data interfaces between centres for Intelligent transport systems (ITS) using XML (Profile B)

Systèmes de transport intelligents - Interface de données entre centres pour les systèmes de commande et d'information des transports —

Partie 4: Interfaces de données entre centres pour systèmes de transport intelligents (ITS) utilisant XML (Profil B)

Reference number
ISO/TS 14827-4:2022(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 204, *Intelligent transport systems,* in collaboration with the European Committee for Standardization (CEN) Technical Committee CEN/TC 278, *Intelligent transport systems,* in accordance with the Agreement on technical cooperation between ISO and CEN (Vienna Agreement).

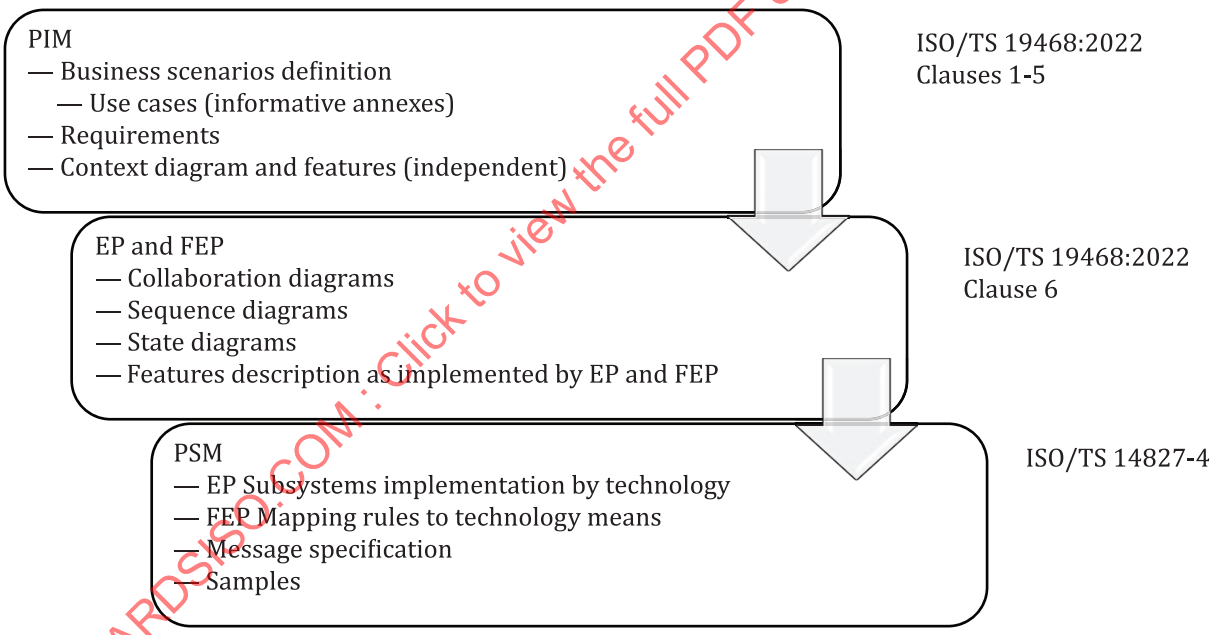
A list of all parts in the ISO 14827 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

Data exchange among centres is a baseline service for implementing intelligent transport system (ITS) services. For interoperability purposes, data delivery and collaborative ITS services need to be implemented according to certain specifications based on fully-described interfaces. The functional exchange profiles implementing push and pull exchange patterns aim to guarantee timely and reliable delivery of information, based on a defined level of service and user requirements. These depend on application level. A variety of options for implementing exchange are therefore described. These enable several interoperable exchange patterns with required features to fully satisfy user requirements: from Snapshot Pull/Push to Simple Push to Stateful Push, also considering a service request/service feedback collaborative ITS services business scenario, which allows interoperable exchange among any number of interconnected and collaborating elements to implement traffic management and traffic information services orchestrated among several ITS actors.

This document aims to define and describe the requirements on XML messages for implementing messages using XML Profile B. In particular, it is intended to be used in platform-specific implementations using simple object access protocol (SOAP) web services to enable DATEX II (EN 16157-1) XML coded messages for Snapshot Pull, Snapshot Push, Simple Push and Stateful Push. It is additionally relevant to collaborative ITS services (CISs) such as Simple CIS and Stateful CIS exchange patterns with relative functional exchange profiles as described by ISO/TS 19468. Figure 1 describes the relationship between exchange-related documents.



Key

PIM	platform-independent model
EP	exchange pattern
FEP	functional exchange profile
PSM	platform-specific model

Figure 1 — Relationship between exchange-related documents

The message structure defined in this document refers to the "basic exchange data model" and derived data dictionary defined in ISO/TS 19468:2022, Annex C, which is implemented in XML schema by the DATEX II methodology defined in EN 16157-1.

This document is not intended to conflict with existing standards on interfaces of data exchange among ITS centres.

Intelligent transport systems — Data interfaces between centres for transport information and control systems —

Part 4:
Data interfaces between centres for Intelligent transport systems (ITS) using XML (Profile B)

1 Scope

This document, based on ISO/TS 19468, specifies a platform-specific method for implementing data exchange among centres based on simple object access protocol (SOAP), supporting the EN 16157 series (DATEX II) for Push/Pull data delivery and service request/feedback collaborative intelligent transport system (ITS) services.

This document defines the message rules and procedures for communication between transport information and control systems using XML (Profile B).

This document clarifies how to package end-application messages and relevant data.

The payload data definition used in specific end-applications and the exact structure of the content payload delivered in the messages are beyond the scope of this document.

Rules and procedures for exchanging data-packets in lower communication layers are also out of the scope of this document. These functionalities can be implemented using generic protocols defined in the industry standards. However, this document does define how to use these protocols.

2 Normative references

The following documents are referred to in the text in such a way that some of or all their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/TS 19468, *Intelligent transport systems — Data interfaces between centres for transport information and control systems — Platform-independent model specifications for data exchange protocols for transport information and control systems*

RFC 2616, *Hypertext Transfer Protocol — HTTP/1.1*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/TS 19468 apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

4 Abbreviated terms

For the purposes of this document, the abbreviated terms given in ISO/TS 19468 and the following apply

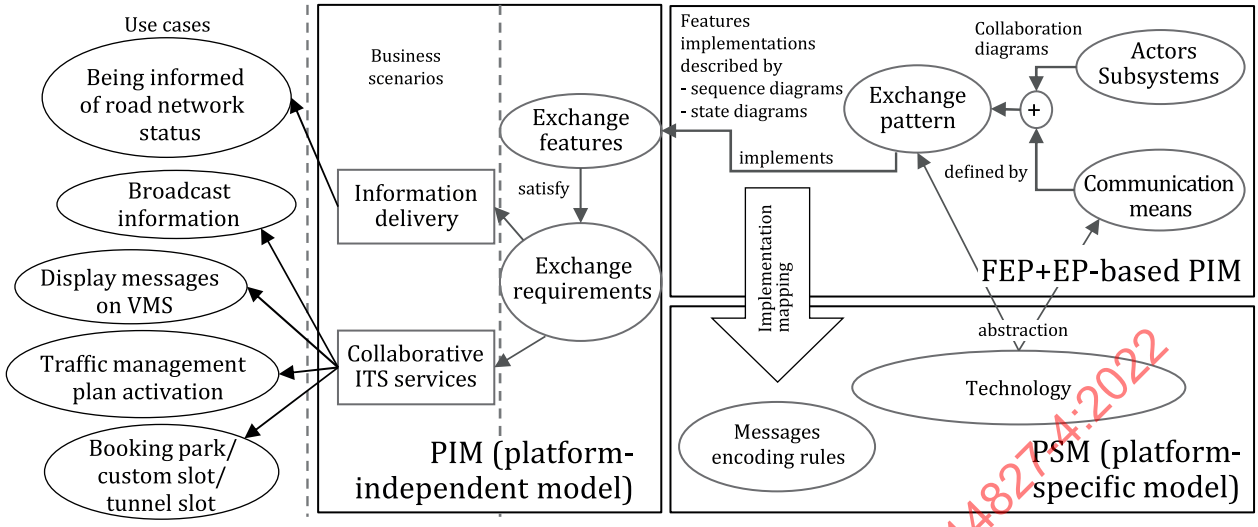
B2B	business to business
CIS	collaborative ITS services
EP	exchange pattern
FEP	functional exchange profile
IANA	Internet Assigned Numbers Authority
ITS	intelligent transport systems
PIM	platform-independent model
PSM	platform-specific model
SAML	security assertions markup language
UCS	universal multi-octet coded character set
UTF	UCS transfer format
VMS	variable message sign
WS	web service
XACML	extensible access control markup language
XMLDSG	XML digital signature
XMLENC	XML encryption

5 Conformance

There are no explicit conformance tests in this document. Conformance is achieved if the exchange data conform to the messaging rules of this document.

6 Exchange modelling framework

The model-driven approach defined in ISO/TS 19468 is resumed in Figure 2.



Key

PIM	platform-independent model
EP	exchange pattern
FEP	functional exchange profile
PSM	platform-specific model
VMS	variable message sign

Figure 2 — Model-driven approach from ISO/TS 19468

This document describes the mapping rules in order to implement specific platform Push and Pull and collaborative ITS services (CIS) FEP+EP-based PIM in SOAP web services technology PSMs. PIM-level descriptions of FEP+EP are detailed in ISO/TS 19468; the relevant clauses are referenced in this document.

6.1 Web services definition and options

Web services (WSs) provide standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks. A full definition for "web service" can be found in Reference [6].

Web service definitions offer several options. Table 1 shows the options chosen by the PSMs described in this document.

Table 1 — Web service options

Web service options	Decision
Discovery	Not dynamic: universal description discovery and integration(UDDI) is not used; the WSs are described at https://standards.iso.org/iso/ts/14827/-4/ed-1/en.
Security	The security set-up shall be decided by the supplier, should be negotiated with the clients, and is outside the scope of this document.
Encryption	This shall be agreed between the supplier and the client before starting the data exchange.

6.2 Web services PSM mapping of FEP+EP PIMs

PIM specification to implement FEP+EP PIM based on WS SOAP technology consists in mapping the abstract unified modeling language (UML) messages (invocation methods and data types) defined

at FEP+EP PIM level as UML collaboration and sequence diagrams in ISO/TS 19468 clauses to the corresponding SOAP web service definition language (WSDL) methods and data structure.

6.3 Security aspects related to WS implementation

As described in ISO/TS 19468, security aspects are considered in the exchange specification framework in the general context diagrams, under the aspects of communication features. Security features are described in ISO/TS 19468:2022, 5.7, while security requirements are defined in ISO/TS 19468:2022, Annex B.

As this document is based on exchange communication among centres based on XML, Profile B (e.g. under SOAP webservices and supporting DATEX II), security for such exchange framework based on WS may rely on the WS-security OASIS international standard specification which is widely used within several frameworks (e.g. the JAVA wsse4j, jax-ws, or the .NET WCF). Some aspects of WS-security are also introduced in ISO/TS 19468:2022, Annex D.

The WS-security framework is composed by profiles that define the interoperability rules to be used to attain specific features. Those profiles tackle the necessity to exchange an identification and authentication token (e.g. the username token profile), or how to use XML digital signature (XMLDSG) and XML encryption (XMLENC) to sign and encrypt parts of a SOAP message, respectively, or to support complex scenarios through the combined use of security assertions markup language (SAML) tokens and potentially extensible access control markup language (XACML) access control policies.

WS-Security needs further profiling before its use which is, usually, policy specific (e.g. which algorithm and cipher-suites to be used, canonicalization algorithms, token layouts, compliance with WS-BSP, etc.). This profiling effort which can be specifically based on regional or project specific rules, is outside the scope of the current specification.

7 Data Delivery FEP+EP PSM definition

7.1 Overview of Data Delivery PSM definition

The following clauses express requirements for implementation of data delivery business scenarios FEP+EP which are fully described at PIM level in ISO/TS 19468 by referencing their exchange agent and relative interfaces.

NOTE A system can be both a client and a supplier of another system simultaneously, defining multiple separated exchange contexts (e.g. set of exchange nodes defined to exchange information in a specific business scenario with a specified EP+FEP).

7.2 Profile B Snapshot Pull SOAP WS PSM definition

The Profile B Snapshot Pull SOAP WS implementation is defined according to the Snapshot Pull FEP + EP PIM description which is described in ISO/TS 19468:2022, Clause 6**,** where "Snapshot Pull" exchange system functional characteristics and features implementation are fully described.

In the Profile B Snapshot Pull SOAP WS exchange interface, the SOAP WSDL supplier method to implement pullSnapshotData shall be the WSDL method named "pullSnapshotData" which shall not require any input and shall return a "MessageContainer" information XML message data structure.

Messages are implemented by SOAP protocol adding SOAP envelope information. The corresponding WSDL file is given in Annex A of the present document. The XSD definition for Message Container is also described in Annex A.

7.3 Profile B Snapshot Push SOAP WS PSM definition

The Profile B Snapshot Push SOAP WS implementation is defined according to the Snapshot Push FEP + EP PIM description which is described in ISO/TS 19468:2022, Clause 7, where Snapshot Push exchange system functional characteristics and features are fully described.

In the Profile B Snapshot Push SOAP WS exchange interface, the SOAP WSDL client method to implement putSnapshotData shall be the WSDL method named "putSnapshotData" which shall accept as input a "MessageContainer" XML message data structure and shall return an "ExchangeInformation" XML message data structure.

Messages are implemented by SOAP protocol adding SOAP envelope information. The corresponding WSDL file is given in Annex A of the present document. XSD definitions for Message Container and Exchange Information are also described in Annex A.

7.4 Profile B Simple Push SOAP WS PSM definition

The Profile B Simple Push SOAP WS implementation is defined according to the Simple Push FEP + EP PIM description which is described in ISO/TS 19468:2022, Clause 8, where "Simple Push" exchange system functional characteristics and features are fully described.

In the Profile B Simple Push SOAP WS exchange interface:

— the SOAP WSDL client method to implement putSnapshotData shall be the WSDL method named "putSnapshotData" which shall accept as input a "MessageContainer" XML message data structure and shall return an "ExchangeInformation" XML message data structure;

— the SOAP WSDL client method to implement putData shall be the WSDL method named "putData" which shall accept as input a "MessageContainer" XML message data structure and shall return an "ExchangeInformation" XML message data structure;

— the SOAP WSDL client method to implement keepAlive shall be the WSDL method named "keepAlive" which shall accept as input an "ExchangeInformation" XML message data structure and shall return an "ExchangeInformation" XML message data structure;

Messages are implemented by SOAP protocol adding SOAP envelope information. The corresponding WSDL file is given in Annex A of the present document. XSD definitions for Message Container and Exchange Information are also described in Annex A.

7.5 Profile B Stateful Push SOAP WS PSM definition

The Profile B Stateful Push SOAP WS implementation is defined according to the Stateful Push FEP + EP PIM description which is described in ISO/TS 19468:2022, Clause 9, where Stateful Push exchange system functional characteristics and features are fully described.

In the Profile B Stateful Push SOAP WS exchange interface:

— the SOAP WSDL client method to implement openSession shall be the WSDL method named "openSession" which shall accept as input an "ExchangeInformation" XML message data structure and shall return an "ExchangeInformation" XML message data structure;

— the SOAP WSDL client method to implement closeSession shall be the WSDL method named "closeSession" which shall accept as input an "ExchangeInformation" XML message data structure and shall return an "ExchangeInformation" XML message data structure;

— the SOAP WSDL client method to implement putSnapshotData shall be the WSDL method named "putSnapshotData" which shall accept as input a "MessageContainer" XML message data structure and shall return an "ExchangeInformation" XML message data structure;

— the SOAP WSDL client method to implement putData shall be the WSDL method named "putData" which shall accept as input a "MessageContainer" XML message data structure and shall return an "ExchangeInformation" XML message data structure;

— the SOAP WSDL client method to implement keepAlive shall be the WSDL method named "keepAlive" which shall accept as input an "ExchangeInformation" XML message data structure and shall return an "ExchangeInformation" XML message data structure.

Messages are implemented by SOAP protocol adding SOAP envelope information. The corresponding WSDL file is given in Annex A of the present document. XSD definitions for Message Container and Exchange Information are fully described in Annex A.

8 Collaborative ITS Services FEP+EP PSM definition

8.1 Overview of Collaborative ITS Service (CIS) PSM definition

The following subclauses express requirements for the implementation of collaborative ITS service (CIS) business scenario FEP+EPs (which are fully described at PIM level in ISO/TS 19468) by referencing their exchange agent and relative interfaces.

NOTE A system can be simultaneously both a CIS requester and a CIS provider for another CIS requester, defining multiple separated CIS exchange contexts (e.g. set of exchange nodes defined to exchange information in a specific business scenario with a specified EP+FEP).

8.2 Profile B Simple CIS SOAP WS PSM definition

The Profile B Simple CIS SOAP WS implementation is defined according to the Simple CIS FEP + EP PIM description which is described in ISO/TS 19468:2022, Clause 10, where the exchange system functional characteristics and features are fully described.

In the Profile B Simple CIS SOAP WS exchange interface at the service provider side:

— the SOAP WSDL method to implement putCISServiceRequest shall be the WSDL method named "putCISServiceRequest" which shall accept as input a "MessageContainer" XML message data structure and shall return an "ExchangeInformation" XML message data structure.

In the Profile B Simple CIS SOAP WS exchange interface at the service requester side:

— the SOAP WSDL method to implement putCISServiceFeedback shall be the WSDL method named "putCISServiceRequest" which shall accept as input a "MessageContainer" XML message data structure and shall return an "ExchangeInformation" XML message data structure.

Messages are implemented by SOAP adding SOAP envelope information.

The corresponding WSDL file is given in Annex A of the present document. XSD definitions for Message Container and Exchange Information are also described in Annex A.

8.3 Stateful CIS

The Profile B Stateful CIS SOAP WS implementation is defined according to the Stateful CIS FEP + EP PIM description which is described in ISO/TS 19468:2022, Clause 11, where this exchange system functional characteristics and features are fully described.

In the Profile B Simple CIS SOAP WS exchange interface at the service provider side:

— the SOAP WSDL method to implement putCISServiceRequest shall be the WSDL method named "putCISServiceRequest" which shall accept as input a "MessageContainer" XML message data structure and shall return an "ExchangeInformation" XML message data structure;

— the SOAP WSDL client method to implement openSession shall be the WSDL method named "openSession" which shall accept as input an "ExchangeInformation" XML message data structure and shall return an "ExchangeInformation" XML message data structure;

— the SOAP WSDL client method to implement closeSession shall be the WSDL method named "closeSession" which shall accept as input an "ExchangeInformation" XML message data structure and shall return an "ExchangeInformation" XML message data structure;

— the SOAP WSDL client method to implement keepAlive shall be the WSDL method named "keepAlive" which shall accept as input an "ExchangeInformation" XML message data structure and shall return an "ExchangeInformation" XML message data structure.

In the Profile B Simple CIS SOAP WS exchange interface at the service requester side:

— the SOAP WSDL method to implement putCISServiceFeedback shall be the WSDL method named "putCISServiceRequest" which shall accept as input an "MessageContainer" XML message data structure and shall return an "ExchangeInformation" XML message data structure.

Messages are implemented by SOAP adding SOAP envelope information. The corresponding WSDL file is given in Annex A of the present document. XSD definitions for Message Container and Exchange Information are also described in Annex A.

Annex A
(informative)

Schemas and WSDL definitions

A.1 WSDL definitions

A.1.1 Profile B Snapshot Pull SOAP WSDL definition

The machine-readable file for this definition can be found at https://standards.iso.org/iso/ts/14827/-4/ed-1/en.

A.1.2 Profile B Snapshot Push SOAP WSDL definition

The machine-readable file for this definition can be found at https://standards.iso.org/iso/ts/14827/-4/ed-1/en.

A.1.3 Profile B Simple Push SOAP WSDL definition

The machine-readable file for this definition can be found at https://standards.iso.org/iso/ts/14827/-4/ed-1/en.

A.1.4 Profile B Stateful Push SOAP WSDL definition

The machine-readable file for this definition can be found at https://standards.iso.org/iso/ts/14827/-4/ed-1/en.

A.1.5 Profile B Simple CIS SOAP WSDL definition

The machine-readable file for this definition can be found at https://standards.iso.org/iso/ts/14827/-4/ed-1/en.

A.1.6 Stateful CIS WSDL definition

The machine-readable file for this definition can be found at https://standards.iso.org/iso/ts/14827/-4/ed-1/en.

A.1.7 Full Exchange WSDL definition

The machine-readable file for the extended union of all FEP+EP implementation described in this document can be found at https://standards.iso.org/iso/ts/14827/-4/ed-1/en.

A.2 XML schemas for exchange-related content

A.2.1 Message Container

The machine-readable file for Message Container can be found at https://standards.iso.org/iso/ts/14827/-4/ed-1/en.

A.2.2 Exchange Information XML schema definition

The machine-readable file for this definition can be found at https://standards.iso.org/iso/ts/14827/-4/ed-1/en.

A.2.3 Information Management XML schema definition

The machine-readable file for this definition can be found at https://standards.iso.org/iso/ts/14827/-4/ed-1/en.

A.2.4 CIS Information XML schema definition

The machine-readable file for this definition can be found at https://standards.iso.org/iso/ts/14827/-4/ed-1/en.

A.3 Snapshot Pull with simple http server profile definition

The machine-readable file for this definition can be found at https://standards.iso.org/iso/ts/14827/-4/ed-1/en.

Annex B
(informative)

"Snapshot Pull with simple http server" profile definition

B.1 Overall presentation

This profile is conformant to Snapshot Pull FEP+EP PIM. It is derived from the DATEX II version 2.0 Exchange Specification. It states all requirements for implementing Snapshot Pull by simple http server profile, i.e. implementing by pure http/get, within the following clauses which implement all PIM-specified features.

Snapshot Pull FEP+EP PIM defines the "pull message" received to be a "MessageContainer" according to basic exchange data model defined in ISO/TS 19468:2022, Annex C, which is implemented in the XSD schema defined in Annex A of this document.

Despite this rule, which always applies to Snapshot Pull implemented by WSDL, some implementations of "Snapshot Pull with simple http server" can consider implementation of the "PayloadPublication" as delivery only without wrapping it in a "MessageContainer". This is assumed to be implicitly defined by subscription agreement among client and suppliers.

This annex uses the term "Message" to refer either to a "Message Container" or a single "Payload Publication".

B.2 Describing payload and interfaces

The profile can handle different information to be delivered (such as different payloads), named information products by assigning a specific URL (potentially plus access credentials) per information product.

The information product itself is denoted by all but one of the path segments in the URL, while the 'filename' (i.e. the middle path segment) is "content.xml" per definition. This convention was introduced to allow the definition of related metadata for this information product in other files in the same directory.

Payload data for information products shall be denoted by a URL according to the following convention:
d2lcp_infop = "http://" host [":" port] infop_path "/content.xml" ["?" query]
where "info path" is a "path" component as specified in RFC 2396:1998, 3.3, but excluding the last path segment.

The end of this clause can sometimes sound awkward. It strives to maintain all the regulations of the URI RFC, thereby not constraining URLs for information products, but incorporating the need to have the final path element (the 'filename') being "content.xml" by convention.

To support authentication, the servers have to provide the credentials required to perform authentication for any particular information product.

Servers requiring authentication shall provide the required access credentials for BASIC authentication (i.e. user name and password) together with the URL for the information product.

B.3 Metadata for link management

Scenarios exist where the "If-Modified-Since" mechanism is not preferred to avoid redundant downloads. This will happen if the server provides information products by updating files on a standard, file-based WWW server (like Apache or Microsoft Internet Information Servers). In this option, the server would have two possible update regimes:

1) Periodical update of the information product's payload file, independent of any changes in the data.

2) Update of the information product's payload file on demand, when changes relevant to this product have occurred in the server's database.

With updating regime 1), the file-based WWW server would cyclically send the file content to the clients, as it will derive the last-modified value from the file modification times. So, the client would receive redundant downloads.

Implementation based upon standard WWW servers and files as information products should update information product payload files only when their traffic domain content has changed.

This means that the file can stay unmodified for some time after an update. This regime has one serious drawback: the client will not be able to determine whether the file remains unchanged because the (road) traffic situation is stable or because the backend server system itself (not the WWW server) is not operating properly. In fact, the response of the (intermediate) WWW system does not have the quality of an end-to-end acknowledgement.

This information is given in a small XML document that is periodically updated, even if the (potentially huge) content file is unchanged. The file is required to refer to an XML schema that contains an element called *MetaData* as root element with two required attributes of type *xsd:dateTime*, called *confirmationTime* and *confirmedTime*, with *confirmationTime* giving the time the acknowledgement was created and *confirmedTime* giving a value equal to the value the Last-Modified header field would have if the payload file (i.e. *content.xml*) had been requested.

The following XML Schema gives a valid example:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <!-- Our XML message consists of 1 element, the metadata record!-->
  <xsd:element name="MetaData" type="MetadataType"/>
  <xsd:complexType name="MetadataType">
    <xsd:attribute name="confirmationTime" type="xsd:dateTime" use="required"/>
    <xsd:attribute name="confirmedTime" type="xsd:dateTime" use="required"/>
  </xsd:complexType>
</xsd:schema>
```

The acknowledgement file shall be put in the information product's directory, besides the content.xml file containing the payload, with a filename of metadata.xml. A sample file for the schema above would be:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- D2LCP Demo File for Metadata -->
  <MetaData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="metadata.xsd"
  confirmationTime="2005-05-19T09:40:22+02:00"
  confirmedTime="2005-05-19T09:32:22+02:00"
/>
```

This leads to the following specifications:

Implementation based upon standard WWW servers and files as information products should provide a cyclically updated acknowledgement, accessible as:

— D2LCP_infop_ack = "http:" "//" host [":" port] infop_path "/metadata.xml" ["?" query]

— If an acknowledgement is provided, it shall be a well-formed XML document with an XML schema reference. The acknowledgement shall validate successfully against the referenced XML schema.

— The XML schema referenced by an acknowledgement shall contain a root element called “MetaData”. This element shall contain two attributes, one named “confirmationTime” and one named “confirmedTime”, both of type “xsd:dateTime”.

— If used, acknowledgements shall be updated cyclically, with a best effort update cycle, but not less than once every three minutes.

— An acknowledgement update shall indicate that the data server is operating properly at the time it is generated and that the content of that payload file (as last updated with modification time given in the “confirmedTime” attribute of the “MetaData”) element is currently still valid.

— The “confirmationTime” attribute of the “MetaData” element shall contain the current time when an acknowledgement is updated.

— The “confirmedTime” attribute of the “MetaData” element should contain the same value that a hypertext transfer protocol (HTTP) response to an authorized HTTP request issued at the time of writing the acknowledgement would contain in the “Last-Modified” header field.

— A server that is based upon standard WWW servers and files as information products shall indicate in the subscription negotiation process whether the acknowledgement option is supported.

— Clients should use the acknowledgement option, if provided, to determine whether payload download is required.

B.4 Use of HTTP

B.4.1 General

This profile exchange uses HTTP request (GET or POST) / Response dialogues to convey payload and status data from the supplier to the client. Note though that this profile supports post requests only for interoperability with the WS profile based on SOAP over HTTP exchange information potentially included in the body of an HTTP POST request, is not processed.

This clause stipulates how to use HTTP options for the suppliers and the clients of this profile.

B.4.2 Basic request/response pattern

a) Suppliers and Clients shall use the HTTP/1.1 protocol. Clients and suppliers shall fully conform to the HTTP/1.1 protocol specification in RFC 2616, as of June 1999.

This protocol is based upon a request/response pattern, where the request can take one of several possible forms, among them the "get" and "post" methods for retrieving data. The "get" and "post" differ essentially in how the parameters of a request can be conveyed to the supplier. While these parameters are conveyed as part of the URL in the HTTP "get" request, the "post" request allows specifying an “entity” (i.e. a message body) that contains these parameters, thus enabling less restricted parameters. "Post" requests were originally intended for server upload functionality.

As the specification foresees no complex request parameters, HTTP "get" requests are preferred. Since other exchange systems sometimes require HTTP "post" requests, they are also accepted. Nevertheless, it is not the intention to establish another exchange protocol layer on top of HTTP, and thus systems are not obliged to process the content of the body of an HTTP "post" request.

NOTE 1 Systems can decide not to process the body of HTTP post requests.

b) Clients shall use the HTTP "get" or "post" method of the HTTP request message to request data from the supplier.

The HTTP "get" or "post" request is served by the supplier by generating an HTTP response message. The "message" conveyed in this response is passed in the entity-body.

NOTE 2 For interoperability reasons (in particular with WS profile clients that require a SOAP wrapper around the XML payload) the profile does not stipulate the "Payload Publication" or the "Message Container" to be the root element of the XML content.

c) Suppliers shall use an HTTP response message to respond to requests. If applicable, one "message" is contained in the entity-body.

d) Suppliers shall not respond to HTTP request messages using the "get" or "post" methods by responding with 405 (Method Not Allowed) or 501 (Not Implemented) return codes.

All communication is initiated by the client. Any data flow from supplier to client can only happen as the supplier’s response to a client’s request. When requesting data, the client is not able to know whether the data they would receive would be exactly the same as that which they had received in response to their last request. This would lead to a serious amount of redundant network traffic, with a potential undesired impact on communication charges and supplier/client workload. HTTP supports avoiding this by letting the client specify the modification time of the last received update of a resource in an HTTP header field (If-Modified-Since). If no newer data is available, the response message will consist only of a header without an entity, stating that no new data is available. Clients are therefore recommended to set this header field in case they already hold reasonably recent information from the accessed URL.

e) Suppliers shall set the “Last-Modified” header field in HTTP response messages that provide payload data (response code 200) to the value that the information product behind the URL was last updated.

f) Clients may set the “If-Modified-Since” header field in all HTTP request messages if they already hold a consistent set of data from a particular URL in their database and the last modification time of that data is known from the “Last-Modified” header field of the HTTP header of the HTTP response message within which the payload data was received.

The semantics of the timestamps used within the "If-Modified-Since" header field are calculated in the server. Therefore, times generated by the client according to their own system clock may not be used here but have to be filled using the content of the "Last-Modified" header field of the most recently received HTTP response message. If clients connect to a resource for the first time or want to resynchronize, they simply do not set this header field.

g) When setting the “If-Modified-Since” header field, the client shall copy the value of the 'Last-Modified' header field received within the last successful HTTP response containing a message (response code 200) into this field.