DATEX II V2.0

EXCHANGE
PLATFORM SPECIFIC MODEL

Document version: 1.1

15 January 2010

European Commission

Directorate General for Transport and Energy

Prepared by :			
	Date	**Comment**	**Version**
DATEX Technical Committee	1 / 7 / 2009		1.0
DATEX Technical Committee	15 / 1 / 2010		1.1

Reviewed by :			
	Date	**Comment**	**Version**
DATEX Technical Committee	1 / 7 / 2009		1.0
DATEX Technical Committee	15 / 1 / 2010		1.1

Approved by :			
	Date	**Comment**	**Version**
DATEX Technical Committee	1 / 7 / 2009	Authorization for publication	1.0
DATEX Technical Committee	15 / 1 / 2010	Authorization for publication	1.1

TABLE OF CONTENTS

1. Introduction 5
1.1. Document objective 5
1.2. Reference documents 5
1.2.1. Datex II documents 5
1.2.2. RFC - documents produced by the Internet Society – IETF 5
1.2.3. W3C documents 5
1.2.4. Web Services interoperability organization (WSI) reference documents 5
1.3. Key words 5
1.4. Platform Independent model features and synthesis 5
1.4.1. Basic system overview 7
1.4.2. Subsystems 7
1.4.3. Use Cases 7
1.5. Web Services definition and options 8

2. DATEX II important features 10
2.1. General network topology 10
2.2. Multiple Client/Server types interoperability 11
2.3. Publications and profiles 12
2.4. Interoperability rules 12

3. Pull exchange 14
3.1. Introduction 14
3.2. Supplier system functional description 14
3.3. Web Services Supplier description 15
3.4. Client system functional description 15
3.5. Web Services Client description 15

4. Client Pull "Simple http Server" profile 17
4.1. Use of HTTP 17
4.1.1. Basic request / response pattern 17
4.1.2. Authentication 19
4.2. Describing payload and interfaces 20
4.3. Metadata for link management 21

5. Push exchanges 24
5.1. Introduction 24
5.2. Preconditions 24
5.3. Supplier system in the Push exchanges 25
5.3.1. Session management on the Supplier side 25
5.3.2. Subscription Service 25
5.3.3. Supplier building 26
5.3.4. PushService call 26
5.4. Client system in the Push Exchanges 26
5.4.1. Session Management on the Client side 26
5.4.2. Push operation 26
5.4.3. KeepAlive request 27

6. Appendix 29
6.1. Appendix A : RFC reference documents 29
6.2. Appendix B : W3C reference documents 29
6.3. Appendix C : Pull Service wsdl reference document 29
6.4. Appendix D : Push Service wsdl reference document 29
6.5. Appendix E : Client pull subscription service – General features (informative) 29

Introduction



1. Introduction

1.1. Document objective

Based on the DATEX II Platform Independent Model for data exchange (Exchange PIM), this document specifies the Exchange Platform Specific Model (Exchange PSM) for Web Services over http. The term *Web Services* is used generically here, since profiles for the Web Services protocol stack (WSDL & SOAP) as well as plain HTTP exchange options are provided.

1.2. Reference documents

1.2.1. Datex II documents

Reference in this document	DATEX II document	Document version	Date
[User guide]	DATEX II v2.0 User guide	1.1	15-01-2010
[Modelling methodology]	DATEX II v2.0 Modelling methodology	1.2	15-01-2010
[Data model]	DATEX II v2.0 Data model	RC2	15-01-2010
[Dictionary]	DATEX II v2.0 Data definitions	1.1	15-01-2010
[XML schema]	DATEX II v2.0 XML schema	RC2_2_0	15-01-2010
[Exchange PSM]	DATEX II v2.0 Exchange Platform Specific Model	1.1	15-01-2010
[Exchange PIM]	DATEX II v1.0 Exchange Platform Independent Model	1.01	08-02-2005
[Software developers Guide]	DATEX II v2.0 Software developers Guide	1.1	15-01-2010

1.2.2. RFC - documents produced by the Internet Society – IETF

The IETF (www.ietf.org) RFC reference documents are listed in Appendix A.

1.2.3. W3C documents

The W3C(www.w3c.org) reference documents describing Web Services and associated technologies are listed in Appendix B.

1.2.4. Web Services interoperability organization (WSI) reference documents

- [DOC 1] Basic Profile 1.1 – Second Edition – 2006-04-10

  http://www.ws-i.org/Profiles/BasicProfile-1.1-2006-04-10.html
- [DOC 2] Attachments profile 1.0 – Second Edition – 2006-04-20

  http://www.ws-i.org/Profiles/AttachmentsProfile-1.0-2006-04-20.html

1.3. Key words

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 : « Key words for use in RFCs to Indicate Requirement Levels ».

1.4. Platform Independent model features and synthesis

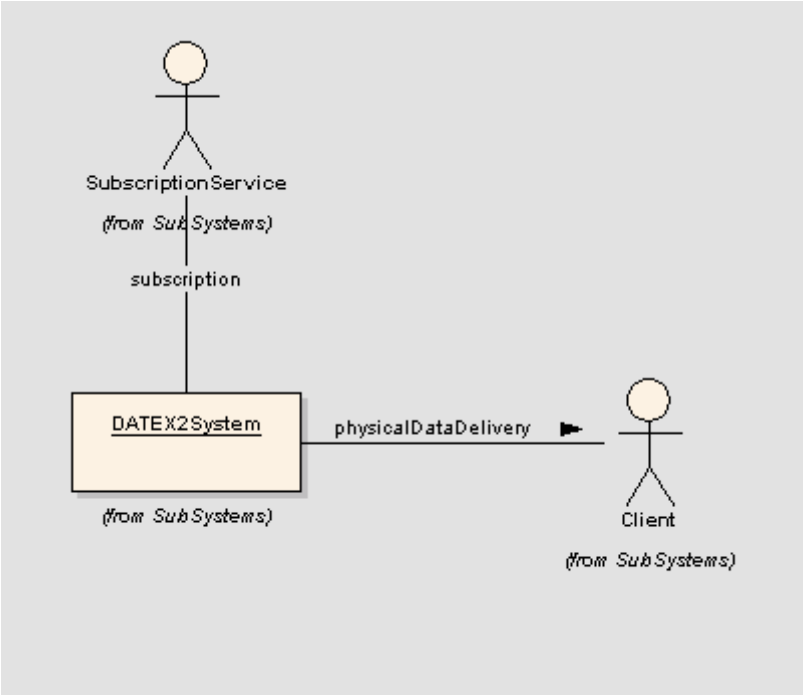
The Platform Independent Model (Exchange PIM) is described in reference document [Exchange PIM].

Those PIM main features that are useful for development are however summarised in the following paragraphs.

Remark : a few elements are not yet sufficiently described to be exchanged in an interoperable implementation:

- Catalogue Exchange,
- Filter Exchange

1.4.1. Basic system overview



The above figure provides a basic system overview, including the two main actors interacting with the system:

- Client :
  - receives traffic and travel related data from the DATEX II System
  - can be either another DATEX II System or a simple Client
  - is mandatory for the exchange process
- SubscriptionService
  - can be an integrated component or an external Service
  - can be done either by the Client (online) or by the ManagementAdministrator (off-line)
  - enters subscriptions
  - is mandatory for the exchange process

Remarks :

- a DATEX II system can interact with several DATEX II Clients,
- a DATEX II system which delivers data is also called « the Supplier » in this document.

1.4.2. Subsystems

A DATEX II system consists of different subsystems (Publisher, Delivery and Management). This document only deals with the Delivery subsystem.

The Delivery subsystem, which delivers data, is also called in this document « the Supplier ».

1.4.3. Use Cases

The PSM document details the « HandleExchange » use case. When necessary, references are made to other use cases which are described in the Exchange PIM document ([Exchange PIM]).

1.5. Web Services definition and options

Web Services provide standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks.

Definition (W3C - Web Service architecture - http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/) : « A Web Service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web Service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialisation in conjunction with other Web-related standards. »

Web Services definitions offer several options. The following table shows the options chosen by DATEX II.

Web Service Options	Decision
Discovery	Not dynamic : UDDI is not used, the Web Services are described in this DATEX II document which is the reference for development
Security	The security set-up has to be decided by the Supplier, should be negotiated with the Clients, and is outside the scope of this specification
Encryption	This has to be agreed between the Supplier and the Client, before starting data exchange

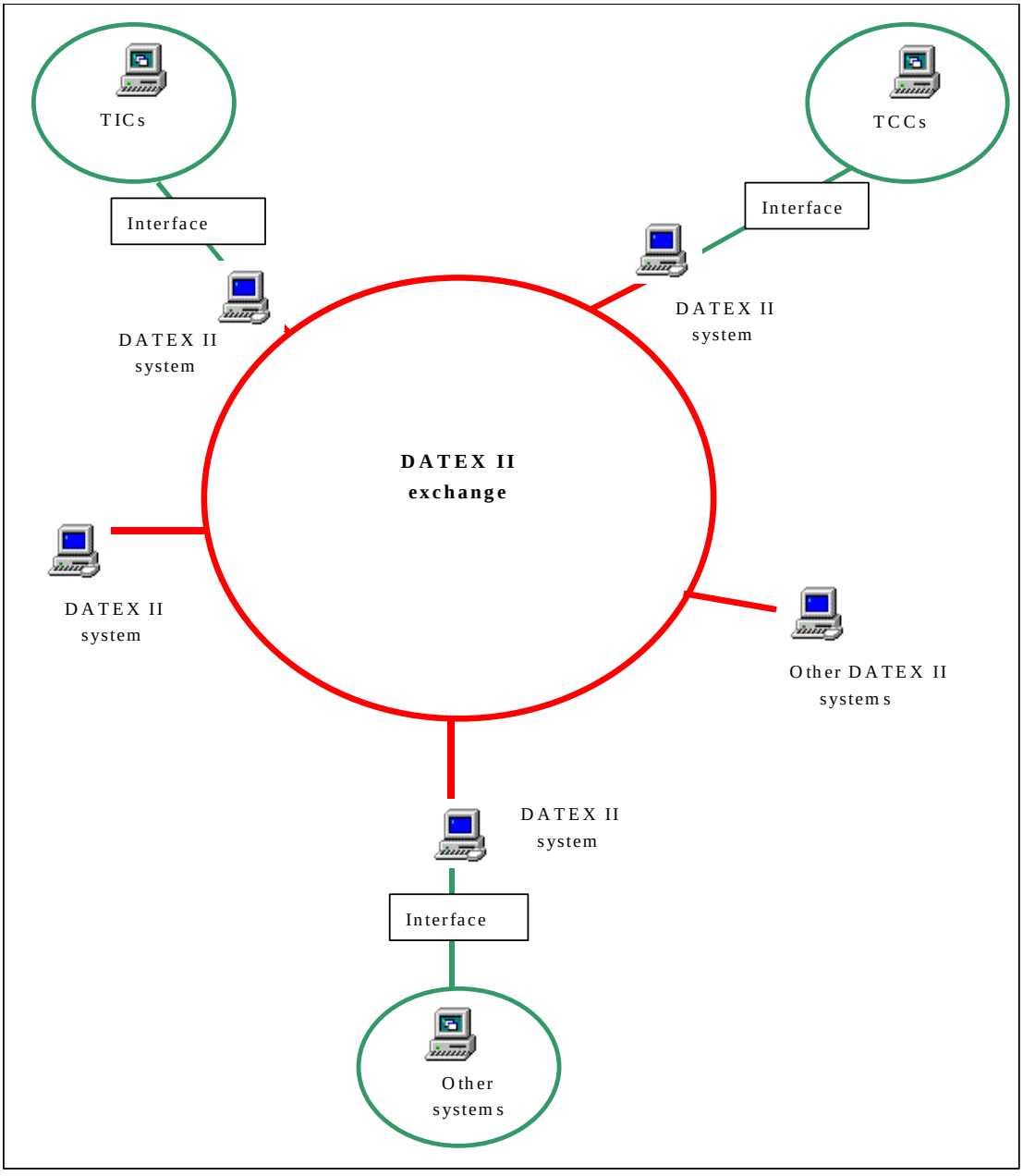
DATEX II important features



2. DATEX II important features

2.1. General network topology

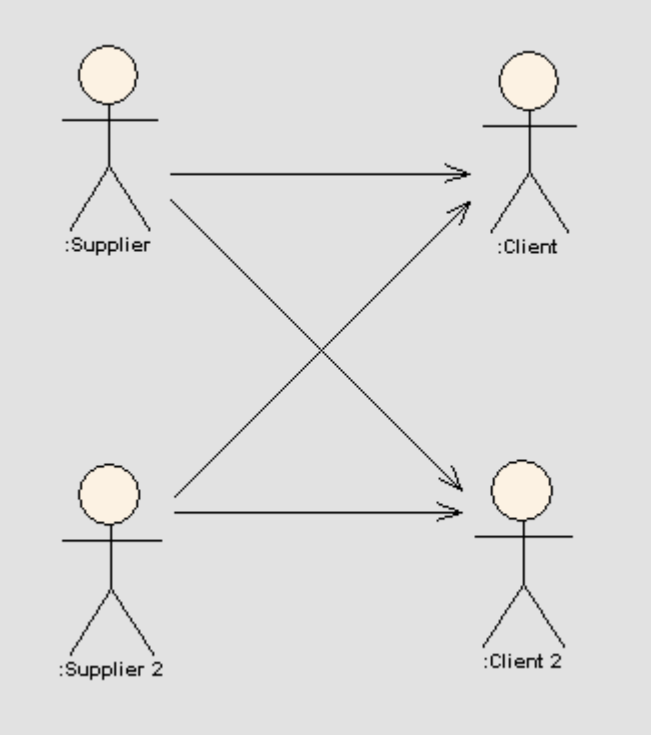
The general network topology is a network which can be summarised by the following schema. On each site, the local architecture depends on needs and implementations.



2.2. Multiple Client/Server types interoperability

In DATEX II architecture, all the exchanges conform to this specification:

- a Client must be able to receive DATEX II data from any Supplier,
- a Supplier must be able to deliver DATEX II data to any Client.



2.3. Publications and profiles

In the exchange PIM document (Ref.: [Exchange PIM]), different operating modes are described and, depending on used platforms, different exchange options may be chosen.

In data PIM (Ref.: [Data model] [Dictionary]), different publication types are described.

A DATEX II system is composed of different publications which can be delivered with different operating modes and different exchange options. Each DATEX II system builder chooses to implement the subset of (publications, operating modes, options) he needs. This subset is called a « DATEX II profile ».

The need is to have profiles and options to allow DATEX II users to choose their implementation in order to provide more or less functionalities / facilities as required and not to force them to implement all features.

Example of a DATEX II profile

Publications	Operating mode		
	Pull	Push On occurrence	Push Periodic
Situation	**X**		
TrafficView	**X**		
MeasuredData			
ElaboratedData	**X**		
PredefinedLocation	**X**		
MeasurementTable			
…			

Example of another DATEX II profile

Publications	Operating mode		
	Pull	Push On occurrence	Push Periodic
Situation	**X**	**X**	**X**
TrafficView	**X**	**X**	
MeasuredData	**X**	**X**	
ElaboratedData	**X**	**X**	
PredefinedLocation	**X**		
MeasurementTable	**X**		
…			

2.4. Interoperability rules

Rule 1: when two Suppliers support the same publication type with the same operating mode, a Client software SHOULD be able to receive data from both of them.

Rule 2: when a Supplier and a Client don't use the same operating mode for a publication and exchange is not possible, the usage of the operating mode "Pull" MUST be implemented on both sides.

Rule 3: when one of the DATEX II partner (Supplier or Client) doesn't support on-line subscriptions, the off-line mode MUST be used.

Pull exchange



3. Pull exchange

3.1. Introduction

This chapter describes the detailed specification of the DATEX II Pull Exchange mechanism.

In this kind of exchange, the Client is the Requester and the Supplier is the Provider: data delivery is initiated by the Client and data is returned by the Supplier as an online response.

The update method (Ref. [Data model]) for each publication is “snapshot”: all information, available for the requesting Client‘s subscriptions, is delivered.

On the Supplier side, different sets of data can be available via different addresses (URL). In this case, we consider in this document that data are available via different services because one service MUST have one unique URL.

The encryption and compression methods depend on the Supplier system capabilities.

General features concerning subscription service, can be found in appendix E.

3.2. Supplier system functional description

This paragraph is valid for all DATEX II profiles.

Exchange parameters:

IN parameters :

There are no IN parameters.

OUT parameters

SOAP envelope containing one XML node “D2LogicalModel” conforming to DATEX II XML schema (Ref.: [XML schema]).

Function

All available data, after Client authentication, are provided, for all active subscriptions (of this Client or group of Clients), when they exist.

Two technologies can be used for the Pull Supplier system:

- One, using Web Services and the Web Service Description Language (WSDL), called “Web Services profile”,
- Another, using only an http Server without WSDL usage, called “Simple http server profile”.

In order to keep interoperability, despite these two technologies, the solution is to respond messages with a SOAP envelope, even if the Supplier is a simple implementation based on standard HTTP server supplying data directly from files.

The “Web Services profile” is described in the following paragraph.

The “Simple http server profile” is described in the following chapter.

3.3. Web Services Supplier description

In this profile, the Supplier uses Web Services classes, automatically generated, to set data and encode the message.

DATEX II reference package contains a reference WSDL description of the Supplier Web Service.

The corresponding WSDL file is given in Appendix C.

This WSDL file imports the DATEX II reference XML schema.

Each DATEX II Supplier web service SHOULD be built with the reference WSDL document. If a Supplier modifies the schema which is imported in the WSDL, he MUST manage and keep interoperability with his DATEX II Clients.

This Pull service WSDL has been built with the following choices :

- define one unique reference WSDL as stable as possible,
- define one WSDL for Pull DATEX II exchange valid for all DATEX II exchanged data,
- reference to Web Services Interoperability (WS-I) Organization (Re [DOC 1]),
- binding SOAP : style=document, use=literal, following WS-I recommendations.

Remark : SOAPAction http header field (Rule coming from the WSI Basic profile (Re. [DOC 1])) : « The HTTP request MESSAGE MUST contain a SOAPAction HTTP header field with a quoted value equal to the value of the soapAction attribute of soapbind:operation ».

3.4. Client system functional description

The Client gets data from the Supplier. For that, like for the Supplier system, two technologies can be used for the Pull Client system:

- One, using Web Services and the Web Service Description Language (WSDL),
- Another, using directly http requests.

In order to keep interoperability, despite these two technologies, solutions have been found:

- at the Supplier level (the solution is to respond messages with a SOAP envelope, even if the Supplier is only an http Server),
- at the Client level: the Client which doesn't use Web Services ignores the SOAP envelope.

The "Web Services profile" is described in the following paragraph.

The Client using directly http requests is described in the following chapter.

3.5. Web Services Client description

In this profile, the Client uses automatically generated Web Services classes, to get data and decode the message.

"DATEX II v2.0 reference set" contains a reference WSDL description of the Supplier Web Service which must be used by the Client to build access to the services.

The corresponding WSDL file is given in Appendix C.

This WSDL file imports the DATEX II reference XML schema.

Each DATEX II Client web service SHOULD be built with the reference WSDL document. If a Client modifies the schema which is imported in the WSDL, he MUST manage and keep interoperability with his DATEX II Suppliers.

Client Pull “Simple http server” profile



4. Client Pull “Simple http Server” profile

4.1. Use of HTTP

This profile exchange uses HTTP Request (GET or POST) / Response dialogs 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 Web services profile based on SOAP over HTTP exchange Information potentially included in the body of an HTTP POST request, is not processed.

This section stipulates how to use HTTP options for the Suppliers and the Clients of this profile.

4.1.1. Basic request / response pattern

[C.1] Suppliers and Clients SHALL use the HTTP/1.1 protocol. Clients and Suppliers SHALL fully comply with the HTTP/1.1 protocol specification in RFC 2616, as of June 1999.

This protocol is essentially 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: systems MAY decide not to process the body of HTTP POST requests!

[C.2] 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 payload data – if any – conveyed in this response is passed in the entity-body. The payload data has to conform to the DATEX II data serialisation rules. The exact structure of the DATEX II content payload is beyond the scope of this specification! It should be noted thought that for interoperability reasons (in particular with Web Services profile Clients that require a SOAP wrapper around the XML payload) the profile does not stipulate the DATEX II content payload to be the root element of the XML content. It only requires the existence of ***exactly one*** DATEX II content payload instance to be available as a subtree of the whole XML content tree.

[C.3] Suppliers SHALL use an HTTP RESPONSE message to respond to requests. If applicable, payload data according to DATEX II payload encoding specifications is contained in the entity-body. The DATEX II payload element MAY be embedded into other XML elements (‘wrapper’) in the contained XML document, but this XML document MUST contain exactly one DATEX II payload element.

[C.4] 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 he would receive would be exactly the same as the one he had received in response to his last request. This would lead to a serious amount of redundant network traffic, with potential undesired impact on communication charges and Supplier/Client work load. 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.

[C.5] Suppliers MUST 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.

[C.6] Clients SHOULD 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.

It must be understood that 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 his own system clock may not be used here but must 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 resynchronise, they simply don't set this header field.

[C.7] 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 payload (response code 200) message into this field.

[C.8] Suppliers SHOULD provide XML coded DATEX II payload as "text/xml" media type. Suppliers SHOULD state the used character set via the "charset" parameter; Suppliers SHOULD use the UTF-8 character set, i.e. the "Content-Type" response-header field SHOULD state "text/xml; charset=utf-8".

Character sets, media types, etc. are a vast area that is notoriously underestimated as a source of potential interoperability problems. This clause aims at recommending the most widely used set of options, namely the use of the UTF-8 character set for XML payload. The "text/xml" is preferred to "application/xml" following a recommendation in RFC 3023 (*"If an XML document -- that is, the unprocessed, source XML document -- is readable by casual users, text/xml is preferable to application/xml."*). Although in principle the profile is solely devoted to B2B communication, readability of the exchange payload has often proved to be beneficial for testing and educational purposes.

It should be noted that omitting the "charset" attribute in HTTP/1.1 for "text/*" type content implies the use of "ISO-8859-1" which is different from the UTF family (UTF-8, UTF-16) that are the minimum requirement for XML processors, and can thus be seen as the de-facto standard for XML. Consequently, sending typical XML payload like this:

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<d2lm:situationPublication xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="D2LogicalModel D2LogicalModel_TiS_48.3_ext3.xsd">

[...]

via HTTP/1.1 actually requires the use of the "charset" attribute:

HTTP/1.x 200 OK

[...]

Content-Type: text/xml;charset=UTF-8

[...]

If "text/xml" without parameter would be sent, HTTP/1.1 would be violated. (Quote from RFC 2616: "When no explicit charset parameter is provided by the sender, media subtypes of the "text" type are defined to have a default charset value of "ISO-8859-1" when received via HTTP. Data in character sets other than "ISO-8859-1" or its subsets MUST be labelled with an appropriate charset value.")

[C.9] Clients MUST accept "identity" content-coding; Clients SHOULD (and if they do, prefer to) accept "gzip" content-coding; Clients MAY accept other "content-coding" values registered by the Internet Assigned Numbers Authority (IANA) in their content-coding registry[1] as long as they also accept "identity" and "gzip" content-coding.

[1] See www.iana.org

[C.10] When including an "Accept-Encoding" request-header field in an HTTP REQUEST message, the Client MUST NOT exclude acceptance of "identity" content-coding.

[C.11] Suppliers MUST provide "identity" content-coding of the payload; Suppliers SHOULD provide "gzip" content-coding of the payload; Suppliers MAY provide other "content-coding" values registered by the Internet Assigned Numbers Authority (IANA) in their content-coding registry as long as they also provide "identity" and "gzip" content-coding.

This set of clauses essentially ensures that a confused situation where the Supplier is not able to provide payload in a content-coding that the Client understands can not exist, as all Suppliers are enforced to support "identity" (which means unmodified text/xml content) content-coding, and Clients are enforced to understand this content-coding.

Furthermore, these clauses include a policy that recommends the use of compression and ensures that compression is always interoperable because it requires all Clients/Suppliers that do support compression to support "gzip" at least as an option. This means that:

- All Client/Supplier interaction will work at least with "identity"
- Clients/Suppliers supporting compression will always be able to agree on "gzip"
- Clients can request preferred other compression ("deflate" or "compress"), and Suppliers will respond accordingly if they support these content-codings.

Implementers should be aware that non-transparent web caches may perform media type transformations on behalf of their Clients. Thus Client running over such a cache might notice that compressed response content is automatically decompressed. However this is only problematic if there is a low bandwidth connection between the Client and the cache, such as a dial up access point. If a service has a significant number of such users then the addition of the no-transform directive to the Cache-Control header field of the generated responses should be considered. For more details on the use of the Cache-Control field, please consult Section 14.9 of RFC 2616.

[C.12] Servers MAY use the 'no-transform' directive in the 'Cache-Control' header field to avoid non-transparent caches from sending non-compressed content.

4.1.2. Authentication

Authentication is supported, i.e. only users with explicit permission are allowed to download payload data. The required access credentials have to be provided to the Client as part of the outcome of the DATEX II subscription creation with the content Supplier, which is here an offline process. The specifications make use of the simplest and most widely spread authentication scheme for HTTP, i.e. BASIC authentication.

Note:

This scheme is in itself not seen as sufficiently strong for commercial strength business and safety relevant application, as the password is not encrypted during transmission. Applications that fit into this description will either have to use other DATEX II profiles or they will need to establish a sufficiently secure transport layer below DATEX II.

In essence, the Client receives a user name and password together with a URL which identifies a specific publication from the server. During access, the Client then builds a single string from this ("<username>:<password>") and encodes it according to base64 encoding rules. The result is put into the Authorization header field of the HTTP REQUEST message.

[C.13] Clients SHOULD fill access credentials they MAY have received during the subscription negotiation process into the 'Authorization' header field of the HTTP REQUEST message.

[C.14] Server providing access credentials (user name & password) during the subscription negotiation phase MAY respond with response code 401 (Unauthorized) to HTTP REQUESTS that do not contain valid access credentials in the 'Authorization' header field

The regulations in this and the previous section are a clarification on how to use standard features of HTTP/1.1 according to RFC2616. The following sections contain additional regulations that go beyond 'pure' HTTP. Anyway, the regulations presented so far have to be seen as clarifications on top of RFC2616. They are compliant with the standard and have to be used in conjunction with the standard itself. This principle holds especially for the handling of HTTP return codes. The following

clause summarises the main return codes as used in connections and refers to the standard for the handling of further codes.

[C.15] Servers SHALL produce and Clients SHALL process the following return codes:
- **200 (OK), in responses carrying payload,**
- **304 (Not Modified), if no payload is send because of the specification in the 'If-Modified-Since' header,**
- **503 (Service Unavailable), if an active HTTP server is disconnected from the content feed,**
- **404 (Not Found), if a file based HTTP server does not have a proper payload document stored in the place associated to the URL.**

Servers SHOULD produce and Clients SHOULD process the following return codes:
- **401 (Unauthorised), if authentication is required but not presented in the request, or if invalid authentication is presented in the request,**
- **403 (Forbidden), if the requested operation is not successful for any other reason.**

Servers & Client SHALL apply an RFC2616 compliant regime for producing / handling all other return code.

4.2. Describing payload and interfaces

Server side Client filtering is not supported! It should also be noted that although such a feature could in principle be incorporated, it would require substantial processing power inside the server and also Client specific information products, which would increase the implementation complexity on the server side substantially. Users aiming at applications based on server side Client filtering are thus advised to consider using the Web Services profile instead!

Nevertheless, it was decided that servers should (statically and equally for all Clients) split their publications into different 'information products' (e.g.. roadwork information, incidents, x-urgent messages). Information products SHOULD be used to split the different DATEX II publications, but can go further in adding particular filters on attributes, locations, etc. In particular, this allows consideration to be given to different amounts of data and typical update cycles / latency requirements for different types of data. An x-urgent message information product will probably hardly contain more than one or two situation elements at a time but Clients may want to poll this information product at high frequency. Roadwork information – on the other hand – will probably only change once per day but may contain hundreds of situation records. Even if the profile provides various mechanisms to reduce redundant downloads, some Clients may decide that updating this data every 2 hours is sufficient.

The profile handles information products by assigning a specific URL (potentially plus access credentials) per information product. The information product itself is denoted by all but one 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 meta-data for this information product in other files in the same directory.

[C.16] 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 "infop_path" is a "path" component as specified in section 3.3 of [RFC 2396], but excluding the last path segment.

The end of this clause may sound awkward. It strives at maintaining all the regulations of the URI RFC, thus 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.

[C.17] Server requiring authentication MUST provide the required access credentials for BASIC authentication (i.e. user name & password) together with the URL for the Information Product.

4.3. Metadata for link management

Scenarios exist where the If-Modified-Since mechanism introduced earlier is not preferred to avoid redundant downloads. In particular, 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 Server). In this option, the server would have two possible update regimes:

- 1 : Periodical update of the information product's payload file independent from any changes in the data.
- 2 : Update 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 he will derive the Last-Modified value from the file modification times. So the Client would receive redundant downloads!

[C.18] 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!

Now this will mean that the file may stay around 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 is one of the major problems of simple FTP based DATEX links, and certainly has to be avoided for DATEX II! Therefore, an additional mechanism is required for servers that build upon file based WWW servers that give additional explicit meta-information as 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:

[C.19] 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]*

[C.20] If an acknowledgement is provided, it SHALL be a well-formed XML document with a XML Schema reference. The acknowledgement SHALL validate successfully against the referenced XML Schema.

[C.21] 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".

[C.22] If used, acknowledgements SHALL be updated cyclically, with best effort update cycle, but not less than once every three minutes.

[C.23] 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.

[C.24] The "confirmationTime" attribute of the "MetaData" element SHALL contain the current time when an acknowledgement is updated.

[C.25] The "confirmedTime" attribute of the "MetaData" element SHALL contain the same value that a HTTP RESPONSE to an authorised HTTP REQUEST issued at the time of writing the acknowledgement would contain in the "Last-Modified" header field.

[C.26] 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.

[C.27] Clients SHOULD use the acknowledgement option – if provided – to determine whether payload download is required.

Push exchanges



5. Push exchanges

5.1. Introduction

This chapter describes the detailed specification of the DATEX II Push Exchanges mechanisms.

In these kinds of exchange :

- the subscriptions can be made online or offline,
- in the Web Service architecture, the Supplier is here the Requester and the Client is the Provider : data delivery is initiated by the Supplier and data is delivered by the Supplier as a parameter of operations offered by Client Services, in online modes.

There are two push operating modes (Ref : [Exchange PIM]) : the difference between these two operating modes is the way the Publisher Subsystem elaborates data to be delivered :

- Operating Mode 1 – Publisher Push on occurrence : the payload is elaborated by the Publisher every time data is changed,
- Operating Mode 2 – Publisher Push periodic: the payload is elaborated by the Publisher on a cyclic time basis.

In both modes, data delivery is the same, the invoked Client Service is the same, only the triggering event (data change or periodic trigger), managed by the Publisher, is the difference.

The update methods (Ref. [Data model]), managed by the Publisher, can depend on operating modes and publications. They can be :

- singleElementUpdate : If an atomic part of the data has been changed, this atomic part, and only this atomic part, will be exchanged,
- allElementUpdate : If an atomic part has been changed the data complex associated with the atomic part will be exchanged,
- snapshot: A snapshot contains all information that is available for a subscription.

The Publisher mechanisms are not described within this document which only deals with exchange mechanisms.

5.2. Preconditions

A subscription is a necessary pre-condition for data exchange. Without a subscription, data delivery is not possible. To enable a Client to make a subscription, either online or offline, contact information is required.

In either case, offline through a contact person or online automated, the encryption and compression methods, if used, have to be clearly defined.

In online mode, the following information can be exchanged:

- Mandatory : address of the DATEX II Supplier System to which the Clients want to connect
- OPTIONAL : username
- OPTIONAL : password

Remarks :

In the case that username and password are required, the DATEX II Supplier System will need to hold this information in a partner list for authentication,

The encryption and compression methods for data exchange depend on the Client system possibilities.

5.3. Supplier system in the Push exchanges

5.3.1. Session management on the Supplier side

In both Push modes (“on occurrence” and “periodic”), the session management is done, on the Supplier side, by the request of a keep-alive Client operation to test the link between the Supplier and the Client.

If no data has been available for transmission to the Client for a stipulated period and thus none has been sent successfully to the Client, the Supplier requests periodically a KeepAlive Client operation.

When a data transfer or a keepAlive request doesn’t succeed, the link with this Client is considered, by the Supplier, as not available. Then, the Supplier stops data sending to this Client and requests periodically the keepAlive Client operation, until it succeeds. After link recovery, data sending can start again.

Remark : the rate of the keepAlive request is predefined and must be given by the Supplier to the Client during the interchange agreement.

5.3.2. Subscription Service

For a subscription, the operation called exchangeSubscriptionRequest (Exchange) is used : this can be done off-line.

Exchange meta data

In this case, the exchange meta data content (Ref. [Exchange PIM] : class diagram of exchange meta data) is :

Exchange

- SupplierIdentification
- ClientIdentification
- RequestType = « subscription »

Subscription

- operatingMode = operating mode 1 or operating mode 2
- updateMethod = “singleElementUpdate”/”allElementUpdate”/”snapshot”
- subscriptionStartTime
- subscriptionStopTime (OPTIONAL)
- deliveryInterval (minutes) : in operating mode 2, only

Target(s)

- Address
- Protocol

Subscription methods

A Client MAY have several subscriptions. There are two ways to establish a subscription : offline or online.

- Offline mode : it is not described by this specification ; readers can use the online descriptions, as a help.
- Online mode valid for several exchanges

  In this mode, before a set of exchanges, the Client uses a Service offered by the Supplier to define a subscription, valid between a subscription start and stop times.

  A Client can define several subscriptions (for instance, to define different delivery intervals depending on data classes).

  Then, during these subscription periods, the Client will receive the corresponding data.

  This subscription service which can be implemented in an Internet variant by an http server, using specific Supplier html pages and specific mechanisms, is not described by this specification.

5.3.3. Supplier building

Each DATEX II Supplier MUST be built in accordance with the DATEX II Push WSDL description.

5.3.4. PushService call

The Supplier calls the DATEX II Client Push Service.

5.4. Client system in the Push Exchanges

5.4.1. Session Management on the Client side

In both Push modes ("on occurrence" and "periodic"), if the Client doesn't receive data nor has no keep-alive request from a Supplier, within a predefined time, the Client MUST understand that the link with this Supplier has failed.

Client actions when link recovers:

- when the link recovers, one procedure is the use by the Client of a Supplier Service to get active data and synchronize, this procedure is called "snapshot recovery" and is, in fact, a request to the "Client Pull" Supplier service, already described in this document,
- other procedures are possible : on a bilateral agreement, the Supplier and the Client could implement other data recovery solutions depending on their specific needs. For instance, "full history recovery" that would allow to recover the complete versions history of the updated elements,
- a combination of different procedures is also possible : for instance, the Client can try to receive all updates by a "full history recovery" and if not successful, as the supplier system provides "not implemented" or other error condition, the Client can try the "snapshot recovery" operation.

Remark 1 : the predefined time for the declaration of the link failure must be greater than the rate of the keep-alive request from this Supplier (which has been given during the interchange agreement).

Remark 2 : when the link recovers, the Client has to ensure old data management.

5.4.2. Push operation

Functional description

This Client Service offers one operation, called PutDATEXIIData.

IN parameters : D2LogicalModel conforming to DATEX II XML schema (Ref. : [XML schema]).

It is filled by the Supplier with available data.

OUT parameters : D2LogicalModel conforming to DATEX II XML schema.

It is an acknowledgement.

The attribute "response" of "element "exchange" is filled with the value "acknowledge".

In this case, the attribute ClientIdentification MUST be filled.

FUNCTION : Available data are provided to this Client, totally or uniquely for all his active subscriptions, when they exist.

Physical description

The Push Client operation is described by its WSDL description.

Each DATEX II Client service MUST be built with this WSDL document.

The corresponding WSDL file is given in Appendix D.

The WSDL file has been built with the following choices :

- define one unique reference WSDL as stable as possible,
- define one WSDL for Push DATEX II exchange valid for all DATEX II exchanged data,
- reference to Web Services Interoperability (WS-I) Organization (Re [DOC 1]),

- binding SOAP : style=document, use=literal, following WS-I recommendations.

5.4.3. KeepAlive request

The keep-alive request is done by invoking the putDATEXIIData operation and giving, as input message, a d2LogicalModel message filled, in the “Exchange” element, with the boolean attribute “keepAlive” set to “true.

Appendix



6. Appendix

6.1. Appendix A : RFC reference documents

- IETF RFC 2045 - *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies* – November 1996
- http://www.ietf.org/rfc/rfc2045.txt
- IETF RFC 2119 - *Key words for use in RFCs to Indicate Requirement Levels* – http://www.ietf.org/rfc/rfc2119.txt
- IETF RFC 2279 – *UTF-8, a transformation format of ISO 10646* –
- http://www.ietf.org/rfc/rfc2279.txt
- IETF RFC 2396 – *Uniform Resource Identifiers (URI) : Generic Syntax* – http://www.ietf.org/rfc/rfc2396.txt
- IETF RFC 2616 - *Hypertext Transfer Protocol – HTTP/1.1* –
- http://www.ietf.org/rfc/rfc2616.txt
- *http/1.1 Specification Errata* – last modified 2004-10-27 –
- http://skrb.org/ietf/http_errata.html
- *IETF RFC 2617 – HTTP Authentication : Basic and Digest Access Authentication – June 1999*
- http://www.ietf.org/rfc/rfc2617.txt

6.2. Appendix B : W3C reference documents

The W3C(www.w3c.org) documents describing Web Services and associated technologies are listed in the following paragraphs (the list has been updated on September, 22nd – 2006).

XML documents

- Extensible Markup Language (XML) 1.0 (Third Edition), W3C Recommendation, http://www.w3.org/TR/2004/REC-xml-20040204, 4 February 2004

SOAP documents

- Simple Object Access Protocol (SOAP) 1.1 - W3C Note 08 May 2000, http://www.w3.org/TR/2000/NOTE-SOAP-20000508

WSDL documents

- Web Services Description Language (WSDL) 1.1 – W3C Note 15 March 2001, http://www.w3.org/TR/wsdl

6.3. Appendix C : Pull Service wsdl reference document

See separate wsdl file.

6.4. Appendix D : Push Service wsdl reference document

See separate wsdl file.

6.5. Appendix E : Client pull subscription service – General features (informative)

A subscription is a necessary pre-condition for data exchange. Without a subscription, data delivery is not possible.

To enable a Client to make a subscription, either online or offline, contact information is required. In either case, offline through a contact person or online automatically, the following information is necessary:

- Mandatory : address of the DATEX II Supplier System to which the Clients want to connect

- OPTIONAL : username, password
- OPTIONAL : encryption and compression methods

In the case that username and password are required, the DATEX II Supplier System will need to hold this information in a partner list for authentication,

In this operating mode, there is no need to build a specific session management to monitor the network link between a Supplier and a Client, since the Client itself initiates the connection and thus monitors the network link.

For a subscription, the operation called exchangeSubscriptionRequest (Exchange) is used.

Exchange meta data

In this case, the exchange meta data content (Ref. [Exchange PIM] : class diagram of exchange meta data) is :

Exchange

- SupplierIdentification
- ClientIdentification
- RequestType = « subscription »

Subscription

- operatingMode = operating mode 3
- updateMethod = snapshot
- subscriptionStartTime
- subscriptionStopTime (OPTIONAL)

Target(s)

- Address
- Protocol

Remark 1 : In addition, catalogues and filters MAY be used within the physical exchange, they are not sufficiently specified thought to be currently used in this document version.

Remark 2 : The DATEX II system MAY :

- Approve a subscription providing a subscriptionReference
- Deny the subscription providing the deny reason

Subscription methods

There are two ways to establish a subscription : offline or online.

A Client MAY have several subscriptions.

A subscription MAY be created or modified before each exchange.

Offline mode

The off-line mode is up to the Supplier and not described by this specification; readers can use the online descriptions below as a guideline.

Online mode

In this mode, before a set of exchanges, the Client uses a Service offered by the Supplier to define a subscription, valid between a subscription start time and a subscription stop time.

A Client can define several subscriptions (for instance, to define different delivery intervals depending on data classes). Then, during this period, when the Client requests data, in the pull mode, he will receive data corresponding to it(s) subscription(s).

This Service, which can be implemented, in an Internet variant, by an http server using specific Supplier html pages and specific mechanisms, is not described by this specification.

The « Client Pull » exchange, could be used in a simple or more sophisticated way :

- in the minimal interoperable configuration, the Client does not provide the subscriptionReference(s) of it(s) subscriptions; he receives each time all information available for it(s) subscription (s),
- in the case where the subscription reference is provided by the Client, the Supplier will deliver information related to this specific subscription.