Regulation F: EDI communication

Appendix report 1:

Syntax and structure of EDI messages

April 2007

Rev. 1

In case of any discrepancy between the Danish text and the English translation, the Danish text shall prevail

memorandum

Table of contents

1. EDIFACT syntax and structure

This appendix report describes the technical syntax and structure of the EDIFACT standard. The contents are based on the 'Common rules and recommendations' of ebIX.

1.1 EDIFACT structure and terms[1]

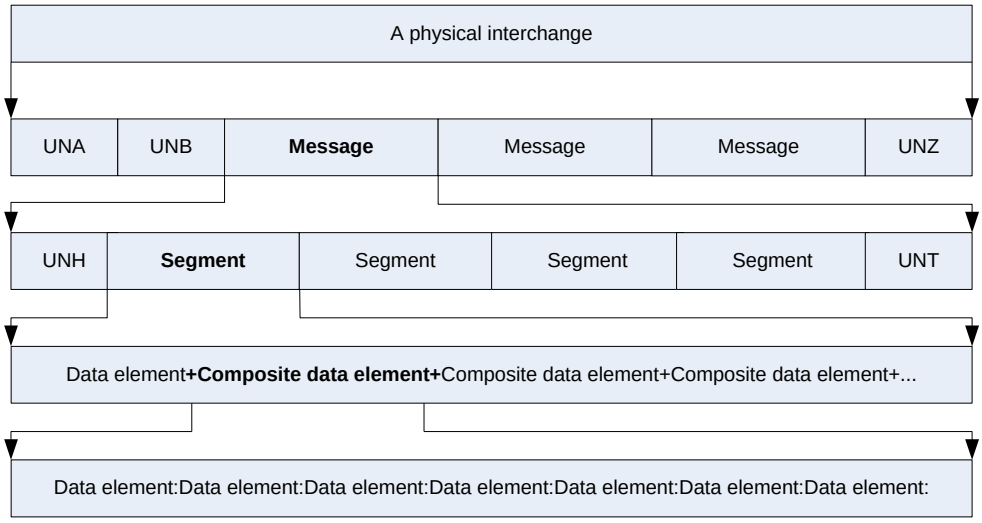
EDIFACT is a United Nations standard (UN/CEFACT) used for interchanging data in structured form. This means that EDIFACT is subject to specific syntax rules, which apply at all times and must always be complied with.

EDIFACT contains the following terms:

- An **interchange** consists of all information in a transmission. A physical interchange can comprise one or more messages. An interchange has one sender and one receiver.
- A **message** is for instance:
  - Metered data
  - Notifications and schedules

In Denmark, the general rule is that only one message per interchange is allowed.

- A **segment** is a group of related data elements/fields. It could, for instance, be an address. In the segment the address is specified by indication of name, road, postcode, town, country, etc.
- A **composite data element** comprises two or more data elements.
- A **data element/field** is the individual information transmitted in a segment. It could for instance be an indication of time or a number of kilowatt hours.



The figure above illustrates the interconnected use of the four terms. UNA, UNB and UNZ are segments used for marking the beginning and end of an

[1] The section is based on UN/CEFACT – http://www.gefeg.com/jswg/

interchange. UNH and UNT are segments used for marking the beginning and end of a message.

The following characters are used for marking separation, end, etc. in connection with data elements and segments:

+	Marks separation between data elements and between segments.
'	Marks that a segment is finished.
:	Marks separation between the individual data fields in composite data elements; this means more fields describing the same.

1.2 EDIFACT syntax

Basic rules for messages used in the electricity market:

1.2.1 Character set and structure

- Send EDIFACT interchanges as one long string without changing lines.
- Always send interchanges in the ISO 8859-1 character set, also known as UNOC.
- Completely fill in yymmdd and hhmm in the UNB. Hence, it is insufficient to send, for instance, 645 to indicate the hour. To be correct, it should be 0645.
- Always write message names and segment names in UPPERCASE when they are transmitted.
- EDIFACT version 3 is used in Denmark.

1.2.2 Use of reserved characters

The UNA segment describes the separators (reserved characters) used in the transmitted EDIFACT. The standard is: UNA:+.? ’

If one of the reserved characters (+, :, ') is used, it should be preceded by a question mark (?). Thus, for the addition of two numbers (eg 10 + 20), write 10?+20. If a question mark is needed, write two question marks (??). The first question mark restores the normal meaning of the next one. If a release character is used, it is not included in the determination of the field length. If a field allows 20 alphanumeric characters and 2 release characters are needed, the field may be up to 22 characters long.

1.2.3 Avoid superfluous signs

- Delete trailing plus signs (+) throughout the segment.
- Delete trailing colons in each individual data. Do not delete leading plus signs (+) or colons (:).
- Data elements (fields) can be excluded by omitting the field ‘values’, eg 24500+45++25600, where the field value between 45 and 25600 has been omitted. In the event of a composite data element, it can for instance be 3650:3575::3400, where the field value between 3575 and 3400 has been omitted.

1.3 Undescribed segments and elements in the message implementation guides

EDIFACT messages must meet all rules set out in the regulation. If a player receives EDIFACT messages that contain additional segments and elements not described in the relevant guidelines, these must be ignored. It must therefore not give rise to an error message report to the sender as long as the message complies with the basic structure of the EDIFACT message in question (eg UTILMD or UTILTS).

1.4 Interchange control reference, message ID and interchange version

The interchange control reference is found in the S004.0020 element of the UNB segment, which must be unique over time for the sender, as defined in S002. If this is not the case, the last received message must be rejected automatically. Message ID is found in the C002.1204 element of the BGM segment and is used for the unique identification of a message. Message ID must be unique over time for each player. Interchange version is not used in EDIFACT.

1.5 Fixed segments

Below follows a description of selected service segments (UN segments). UNA and UNB are used for the interchange header and can be regarded as the envelope of the interchange. UNB is used by the receiving EDI system for identification of sender and receiver.

1.5.1 UNA – Syntax segment

UNA Service String advice
Function: For specification of notation in EDIFACT interchange
Classification: Conditional, (1)
Comments: The examples below are default. Used in Denmark.
UNA UNA:+.? '

Ref.	Name	Cl.	Form.	Description
	COMPOSITE DATA ELEMENT, SEPARATOR	M	an1	":" (colon)
	DATA ELEMENT, SEPARATOR	M	an1	"+" (plus sign)
	DECIMAL NOTATION	M	an1	"." (full stop)
	RELEASE CHARACTER	M	an1	"?" (question mark)
	RESERVED FUTURE USE	M	an1	" " (blank)
	SEGMENT TERMINATOR	M	an1	"'" (apostrophe)

1.5.2 UNB – EDIFACT envelope for all documents except CONTRL

UNB Interchange header

Function: For identification and addressing of the interchange, specification of a request for acknowledgement as well as test indication.

Classification: Mandatory (M1).

Comments: The use of the UNB segment must be agreed in the interchange contract.

Example: UNB+UNOC:3+5790000395620:14+5790000432752:14+070120:1606+31929++DK-TIS-MET++1+DK

Ref.	Name	Cl.	Form.	Description
S001	SYNTAX IDENTIFIER	M		
0001	Syntax identifier	M	a4	**Code:** **UNOC** (ISO 8859-1)
0002	Syntax version number	M	n1	**Code:** **3** Version 3 of EDIFACT syntax must be used if syntax identifier is "UNOC"
S002	INTERCHANGE SENDER	M		
0004	Sender identification	M	an..35	GLN or EIC number.
0007	Partner identification code qualifier	R	an..4	**Code:** 14 GLN (Global Location Number) 305 EIC (European Identification Code)
0008	Address for reverse routing	O	an..14	Only if bilaterally agreed.
S003	INTERCHANGE RECIPIENT	M		
0010	Recipient identification	M	an..35	GLN or EIC number.
0007	Partner identification code qualifier	R	an..4	**Code:** 14 GLN (Global Location Number) 305 EIC (European Identification Code)
0014	Routing address	O	an..14	Only if bilaterally agreed.
S004	DATE AND TIME OF PREPARATION	M		
0017	Date	M	n6	Date of interchange creation (YYMMDD)

0019	Time	M	n4	Local time of day when interchange was created (YYMMDD)
0020	INTERCHANGE CONTROL REFERENCE	M	an..14	Unique reference assigned to the interchange. Must match data element 0020 in UNZ. Must be unique over time for sender, as defined in element S002.0004, otherwise the interchange must be rejected.
S005	RECIPIENTS REFERENCE, PASSWORD	X		
0022	Recipient's reference/ password	X	an..14	
0025	Recipient's reference/ password qualifier	X	an..2	
0026	APPLICATION REFERENCE	O	an..14	BPI is used, see national rules or ”ebIX Business information models”.
0029	PROCESSING PRIORITY CODE	X	a1	
0031	ACKNOWLEDGEMENT REQUEST	O	n1	**Code:** 1 Acknowledgement is sent blank No acknowledgement
0032	COMMUNICATION AGREEMENT ID	O	an..35	**Code:** DK Danish communication provisions.
0035	TEST INDICATOR	D	n1	**Code:** 1 Is used if the interchange is a test. Must be bilaterally agreed. blank The message is in production.

1.5.3 UNH EDIFACT document header

The UNH segment has been described as Danish practice for element 0068 differs from ebIX.

UNH Message header

Function: UNH is sent once per document in an interchange, identifying the message, version number and code of the interchange cooperation.

Classification: Mandatory (M1).

Comments:

Example: UNH+1+UTILMD:D:02B:UN:E5DK02+DK-BT-001-002'

Ref.	Name	Cl.	Form.	Description
0062	MESSAGE REFERENCE NUMBER	M	an..14	*A unique reference assigned to the message.*
S009	MESSAGE IDENTIFIER	M		
0065	Message type identifier	M	an..6	Identifies the message used (eg UTILMD)
0052	Message type version number	M	an..3	
0054	Message type release number	M	an..3	EDIFACT message version
0051	Controlling agency	M	an..2	The agency controlling the message.
0057	Association assigned code	C	an..6	MIG version number. See MIG. Mandatory.
0068	COMMON ACCESS REFERENCE	C	an..35	Business Combined id
S010	STATUS OF THE TRANSFER	C		
0070	Sequence message transfer number	M	n..2	
0073	First/last seq. mess. transfer. Indicator.	C	a1	

1.5.4 UNZ

UNZ Interchange trailer

Function: For termination of interchange and specification of total number of messages in interchange

Classification: Mandatory (M1).

Comments: Data element 0020 must be identical to 0020 in UNB

Example: UNZ+1+358765298'

Ref.	Name	Cl.	Form.	Description
0036	INTERCHANGE CONTROL COUNT	M	n..6	Number of messages in interchange
0020	INTERCHANGE CONTROL REFERENCE	M	an..14	Must match data element 0020 in UNB segment

2. XML syntax and structure

This section provides support to players in the Danish electricity market that are going to use XML messages. As XML is not a standard, the electricity market uses a methodology made up of a number of basic components from which the messages are created. The individual messages are not described in the same way as EDIFACT messages.

2.1 What is XML?

XML[2] is a language used for describing data in a structured and general way. Data are described in XML by means of pure text organised as a tree structure. The tree structure is specified by using *tags*, which are a data markup as shown in the following example:

```
<person>
  <name>Per</navn>
  <postcode>5000</post>
</person>
```

When data are marked up, it enables the structured interchange between two or more partners. In XML everyone is free to define an arbitrary markup language. In most contexts it is therefore necessary to define a standardised markup language for the data interchange.

Various bodies and associations in the electricity market have agreed on a common standard for XML interchanges, which standardises the data markup language. This ensures that the data content of the interchanges is interpreted in the same way by sender and receiver.

The XML interchange in itself is pure text stored in a text file. It must therefore also be decided what method should be used for transmitting the interchange to the receiver. This can, for instance, be done through a web service or by e-mail.

2.1.1 XML in the electricity market

The electricity market uses a standardised markup in XML, based on ETSO's XML standards. In connection with interchanges related to the handling of notifications and schedules, it is particularly ETSO's ESS standard that provides the basis for the standardisation.

ESS has been introduced as a pan-European standard for the interchange of data between transmission system operators and between transmission system operators and balance responsible parties (BRPs). ESS comprises a number of standard messages and core components, which enable the design of new messages by making use of the framework included in the core components.

ESS does not resemble earlier XML messages used in the Danish electricity market, such as Eltras XML-DELFOR, which was a direct translation of EDIFACT into XML.

[2] eXtensible Markup Language

2.2 XML syntax

XML (eXtensible Markup Language) is composed of *tags* that surround and, in that manner, mark the type of data. A postcode could, for instance, be marked by a start tag <postcode> and an end tag </postcode> (see example below).

```
<postcode>5000</postcode>
```

In the example above, the value 5000 now makes sense as it is defined as a postcode. Alternatively, the value could symbolise anything.

Start tag is defined by: <*tag name*> End tag is defined by: </*tag name*>

Together, the start tag, end tag and the content between the tags define an element. Hence, in the example above, the "postcode" tag thus defines the element.

A tag can also contain attributes, which provide additional information and, therefore, value in relation to the content of the tag besides the name. Attributes are declared within the start-tag by name, as shown in the example below.

```
<postcode countrycode="DK">5000</postcode>
```

In the example above, for instance, the *countrycode="DK"* provides information that the content of the tag (5000) is a Danish postcode.

Attributes are defined in the start-tag by: attribute name="attribute value"

The alternative could be a following tag for the country code (see example below).

```
<postcode>5000</postcode>
<countrycode>DK</countrycode>
```

2.2.1 Well-formness

A well-formed XML document must contain W3C's XML specification in such a way that the document has exactly one root element. All elements must be surrounded by a start-tag and an end-tag. If the element is empty, tags must be used as described in section 2.2.2. See an example of the use of tags below.

```
<root element>
  <name>Per</name>
  <postcode>5000</post>
</root element>
```

The example above is well-formed according to the specification described.

2.2.2 Use of attributes and empty tags

A tag can contain any number of different attributes.

An attribute is defined by: <tag name attribute name=”attribute value”>

A tag need not necessarily contain data, in which case it is an empty tag. An empty tag can be defined by either a start and end tag without any content in between or as shown in the example below.

```
<postcode/>
```

An empty tag is defined by either: <*tag name*></*tag name*>
or by: <*tag name*/>

2.2.3 DTD compared with XML schema

Document Type Definition (DTD) and XML Schema Definition (XSD) are two methods to describe content, structure and type definitions for an XML document. DTD used to be in fairly widespread use in connection with the definition of the content of the XML message directly in the XML document. Today, however, the trend is moving towards separate schema files (XSD) for defining the XML document content.
XSD makes it easy to:

- describe the content, if applicable
- validate the correctness of data
- define data facets (restrictions on data content)
- define data patterns (data formats)
- convert data between different data types (by use of redefine)

Moreover, the development and validation of XSD are supported by various XML editors, parsers and other tools.

An XML document is validated in relation to the data definition (either DTD or XSD).

2.2.4 Use of encoding and characters

The standardised XML standard uses the UTF-8 encoding form, which is backward compatible with ASCII. UTF-8 is a 2-bytes-per-character encoding, which is a compressed version of Unicode. Encoding is specified at the beginning of the XML file as follows:

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

The XML syntax contains a number of reserved characters, which are not allowed to appear in the content of elements or attributes.

Reserved character	Alternative notation
<	Can be written as “<”
>	Can be written as “>”
&	Can be written as “&”
"	Can be written as “"”
'	Can be written as “'”

The XML notation is case sensitive (does not apply to the content of an element).

The element *<Identification></ Identification>* is not the same as *<identification></ identification>* because of the difference between upper- and lowercase I/i.

2.3 Interchange control reference, message ID and interchange version

An XML message has no interchange control reference, but a unique message ID, which is obtained by combining the elements ”MessageIdentification” and “MessageVersion”. The version number is consecutive, starting with 1. Is placed in the element ”MessageVersion”.

2.4 Core components

Core components are defined by UN/CEFACT (United Nations Centre for Trade facilitation and Electronic Business). Core components are syntax neutral and technology neutral in their basic form and can be used for data modelling. CCTS (Core Component Technical Specification) ensures a high degree of reuse across messages and improves interoperability between IT systems.

The purpose of these components is to make them so generic that they become suitable for a variety of messages. The components define the role concept, date/time stamps and message types, to mention a few examples. The components are used several times in connection with different message types.

These core components should be used wherever possible, and candidates for core components should also be considered in conjunction with the development of new message types.

2.5 Use and structure of XML schemas

XML schemas are used for validating the XML message when the message is sent and received. If the XML message does not comply with the schema definitions, the message is not valid and should therefore be rejected. This is done with an acknowledgement document.

XML schemas must be structured in such a way that it offers strong data binding internally in the schema and the possibility of reuse. Strong data binding means that an element of a schema must have a relation to the residual schema elements. If elements can be reused in other schemas, however, these must generally be extracted to separate schemas (generalisation).

Each XML schema must be structured to ensure that child notes are embedded in parent notes, which means that only one main element is defined in each schema (see example below).

```
<element name="Address" type="AddressType"/>

<complexType name="AddressType">
  <xsequence>
    <element name="Streetname" type="string"/>
    <element name="Number" type="integer"/>
    <element name="Postcode" type="PostcodeType"/>
  </sequence>
  <attribute name="id" type="string"/>
  <attribute name="version" type="string"/>
</xs:complexType>

<simpleType name="PostcodeType">
  <restriction base="integer">
    <pattern value="\d{4}"/>
  </restriction>
</simpleType>
```

2.6 Versioning for XML schema

XML schemas developed for communication between Energinet.dk and external partners use a target name space with the following structure:

"**http://www.energinet.dk/schemas/***"+<subnamespace>+"**/**"+<document>+"/**v**"+<version>.*

The example below shows what the naming of a namespace can look like for the XML schema concerning BRPs' exchange of notifications:

http://www.energinet.dk/schemas/BalResp/XML/MarketScheduleDocument/v2

The XML schemas are versioned by use of the file name. The name is built on the basis of the XML schema root element combined with the version number. The combination of the two parts is separated by - (hyphen) as shown below:

*<rootelementname>+"-"+<version>***+**"**.xsd**"

The example below shows the naming of the first version of an XML schema where the root element is named "MarketScheduleDocument":

MarketScheduleDocument-1.xsd

Besides the version number, the "version" attribute in the schema element must also reflect the release number separated by a full stop. The following example is used for version 2, release 4:

version="2.4"

A change in the version number will be due to structural changes in the schema. Structural changes can be the addition or removal of elements, name

changes of elements or attributes or changes in the element structure. A change in the release number can be due to minor changes. Minor changes can be the addition of optional elements, changes in the rules for attribute content (other than restrictions) and the like. This means that elements are not removed from the structure.

It will therefore be possible to use different releases of a version of an XML schema without causing conflict between the releases (in other words, they must be backward compatible). An earlier version of an XML schema will conflict with a newer version, however. Energinet.dk takes steps to ensure that it is capable of handling the latest version released and its predecessor. Thus, two different versions are always available, comprising two or more releases within each version.

2.7 References

- **ETSO TASK FORCE 14 EDI**
  www.edi.etso-net.org/
- **ETSO Scheduling System**
  www.edi.etso-net.org/schedulev3r0/documentation/ess-guide-v3r0.pdf
- **The ETSO CORE COMPONENTS v.4.0R0**
  www.edi.etso-net.org/core/ecc-v4r0.pdf