SECTION

0 Introduction

Version 4.0

Edig@s

EASEE-gas/Edig@s Workgroup

Document version 2

TABLE OF CONTENTS

Figures

1 GENERAL

The following paragraphs contain a very high level description of EDI, what it is, what are its advantages and how it works.

What is EDI?

Electronic **D**ata **I**nterchange or simply EDI is the exchange of structured business information in either direction between partners in an organized, standardized manner, using modern communication methods. In this definition the word "standardised" plays a key role as it rules out proprietary solutions resulting in conversion chaos for those dealing with more than one partner.

Its Advantages

The major advantage of EDI is that it permits the rapid transmission and comprehension of bulk data from computer to computer, making the control of the commercial activities far more efficient and cost effective. In addition, the ability to receive information through EDI offers an opportunity for the data to be fed directly into the in-house system application, thereby saving time on data preparation and data re-entry.

The elimination of data re-entry also caters for the elimination of associated errors, the direct transfer of data from one application to another safeguards the integrity of the data.

As the data transmitted is no longer meant to be read by an operator, but is destined to be integrated into the in-house application, this data can be reduced to strictly dynamic data in coded form whenever possible. This caters for compact data transmissions.

Those are only a few examples of the multiple advantages linked to EDI. Properly implemented EDI offers a wide range of opportunities to make business more efficient.

How it works

The following chart illustrates the different tools required to run an EDI process.

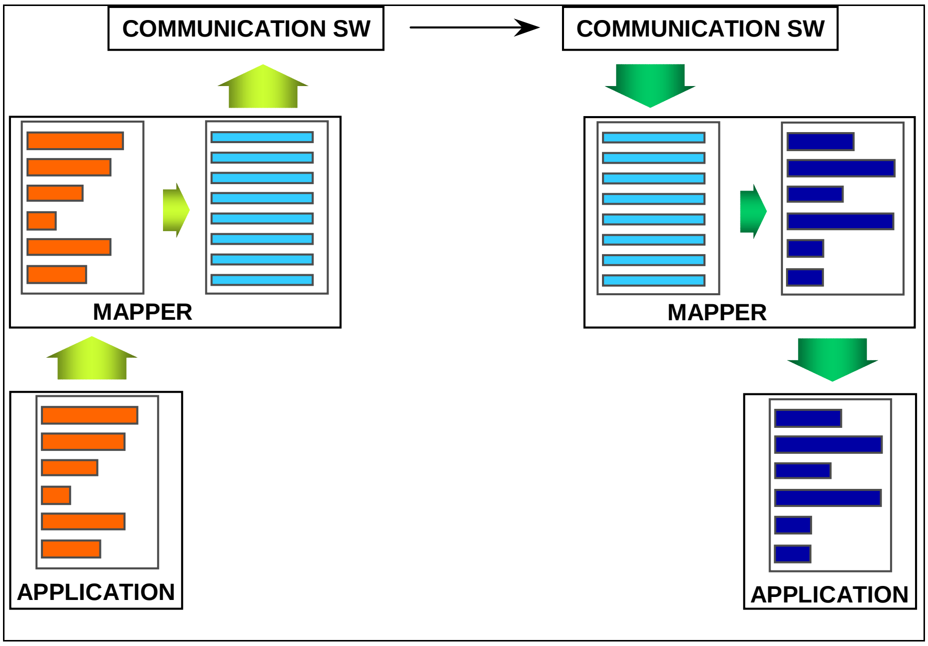


FIGURE 1: EDI OPERATION

The mapper will convert outgoing in-house format files supplied by the application into EDIFACT standard formats. It will add qualifier values where required, compact the data and wrap the data in service segments required for the correct transmission of the data.

For incoming data the mapper will perform the data validation (syntax compliance) and convert the EDIFACT standard format into an in-house format that can be fed into the application.

EDIFACT Syntax

To achieve the goals of EDI a universal electronic language is required. As with any language, the basic rules of grammar and vocabulary have been defined. Syntax is the same as grammar and aims at two goals: the first is to maintain a standard structure to the data by defining the data element relationships; the second is to assemble exchanges in such manner that the computer can recognize and manage the interchange. The syntax rules are independent from machine, media, system and application and can be used in any trading environment with any communication protocol.

More specific information on the EDIFACT syntax can be found on the following website: http://www.gefeg.com/jswg/

UN/EDIFACT Messages The basic tools for the transmission of information between parties is the use of the UN/EDIFACT message set which provides a standardised set of document structures that can be adapted for use in a specific information exchange situation. Such adaptations are known as "subsets" A number of these message structures are used by *Edig@s*, who has defined specific subsets for its use. The formal structures may be found on the following website: http://www.unece.org/trade/untdid/welcome.htm.

2 OBJECTIVES

(Market evolution) The previous version of the *Edig@s* implementation guide (version 3.2) has a number of significant changes to some of its principal messages (NOMINT, NOMRES, GASDAT, ALOCAT, CONTRL, APERAK). The modifications to these messages were triggered by market evolutions that needed changes to the messages in question. In addition Version 3.2 was never fully completed as it only covered the messages that were impacted by the introduction of the CBP.

Version 3.2 initialised a modification to the structure and presentation of the implementation guides. This new version completes these modifications and covers the complete range of *Edig@s* messages. It has modeled all its messages to bring the implementation guide in line with current practices.

(User Feedback) Version 3.2, after a year of use in production, has highlighted a certain number of issues due to varying interpretations of the implementation guides which have been resolved.

The EDIFACT libraries used were over 10 years old and many improvements in semantic understanding have been introduced. This revision includes all these improvements.

(Technical evolution) By introducing information models of all the *Edig@s* messages into the implementation guides *Edig@s* is laying down the foundations for the evolution of the EDIFACT syntax to that of XML.

The *Edig@s* working group also took the opportunity to suppress a significant amount of redundant information in order to optimize document maintenance and to facilitate the introduction of newcomers.

During the course of the modification of the messages the Edig@s working group came across a number of inconsistencies in the previous implementation guides as well as a number of documentation problems that could bring about divergent implementations.

Consequently these changes were of such a significantly important nature that it was necessary to clearly distinguish this new implementation guide from the previous one.

3 MESSAGE IMPLEMENTATION GUIDE STRUCTURE

1 INTRODUCTION
1.1 Functional definition
1.2 Principles
1.3 Field of application
1.4 References
2 INFORMATION MODEL OF xxxxxx
2.1 Information model structure
2.2 Information model description
3 EDIFACT IMPLEMENTATION OF xxxxxx
3.1 Edig@s subset of the UN/EDIFACT mmmmmm Branching Diagram
3.2 EDIFACT Template Description
4 XML IMPLEMENTATION OF xxxxxx
4.1 XML Structure
4.2 XML Schema
5 DOCUMENT CHANGE LOG

4 HOW TO READ AN INFORMATION MODEL

4.1 INTRODUCTION

The UML consists of a number of graphical elements that combine to form diagrams. The purpose of the diagrams is to present multiple views of a system. A set of such multiple views is called a model. The model fundamentally describes what a system is supposed to do. It does not describe how one implements the system.

Within the Edig@s workgroup the UML has been used to build a model of the requirements for the different phases of the Edig@s information model. This model has been built using several diagrams :

1. The use case diagram.
2. The data model.
3. The sequence diagram.

Each of these diagrams will be briefly described below.

4.2 THE USE CASE DIAGRAM

A Use case diagram is a description of a system's behaviour from a user's standpoint. Use-case diagrams graphically depict system behaviour (use cases). These diagrams present a high level view of how the system is used as viewed from an outsider's (actor's) perspective. A use-case diagram may depict all or some of the use cases of a system. A use-case diagram can contain actors ("things" outside the system), use cases (system boundaries identifying what the system should do) interactions or relationships between actors and use cases in the system including the associations, dependencies, and generalisations. Use-case diagrams can be used during analysis to capture the system requirements and to understand how the system should work. During the design phase, you can use use-case diagrams to specify the behavior of the system as implemented.

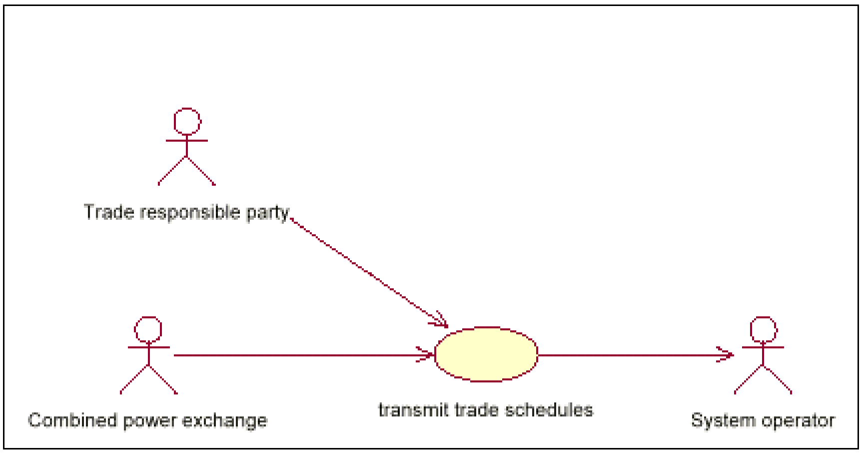


FIGURE 2: THE USE CASE DIAGRAM

The use case diagram shown in figure 2 shows that a trade responsible party and a combined power exchange send information to the "use case" (the ellipse being the use case) "transmit trade schedules" which routes them to the system operator.

4.3 THE DATA MODEL

A data model or class diagram that is used to define message structure and content. This is the general use of the class diagram as described above. The data model contains all the information required to describe the message defined by the model.

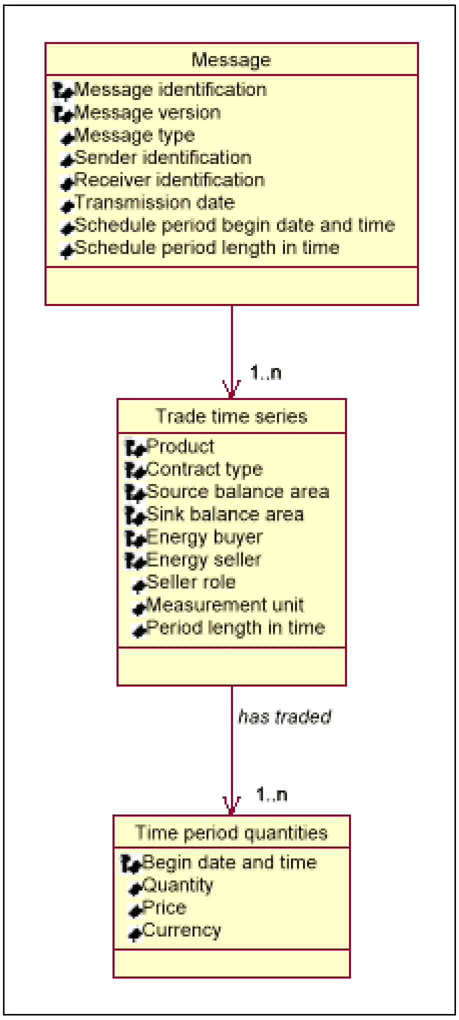


FIGURE 3: THE DATA MODEL DIAGRAM

Figure 3 shows a data model diagram for a trade time series. It is composed of three classes; the message class, the trade time series header class and the time period quantities class. Notice here each class contains attributes, which are the particular characteristics associated with the class. For the moment no particular operations have been identified for the classes. The arrows going between the boxes indicate the relationships between them. For example a message is composed of one to many time series and each time series is composed of one to many time period quantities. This information is deduced from the figures which are opposite the arrowheads.

4.4 THE SEQUENCE DIAGRAM

A sequence diagram is used to describe the interactions between actors (objects) over time. It is a graphical view of a scenario that shows object interaction in a time-based sequence; what happens first, what happens next.

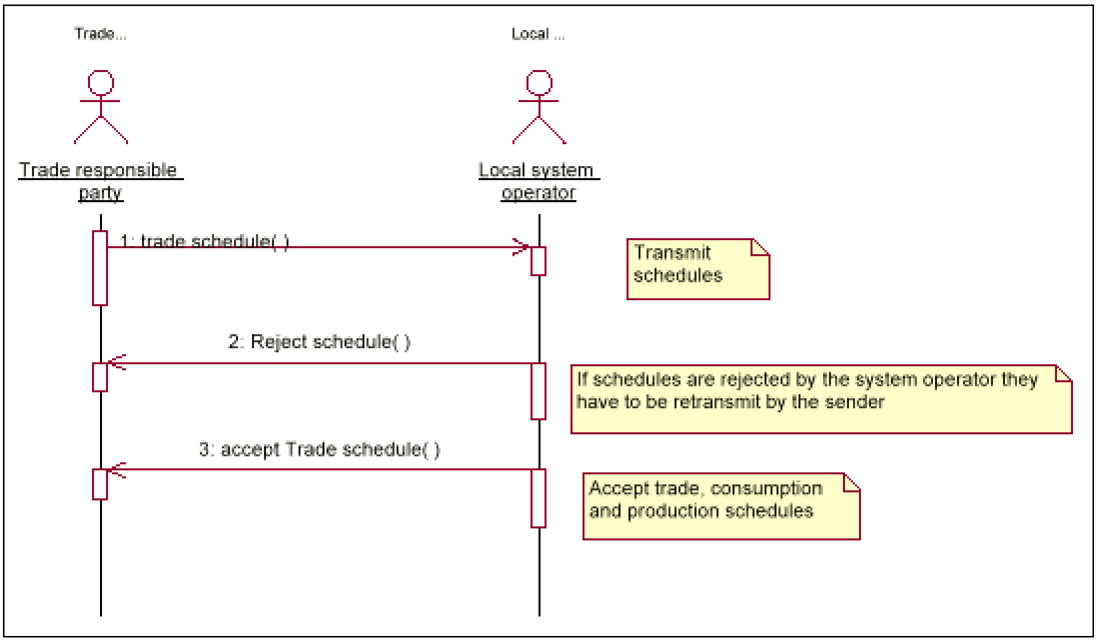
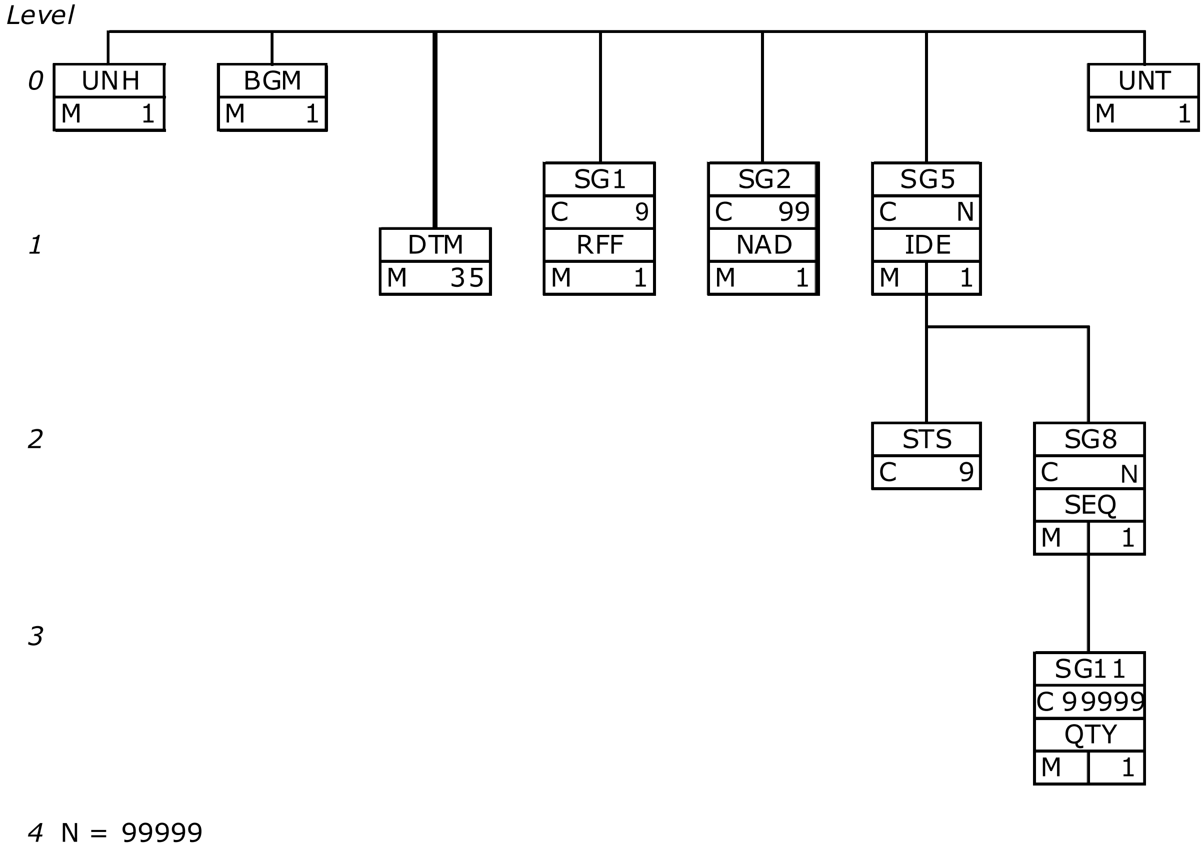


FIGURE 4: THE SEQUENCE DIAGRAM

The sequence diagram outlined in figure 4 shows the possible interactions between the trade responsible party and the local system operator. The trade responsible party initially sends the system operator his trade schedule. The system operator validates the schedule contents and may either reject the schedule, in which case the trade responsible party has to retransmit the schedule, or alternatively he may accept the schedule thus putting an end to this interaction.

5 INFORMATIVE ANNEXES

5.1 HOW TO READ AN EDIFACT BOILERPLATE



An EDIFACT Branching diagram is read left to right from top to bottom. The status of a segment is indicated as M for mandatory and C for conditional. The repetition factor indicates the number maximum of repetitions permitted by the UN/EDIFACT standard. The Edig@s EDIFACT implementation may use a number of repetitions lower than this maximum value.

The numbers on the lefthand side of the diagram indicate the depth of the segments within the diagram.

Note: The Branching diagram always provides the repetition factor as defined within the standard Edifact boilerplate for the message in question.

5.2 HOW TO READ AN EDIFACT SEGMENT TEMPLATE

All templates have been structured the same way and should be read as follows:

SG1 – M	**RFF**
Remarks	*The mandatory segment group 1 consists only of RFF.* *There will be only one occurrence of segment group 1 at header level to provide the contract identification relevant for the whole message.*

When a segment is the header segment of a segment group then the segment group is identified, its status (M= mandatory, C = Conditional) followed by the tag of the segment that triggers the segment group. This may be followed by other segments or segment groups that constitute the complete segment group. This is followed by remarks that explain the use of the segment group.

Every segment has the following structure:

RFF – M	**REFERENCE – To specify a reference.** **Identifies the contract relevant for this message**			
C506:1153	M	an..3	Reference code qualifier	Code qualifying a reference. **CT** = *Contract number*
C506:1154	M	an..35	Reference identifier	Identifies a reference. *Mutually agreed contract identification*
C506:1156	C	an..6	Document line identifier	To identify a line of a document. **NOT USED**
C506:1056	C	an..9	Version identifier	To identify a version. **NOT USED**
C506:1060	C	an..6	Revision identifier	To identify a revision. **NOT USED**
Remarks				
Example	**RFF+CT:TRABCRR01'**			

1. Segment tag and status (M = mandatory / C = conditional) followed by the name and purpose of the segment. The first line of the purpose describes the standard UN/EDIFACT purpose. The line following the standard purpose provides the explanation of the segment within the Edig@s environment.
2. Composite or single data element tag – tags starting with C or S identify composite data elements. This is followed by the Composite or single data element status (M = mandatory / C = conditional / N = not used) and the Composite or single data element format. Finally follows the definition of the element or the codes to be used. Any shaded elements are not allowed within the Edig@s specification.
3. Any eventual remarks concerning the segment.
4. An example of segment use.

5.3 HOW TO READ AN XML SCHEMA

Please refer to the W3C XML schema standard for details of reading an XML schema which can be found at the following address: http://www.w3.org/TR/xmlschema-0/. There is also an interesting tutorial on the subject on the same site.

DOCUMENT CHANGE LOG

Version	Date	Description
1	2007.12.31	version 4 issued
2	2009-04-27	Added the section “Quantity assignment” describing the rule for assigning values to data element 6060. Clarified the Branching diagram use of the repetition factor.