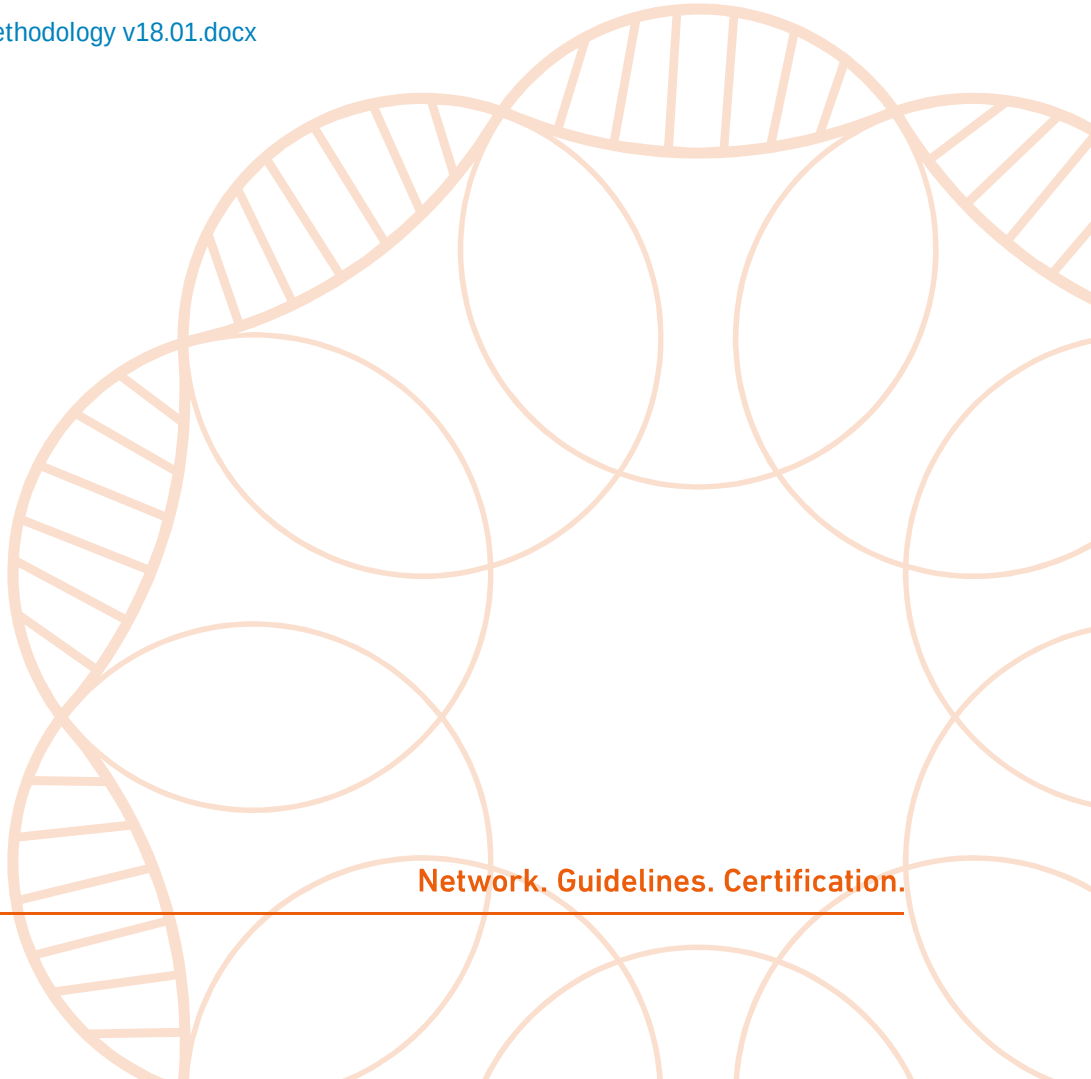
THE GLOBAL STANDARD
FOR LIVESTOCK DATA

Procedure 1 of Section 15 of ICAR Guidelines - Methodology

Procedure 1 – Methodology

Version July, 2018

File Ref: 15 Procedure 1 Methodology v18.01.docx



Network. Guidelines. Certification.

Table of Contents

Tables

Equations

No table of figures entries found.

Figures

Summary of Changes

Date of Change	Nature of Change
July 2018	New procedure created from methodogy chapter in Section 15 Overview. Replaced by new version from ADE-WG. Template applied. ADE WG feedback used to update last para in 4.2 to better reflect priorities of ADE WG.

1 Introduction

For electronic data exchange a lot of technologies exist and new technologies are continously evolving. However, in order to establish standardized data exchange we have to concentrate on a defined set of accepted and mature technologies and describe the usage in detail.

This procedure, as part of Section 15 Guidelines, introduces the common methodology and technology used by the specific business processes described in procedures 2, 3 and 4:

a. **General Data Transport Specifications**
describes the applied web service transport protocols and data protocols

b. **Access to the Service**
describes some aspects to be taken into account when implementing and installing the services

c. **Communication work flow**
covers the communication work flow aspects in a business process session like request, response and data processing.

d. **Dealing with Local Requirements**
describes in detail how locally obligatory data elements which are not covered by the ICAR scope can be added to the interfaces

e. **Common Components**
describes all the common protocol components which are reused in the specific business processes

2 Definitions and Terminology

Table 1 contains a list of important definitions for terms and abbreviations used in these guidlelines.

Table 1. Definitions of Terms used in these guidelines.

Term	Definition
TCP/IP	Transmission Control Protocol
HTTP	Hyper Text Transfer Protocol
XML	Extensible Markup Language
XSD	XML Schema Definition
SOAP	Simple Object Access Protocol
WSDL	Web Service Description Language
REST	Representational State Transfer
ADIS	Agricultural Data Interchange Syntax
ADED	Agricultural Data Element Dictionary
JSON	JavaScript Object Notation
UNCEFACT	UN/CEFACT is the United Nations Centre for Trade Facilitation and Electronic Business
ISO	International Organization for Standardization
IANA	Internet Assigned Numbers Authority

3 Scope

gives a pictorial summary of the main elements of this procedure. The numbers in this figure refer to the heading numbers of this procedure.

Figure 1. Scope of procedure 1 of Section 15 of ICAR Guidelines.

4 General Data Transport Specifications

4.1 SOAP/XML Web Services

For the primary data transmission method the W3C standard SOAP/XML version 1.2 has been chosen. Detailed descriptions of this standard can be found here:

http://www.w3.org/TR/2007/REC-soap12-part0-20070427/

Since a "top down" approach has been adapted, ICAR ADE provides WSDL files and a set of XSD files which make up a complete set of machine readable definition files from which SOAP web service implementations can be created. Most modern programming languages e.g. Java, .NET etc. provide tools which facilitate an easy setup process of interfaces classes to be used as a link between data transport layer and business process layer.

For each release of the ICAR ADE specifications a specific set of WSDL/XSD files can be downloaded from the ICAR ADE website – link here. See Table 2. ICAR ADE schema files" for a description of the files. The files will be provided in a zip archive file e.g. "ADE_Schema20150309_1.8.zip" for version 1.8 available here.

The WSDL/XSD files describe different aspects of the transport protocol:

f. Service end points
g. Input and output messages
h. Basic data type definitions (XSD types, UNCEFACT types etc.)
i. Complex data type definitions
(entity tags, item tags, compositions, attributes, usage of basic data types)
j. Constraints (data type and value range restrictions)

Table 2. ICAR ADE schema files

File Name / File Name Schema	File Description
*.XSD	XML Schema Definition files, machine readable description of the ICAR ADE XML data structures
ICAR_*.XSD	Data element definitions created by the ICAR ADE group
ISO_*.XSD	Data element definitions derived from ISO standards
UNECE_*.XSD	XSD files provided by the UN/CEFACT project: (see http://www.unece.org/cefact/xml_schemas/index.html)
UnqualifiedDataType_13p0.XSD	XSD file which defines the unqualified UNCEFACT data types. These data types extend the standard XSD data types with additional attributes. In the ICAR and UNCEFACT definitions they are used instead of the standard XSD types.
*_CODE_*	XSD files defining code lists
wsMrAde.wsdl	ICAR ADE SOAP service description, (WSDL = Web Service Description Language) It defines a set of message pairs each consisting in a request and a response message.
ADE_v1p8.xsd	This file provides the entry point to the ICAR ADE message data structure definitions

Throughout this document messages and data definitions are illustrated using graphical presentations of XSD structures created by the commercial software XMLSPY from ALTOVA.

See for example Figure 2 Example “GetHerdListRequest”

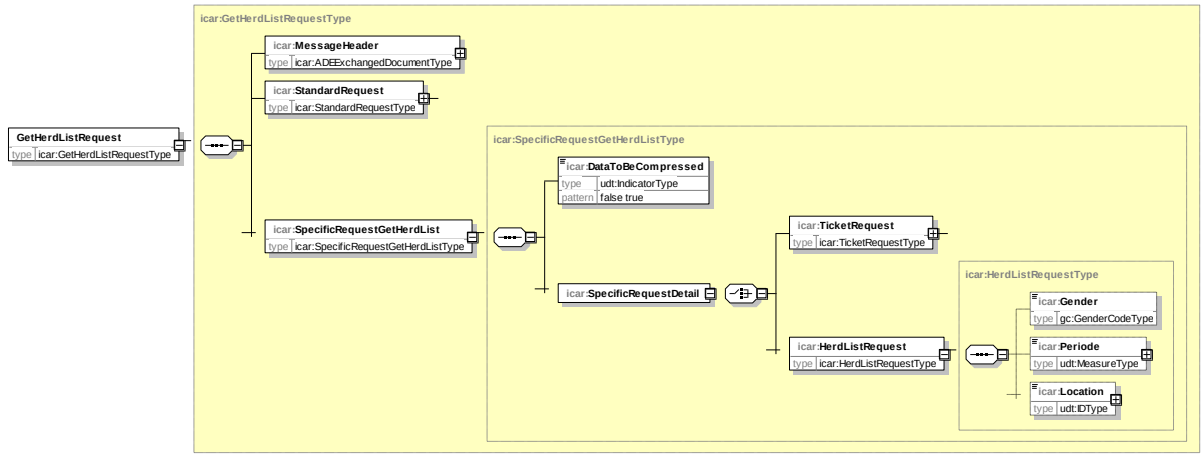
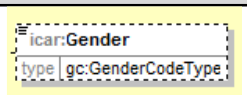
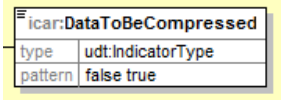
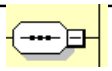
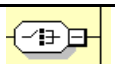
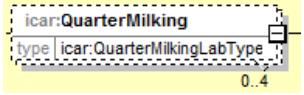


Figure 2 Example “GetHerdListRequest”

Below the symbols used to represent the XSD schemas in the following pages:

Table 3 XSD schema symbols

Description	Symbol
Optional tag	
Mandatory tag	
Sequence	
Choice	
Multiplicity	

4.2 Alternative Transport Protocols (ADIS/ADED/IsoAgriNet, REST)

Parallel to SOAP other international transport protocols exist. This is for example the ISO ADIS/ADED standard and its last extensions described in ISO 17532(2007). On the other hand the REST protocol, based on the HTTP transport protocol using XML, JSON and other data encoding technology, has gained a broader acceptance in the domain of internet data transfer during the recent years.

The ADE-WG is working on the development of a standardized REST-API description.

4.3 ICAR Types and Code Lists

Elements created within the process of the ICAR ADE specification can be identified by the prefix "**icar:**" within the WSDL/XSD files and descriptions below.

All data items are based on UNCEFACT data types (see section UNCEFACT Data Types)

A detailed description of the data elements is given below.

4.4 External Elements

It is the aim of the ICAR ADE standardization process to make as far as possible use of elements already defined within existing standardization frameworks as ISO, IANA and UNCEFACT.

4.4.1 UNCEFACT Data types

The ICAR ADE WSDL/XSD definitions make extended use of UNCEFACT basic types.

UNCEFACT basic types are types derived from basic XSD types. The concept consists mainly of adding further attributes and restrictions to the XSD types. They are defined in file UnqualifiedDataType_13p0.xsd.

In the WSDL/XSD files and descriptions below those elements can be identified by the prefix "**udt:**"

Table 4 UNCEFACT basic data types" lists the UNCEFACT types currently used for the types of a data item.

Table 4 UNCEFACT basic data types

Name	Description	Comment	XSD Type
udt:CodeType	A character string that may be used to represent or replace the definitive value.	A code is referring to an enumeration	String
udt:DateType	A particular point in the progression of date	Representation as defined by transport protocol	String
udt:DateTimeType	A particular point in the progression of time	Representation as defined by transport protocol	String
udt:IDType	A character string used uniquely to establish the identity of, and distinguish, one instance of an object within an identification scheme from all other objects within the same scheme.		String
udt:MeasureType	A numeric value determined by measuring an object	Measures are specified with a unit attribute holding a measure unit, a resolution and when it is required a minimum and a maximum value.	Decimal
udt:NameType	A word that constitutes the distinctive designation of a person, place, thing, or concept		String
udt:BinaryObjectType	Base64 coded binary data	Used e.g. for zip compressed message	String
udt:TextType	A general text type		String
udt:IndicatorType	A Boolean indicator or true and false		Boolean

4.4.2 UNCEFACT Code Lists

UNCEFACT provides an extensive set of code lists. See Table 5 UNCEFACT Code Lists" for the code lists used by the ICAR ADE specifications.

Table 5 UNCEFACT Code Lists

Code List	Description

AgencyIdentificationCode	A list of agencies responsible for code list maintenance to be used as value of the attribute **listAgencyID** in the UNCEFACT data type **udt:CodeType**
CharacterSetEncodingCode	A list of character set encoding codes to be used as value of the attribute **encodingCode** in the UNCEFACT data type **xsd:base64Binary**
MeasurementUnitCommonCode	An extensive set of commonly used units to be used as value of the attribute **unitCode** in the UNCEFACT data type **udt:MeasureType**

4.4.3 ISO Code Lists

See Table 6 ISO code lists for the ISO code lists used by the ICAR ADE specifications.

Table 6 ISO code lists

Code List	Description
ISO3AlphaCurrencyCode	A list of ISO currency codes to be used as value of the attribute **currencyID** in the UNCEFACT data type **udt:AmountType**
ISOTwoletterCountryCode	A list of two letter ISO country codes to be used as value of the item **Country**

4.4.4 IANA Code Lists

See Table 7 IANA Code Lists for the IANA code lists used by the ICAR ADE specifications.

Table 7 IANA Code Lists

Code List	Description
CharacterSetCode	A list of IANA character set codes to be used as value of the attribute **characterSetCode** in the UNCEFACT data type **xsd:base64Binary**
MIMEMediaType	A list of IANA MIME media type codes to be used as value of the attribute **mimeCode** in the UNCEFACT data type **xsd:base64Binary**

5 Access to the Service

5.1 Exchange Parameters

The equipment should provide the operator with an interface to capture exchange parameters for a given service provider consisting of:

a. Service provider URL
b. Authentication information (user/password, token)
c. Sender ID
d. Sender name
e. Sender country
f. Recipient ID
g. Recipient name
h. Recipient country
i. Type of location
j. Type of identifier of the location
k. Identifier of the location

l. Name of the location
m. Country of the location
n. Type of primary animal identifier
o. Type of secondary animal identifiers (e.g. on farm animal id, animal name)
p. Other manufacturer or service provider specific parameters

For a given service provider the exchange parameters should be the same for all the consumed services.

5.2 Data Transmission

As the services are not permanently accessible, no request should be sent by the equipment before having checked if there is access to:

a. The network
b. The service provider server
c. The service itself

Any new data should be registered by the information system as soon as possible.

No new data should be sent to the information system as long as previous data is still being processed. The service consumer has to wait for a processing result or a ticket in case of asynchronous processing.

No ticket should be sent to the information system before the time given by the information system.

Any message whose syntax is not in compliance with the syntax requirements should not be sent.

As soon as a data is included in a request an answered by a response it should be considered as sent to the information system.

Service providers and consumers should be prepared for updates and deletes. The service provider should take also take into account that the sending party is declaring updates as a create operation not knowing that the data is already present in the service provider's data base. (e.g. in case of retransmission initiated by the consumer)

Preparing a request in compliance with business requirements should be performed separately from preparing a message whose syntax depends on the type of technical mapping (web service with SOAP at the moment).

5.3 Encryption

If possible encryption should be used in order to insure privacy and authenticity. The process of encryption itself is not described here. The encryption capabilities of the applied transport protocol should be used e.g. HTTPS for HTTP-Transport.

5.4 Compression

In order to minimize the transport delay of big messages, especially when coded in xml, compression should be applied. Some transport protocols offer compression features like transparent gzip

compression of http responses. However since http compression is only applicable to the response it is not usable for the compression of data loads in http requests.

In order to avoid this limitation a special XML based compression mechanism is described for ICAR data exchange. Specific messages can be defined as zip compressible offering the alternative of uncompressed or zip compressed Base64 binary encoded xml structures both for request and response. For details see chapter “Message Design” below.

5.5 User Right Verification

The right of a given user to use a given service is verified by either using an “**authentication token**” which prevents him from sending a user ID and a password at each request or by a **user name authenticated by a password**.

Authentication by* user name and password *should only be used with encrypted transport.

6 Communication work flow

6.1 Data Processing

Any service may be provided either in real time or in time differed.

A response should be sent to any response within several seconds.

Any request and any response should be stored by the equipment and the information system.

The requirements to update the data base of the information system differ from one service provider to another one.

Any request which is not in compliance with one the following conditions should not be processed:

a. Compliance with the syntax
b. Validity of the identification token
c. Validity of the ticket if any

6.2 Request Specification

Any request should consist of three parts:

a. A single **message header**
(see the entity MessageHeader of Type “ADEExchangedDocumentType”
in ‘Data description’)
The sender of the request creates and provides his own unique MessageID
(Item ADEExchangedDocumentType.Identifier).
See chapter “Best Praxis for MessageID Creation” below.

b. A single **standard request**
(see the entity StandardRequest in ‘Data description’)

c. A **specific request** which may consist of
a specific message for a given service
or the zip compressed base64 binary of the specific message
or a ticket.
A ticket is sent in order to retrieve the results of an asynchronous message processing task. (see diagrams below)

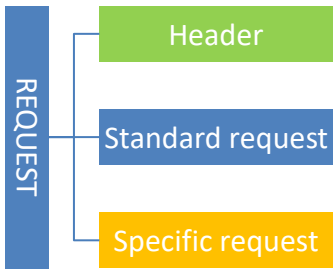


Figure 3 Request with a specific request

6.3 Response Specification

Any messsage response should consist of three parts:

a. A single message header (see the entity MessageHeader of Type "ADEExchangedDocumentType" in 'Data description')
The sender of the response creates and provides his own unique Message id.
(See chapter "Best Praxis for MessageID Creation" below)

b. A single standard response (see the entity StandardResponse in 'Data description')
In the standard response the original MessageID from the sender is returned in the item RequestID.
The standard response contains the item RequestProcessingStatus which identifies the status of the request processing on the side of the service provider:

Table 8 RequestProcessingStatusCodeType (base xsd:string)

Key	Description
O	Processed without errors
P	Data accepted for asynchronous processing (client can retrieve the processing result later using provided ticket)
E	Processed and rejected due to errors
W	Processed with warnings

c. A specific response which may consist of either a ticket or specific message for a given service or the zip compressed base64 binary of the specific message.
The existence and content of the specific response depends on the definition of the service and the content of the RequestProcessingStatus item.
In case of RequestProcessingStatus = 'P' the specific response holds a Ticket
If a specific response message is defined it can either be represented uncompressed or zip compressed of type Base64binary.
If RequestProcessingStatus = 'P', 'E' or 'W' there might be no specific response at all depending on the message definition.

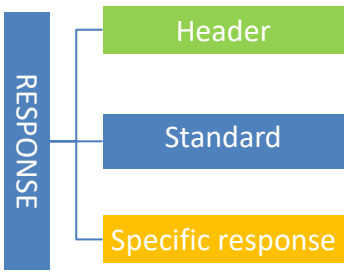


Figure 4 Response with a Specific Response

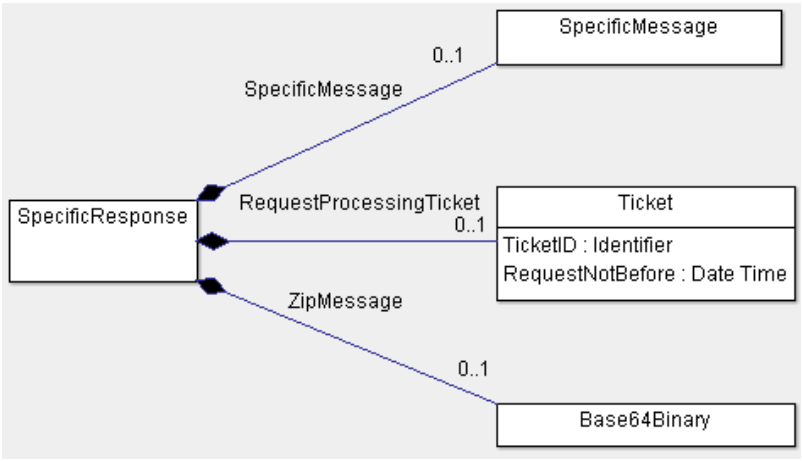


Figure 5 Specific Response Schema

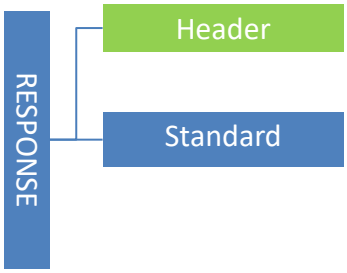


Figure 6 Standard Response

6.4 Best Praxis for Message-Id Creation

The message ID created by both service consumer and service provider should be globally unique in order to allow a simple identification of each message e.g. for tracing and debugging purposes.

The MessageID should be based on a chain of unique hardware identifier, a unique software identifier and an additional distinguishing component large enough to uniquely identify messages created at very fast rates. This dynamic component could be a timestamp with ms precision or a sequence. The unique hardware identifier could be the globally unique MAC address of the network interface. The unique software identifier should be unique on the system running the software.

There are tools in different programming languages which provide simple means of unique id creation e.g. Guid.NewGuid() in C#

7 Dealing with Local Requirements

The ICAR ADE standardization process cannot take into account all local requirements. In the first place it offers a common basis to build on.

In the first place "local" is a placeholder for national standards. However it is also possible to add local extensions based on an agreement between a service consumer and a service provider (e.g. for special projects, prototypes etc.).

In order to allow local extensions and to fill standardization gaps which cannot be covered by ICAR the ADE specifications provide two ways of local variations and extensions:

a. **Local Code Lists**:
   Local code lists for all items of type **udt:CodeType** for which ICAR is not yet able to provide a commonly accepted code set

b. **Local Additional Data**:
   An optional list of key value pairs in each specific entity to be used for locally required data elements

Local variation should be reduced to a minimum since it is contradictory to the goal of ICAR to establish global standards, leads to increased complexity, complicates software development and creates additional costs.

7.1 Local Code Lists

Local code lists must be provided for items of type **udt:CodeType** in the context of a local standardization process, e.g. breed codes.

The content cannot be checked with the SOAP/XML validation procedure but must be evaluated in an extra program step before filling the XML data structures and after extracting the data from the XML structures. It is the responsibility of the software developers to assure that the data exchange is in accordance to ICAR and to local specifications.

In order to help manufacturers with the maintenance and integration of local code lists these specifications define the service **GetLocalCodeList** which can be used to query local code lists from the local service providers. For details see section "Technical Services".

7.2 Local Additional Data

Sometimes it is necessary for local implementations of the ICAR messages to add local data items to the specific messages which are not known or not covered by the ICAR specification process.

For those scenarios a simple and flexible extension (**LocalAdditionalData**) has been added to each specific entity.
It consists of an optional list of code/value pairs of type udt:textType which can be filled according to local definitions.

It is the responsibility of the local business process owners to document and assure the proper handling of this code/value pairs. The ICAR wsdl/xsd definitions only check the proper usage of the XML structure not the content.

For details see the wsdl/xsd files and the section "CommonComponents"-"LocalAdditionalDataType".

8 Common Components

This part describes common components which are used in all the specific services described in anex B, C and D.

8.1 ADE Exchanged Document Type

The entity **ADEExchangedDocumentType** (Figure 7 ADEExchangedDocumentType) gives information describing general aspects of the message.

Used by data element: **MessageHeader**

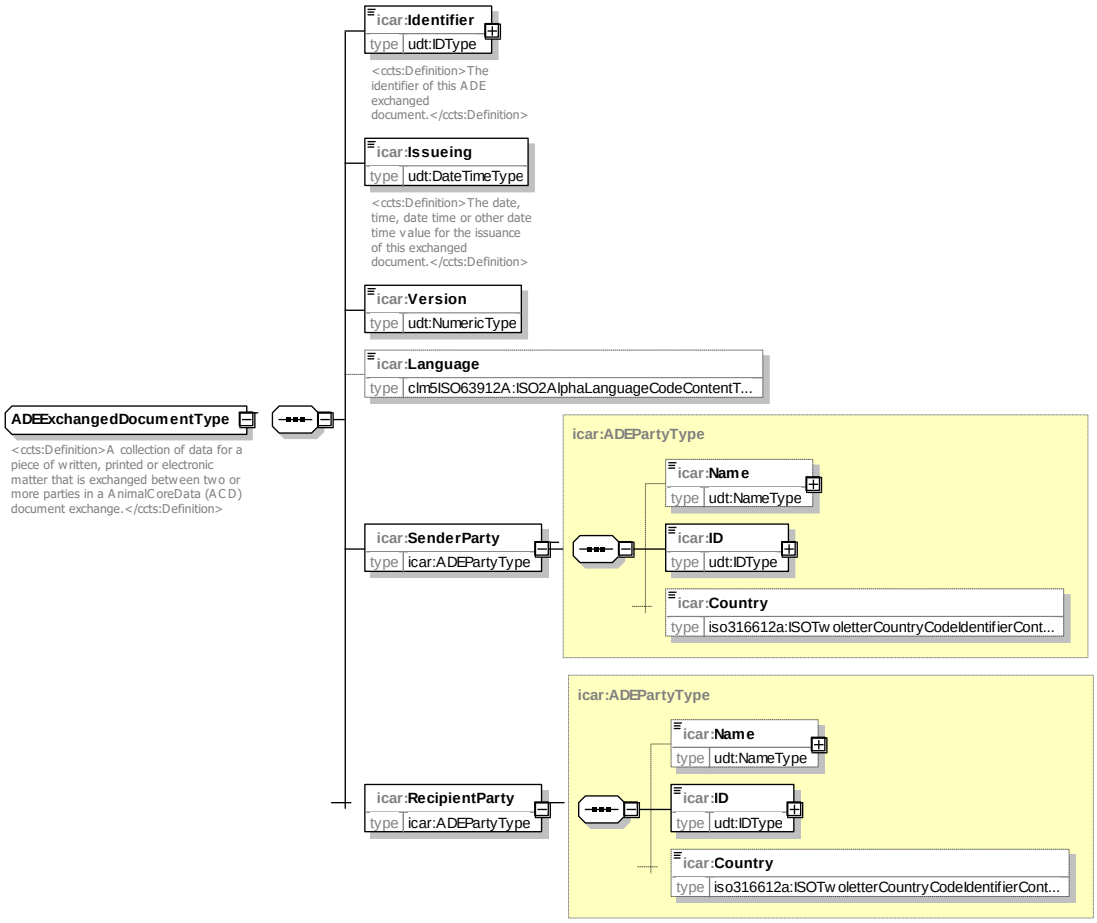


Figure 7 ADEExchangedDocumentType

a. **Identifier**: Unique identifier of the message given by the sender
b. **Issueing**: The date and the time where the message is issued.
c. **Version**: Version number of the message
d. **Language**: Language used for the message
e. **LocalAdditionalData**: Optional list of key/value pairs used in a local context as described in section "Common Components – Local Adaptions"
f. **SenderParty**: Organization or person responsible for the content of the message, not a server identifier.
g. **RecipientParty**: Organization or person responsible for processing the message, not a server identifier.

8.2 ADE Party Type

The entity **ADEPartyType** (Figure 8 ADEPartyType) gives the name and the identifier of a party which may be either the sender or the recipient of a message.

Used by data elements: **SenderParty**, **RecipientParty**

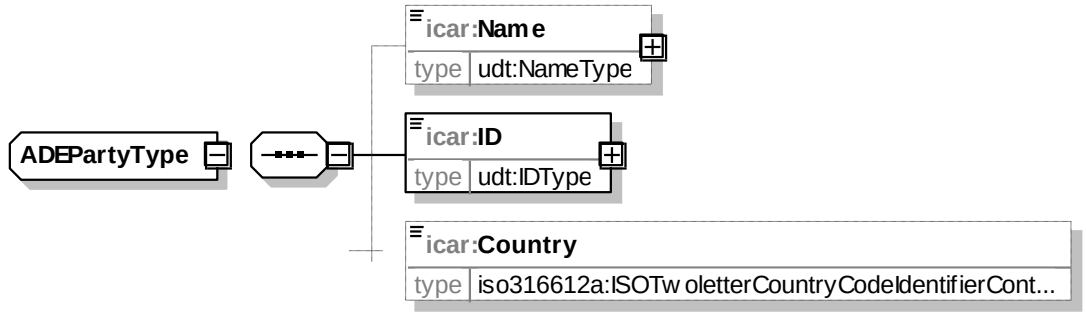


Figure 8 ADEPartyType

- **Name**: Identifier of the specified party
- **ID**: Identifier of the specified party
- **Country**: Country of the specified party in accordance to code list **ISOTwoletterCountryCodeIdentifierContent**

8.3 Standard Request Type

The entity **StandardRequestType** (Figure 9 StandardRequestType) gives the standard content of a request.

Two different types of authentication can be used, token or user/password based.

Used by data element: **StandardRequest**

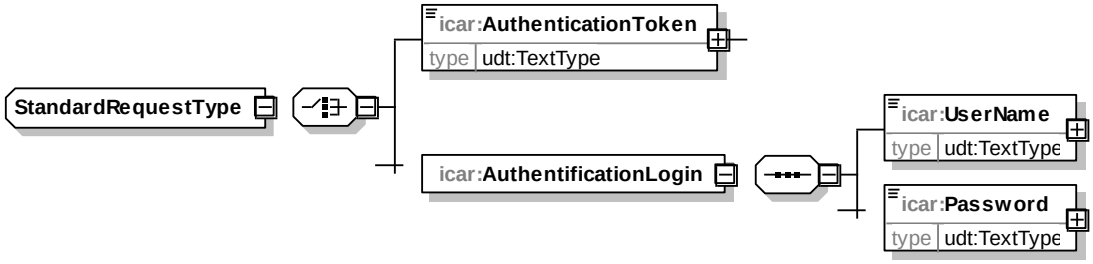


Figure 9 StandardRequestType

a. **AuthenticationToken:** temporal code used for verification of the user's identity and of its right to use a service.

b. **AuthentificationLogin**: Pair of username and password used for login

c. **UserName**: User name used for authentication

d. **Password**: Password used for authentication

8.4 Standard Response Type

The entity **StandardResponseType (**Figure 10 StandardResponseType**)** gives the standard content of a response.

Used by data element: **StandardResponse**

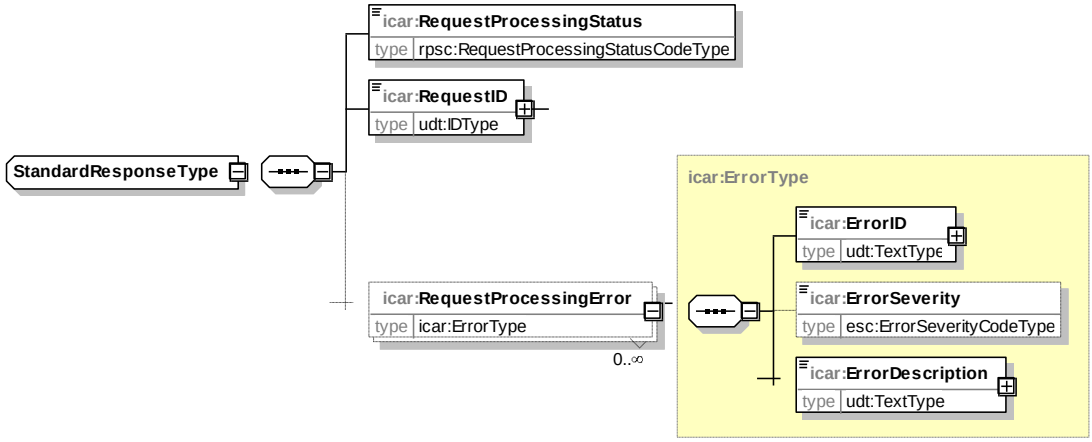


Figure 10 StandardResponseType

a. **RequestProcessingStatus**: Status of request processing returned by server according to code list **RequestProcessingStatusCode**

b. **RequestID**: Copy of the message identification in the **request's MessageHeader.Identifier** item

c. **RequestProcessingError**: In case an error happens during the processing of the request an error description is provided here

8.5 Error Type

The entity **ErrorType** (Figure 11 ErrorType) gives the request processing errors.

Used by data element: **RequestProcessingError**

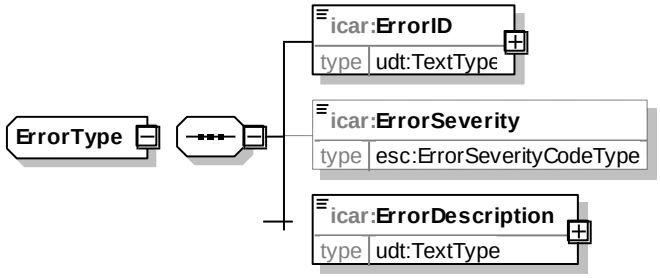


Figure 11 ErrorType

a. **ErrorID**: Identifier the error, e.g. error text message, local error code etc.

b. **ErrorSeverity**: Severity of the error according to code list **ErrorSeverityCode**

c. **ErrorDescription**: Human readable description of the error as text

8.6 Ticket Response Type

The entity **TicketResponseType** (Figure 12TicketResponseType) conveys a ticket in order to retrieve a response processed in time differed (asynchronous processing mode).

It is used in **SpecificRequest** and **SpecificResponse** by data element **TicketResponse**.

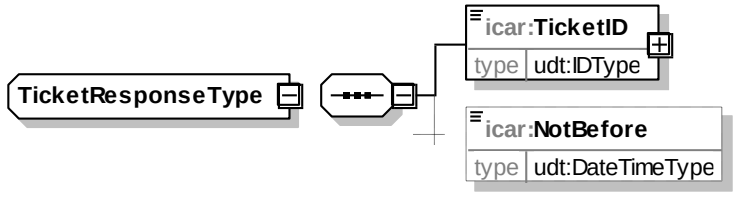


Figure 12TicketResponseType

a. **TicketID**: Identify the response to retrieve

b. **NotBefore**: Date time of probable response availability

8.7 ZIP Message Type

ZipMessageType (Figure 13 ZipMessageType) is a wrapper type designed to transport the zip compressed version of the specific data part of a message. Contrary to http compression method which can only be used for the response part this method can be applied to request and response. It is defined as a choice for each specific request or response which transports data entities.

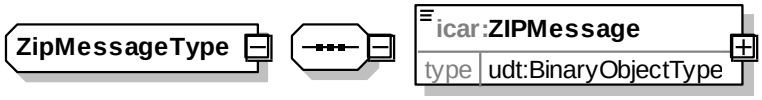


Figure 13 ZipMessageType

a. **ZIPMessage**: zip compressed data

8.8 Local Additional Data Type

The entity **LocalAdditionalDataType** (Figure 14 LocalAdditionalDataType) is a container for code value pairs for local usage outside the ICAR specification scope.

It is used as an optional list element **LocalAdditionalData** in each specific entity.

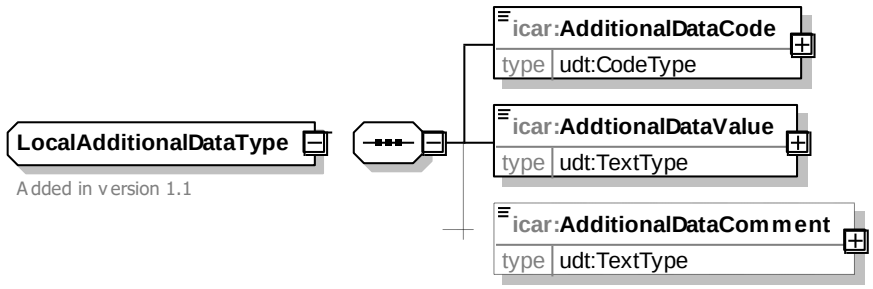


Figure 14 LocalAdditionalDataType

a. **AdditionalDataCode**: Name of the code value pair

b. **AdditionalDataValue**: Value of the code value pair

c. **AdditionalDataComment**: Optional description of the local data

8.9 Time Period Type

The utility type **TimePeriodType** (Figure 15TimePeriodType) is a wrapper type designed to transport the beginning and end of a time period.

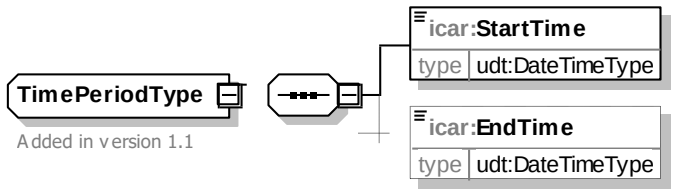


Figure 15TimePeriodType

a. **StartTime**: The start of the time period
b. **EndTime**: The optional end of the time period

9 Naming Rules

Names of data elements, operations and messages use upper camel case.

Service name consist of:

a. A prefix which gives the type of operation according the following:
   - Create: Insert data into a data base
   - Update: update of a data base
   - Get: retrieve data from data base
   - Delete: delete data from data base
b. The name of the service

For example UpdateMilkingResult is the service which allows updating a data base from milking results exchange.

Message name consists of:

a. The name of the service
b. A suffix, gives the type of message:
   - Request
   - Response

For example, UpdateMilkingResultRequest is the request to trigger the service UpdateMilkingResult.

10 References

1. Semantics for Smart Dairy Farming: a milk production registration standard – SDF June 2013
2. UN / UNCEFACT Modeling Methodology User Guide (CEFACT / TMG/N093)
3. UN / UNCEFACT Business Requirements Specifications Document Template (CEFECT/ICG/005)
4. ISO 11787 : Electronic data interchange between information systems in agriculture — Agricultural data interchange syntax
5. ISO 11788 : Electronic data interchange between information systems in agriculture — Agricultural data element dictionary —Part 1: General description —Part 2: Dairy farming
6. ISO 17532 : Stationary equipment for agriculture —Data communications network for livestock farming
7. ISO 11784: Radio frequency identification of animals - code structure

8. ISO 3166 -1 : Country code
9. ICAR G