INTERNATIONAL STANDARD

ISO 11783-10

First edition
2009-12-15

Tractors and machinery for agriculture and forestry — Serial control and communications data network — Part 10: Task controller and management information system data interchange

Tracteurs et matériels agricoles et forestiers — Réseaux de commande et de communication de données en série —

Partie 10: Contrôleur de tâches et échange de données des systèmes d'information de gestion

Reference number
ISO 11783-10:2009(E)

Contents

Page

Foreword

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

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 11783-10 was prepared by Technical Committee ISO/TC 23, *Tractors and machinery for agriculture and forestry*, Subcommittee SC 19, *Agricultural electronics*.

ISO 11783 consists of the following parts, under the general title *Tractors and machinery for agriculture and forestry — Serial control and communications data network*:

— *Part 1: General standard for mobile data communication*

— *Part 2: Physical layer*

— *Part 3: Data link layer*

— *Part 4: Network layer*

— *Part 5: Network management*

— *Part 6: Virtual terminal*

— *Part 7: Implement messages application layer*

— *Part 8: Power train messages*

— *Part 9: Tractor ECU*

— *Part 10: Task controller and management information system data interchange*

— *Part 11: Mobile data element dictionary*

— *Part 12: Diagnostics services*

— *Part 13: File server*

— *Part 14: Sequence control*

Introduction

ISO 11783 specifies a communications system for agricultural equipment based on the CAN 2.0 B[2] protocol. SAE J1939[3] documents, on which parts of ISO 11783 are based, were developed jointly for use in truck and bus applications and for construction and agriculture applications. Joint documents were completed to allow electronic units that meet the truck and bus SAE J1939[3] specifications to be used by agricultural and forestry equipment with minimal changes.

General information on ISO 11783 is to be found in ISO 11783-1. The purpose of ISO 11783 is to provide an open, interconnected system for on-board electronic systems. It is intended to enable electronic control units (ECUs) to communicate with each other, providing a standardized system.

The International Organization for Standardization (ISO) draws attention to the fact that it is claimed that compliance with this part of ISO 11783 may involve the use of a patent concerning the controller area network (CAN) protocol referred to throughout the document.

ISO takes no position concerning the evidence, validity and scope of this patent.

The holder of this patent has assured ISO that he is willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with ISO. Information may be obtained from:

Robert Bosch GmbH
Wernerstrasse 51
Postfach 30 02 20
D-70442 Stuttgart-Feuerbach
Germany

Attention is drawn to the possibility that some of the elements of this part of ISO 11783 may be the subject of patent rights other than those identified above. ISO shall not be held responsible for identifying any or all such patent rights.

Tractors and machinery for agriculture and forestry — Serial control and communications data network —

Part 10: Task controller and management information system data interchange

1 Scope

ISO 11783 as a whole specifies a serial data network for control and communications on forestry or agricultural tractors and mounted, semi-mounted, towed or self-propelled implements. Its purpose is to standardize the method and format of transfer of data between sensors, actuators, control elements and information storage and display units, whether mounted on, or part of, the tractor or implement. This part of ISO 11783 describes the task-controller applications layer, which defines the requirements and services needed for communicating between the task controller and electronic control units. The data format to communicate with the farm-management computer, the calculations required for control and the message format sent to the control function are defined in this part of ISO 11783.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 11783-1, *Tractors and machinery for agriculture and forestry — Serial control and communications data network — Part 1: General standard for mobile data communication*

ISO 11783-3, *Tractors and machinery for agriculture and forestry — Serial control and communications data network — Part 3: Data link layer*

ISO 11783-5, *Tractors and machinery for agriculture and forestry — Serial control and communications data network — Part 5: Network management*

ISO 11783-6, *Tractors and machinery for agriculture and forestry — Serial control and communications data network — Part 6: Virtual terminal*

ISO 11783-7, *Tractors and machinery for agriculture and forestry — Serial control and communications data network — Part 7: Implement messages application layer*

ISO 11783-11, *Tractors and machinery for agriculture and forestry — Serial control and communications data network — Part 11: Mobile data element dictionary*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 11783-1 apply.

4 Abbreviations

For the purposes of this document, the abbreviations given in ISO 11783-1 apply.

5 General description

5.1 Task management

There are two main purposes of task management in the mobile implement control system.

The first is the management of the farm resources, including tractors, implements, sensor systems, workers and the products used. It is possible for the farmer to plan and evaluate the use of the resources. He is then able to automatically control the use of his inventory of products and can keep track of the status and conditions of his machinery. Resource designators are globally transferred as coding data and are part of the data transfer file as detailed in Clause 7.

The second purpose is the management of the farm activities carried out in the fields. These activities are described by tasks to distinguish all the work that is planned or has been done by the farmer or by a contractor for one customer in one partfield.

The data transfer is possible in two directions. The planned tasks are sent to the task controller on the MICS (mobile implement control system) and the results of the work are sent back to the FMIS (farm-management information system). Tasks can be generated both on the FMIS and on the MICS.

Task management has the following workflow:

a) Planning field tasks and maintaining coding data using the software of an FMIS computer operated by the farmers or contractors, as detailed in 5.2.

b) Converting the task data into XML format.

c) Assigning the task data produced by the planning software to the data required for the implements or sensor systems to be used to complete the planned tasks. This step is optional.

d) Transferring the task data from the FMIS system to the task controller of the MICS, as detailed in 5.4.

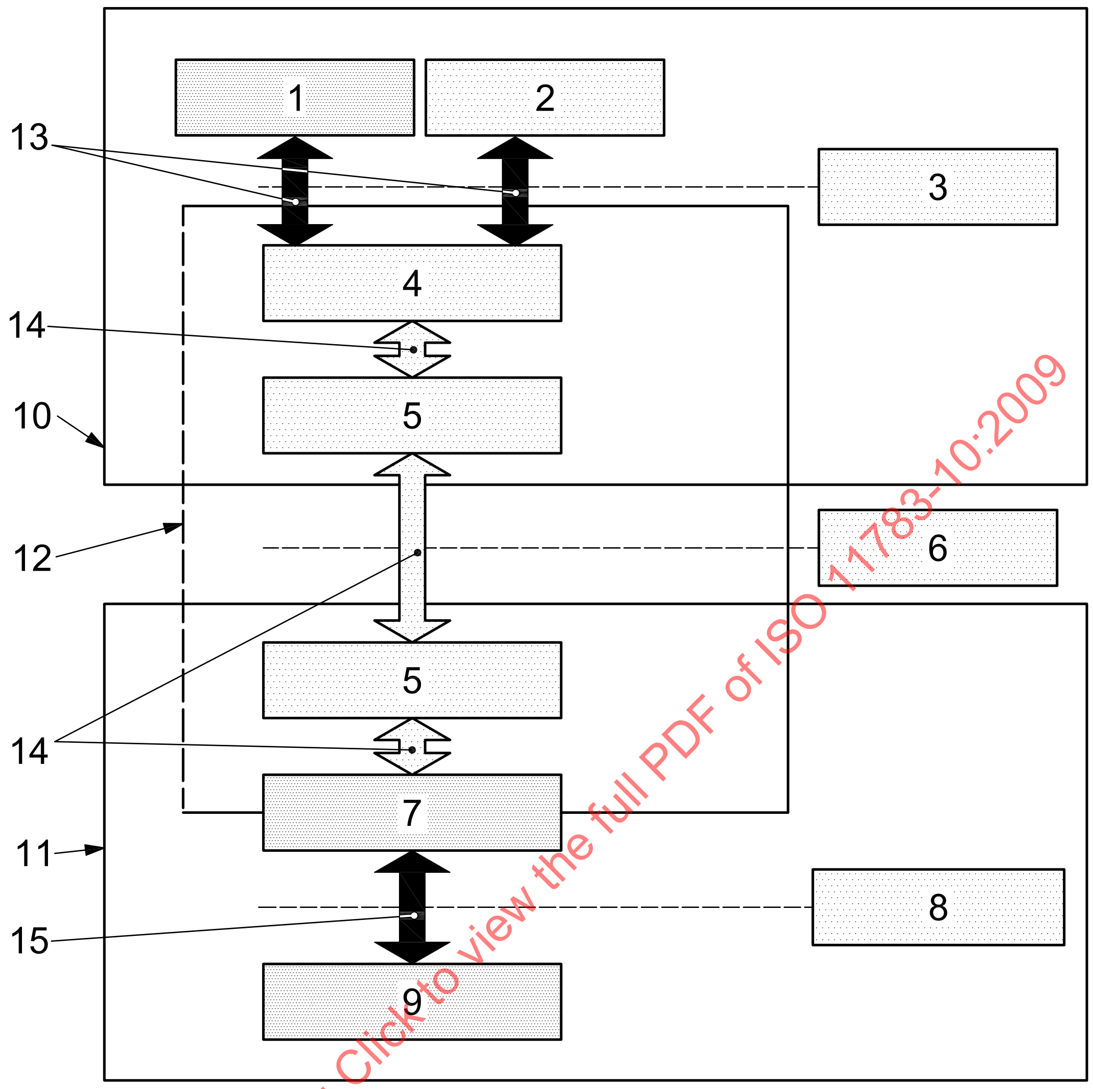
e) The task controller uses the task data to transmit process data messages to the ECUs on the implement.

f) The task controller collects data according to the DataLogTriggers specified in the task data.

g) Transferring the collected data to the FMIS. Collected data can be in the XML format or in a proprietary format. When a proprietary format is used, this step involves converting the proprietary format into the XML format.

h) Reading the XML files and converting into the FMIS format for storage and evaluation of the data.

Figure 1 illustrates the interfaces between the software on the FMIS computer and the ECUs mounted on an ISO 11783-configured implement.



Key

1 management information system
2 implement configuration data
3 XML format
4 task-controller configuration program
5 task-controller interface driver
6 task-controller data carrier
7 ISO 11783 task controller
8 ISO 11783 network
9 ISO 11783 control function
10 farm-management information system (FMIS)
11 mobile implement control system (MICS)
12 not standardized
13 standardized data transfer with data transfer file
14 standardized or proprietary data format
15 standardized data transfer within the ISO 11783 network

Figure 1 — Task management entities and interfaces

5.2 Task management on FMIS computer

Task management is defined as a part of an FMIS, responsible for planning and evaluation of field work. Tasks specify what, where, how, by whom and when work is planned to be carried out.

The amount of data transferred between the FMIS and the MICS is dictated by the administrative requirements of a farming enterprise. For the recording of field activities only, the task management can be used to file the data in a working journal. For this purpose, only the coding data need be transferred from FMIS to MICS, and the tasks are created on the MICS by selection of the involved resources. In this case, only the data transfer file from MICS to FMIS contains tasks. In enterprises where tasks are planned on the FMIS, these will be included, together with the coding data, in the data transfer file from FMIS to MICS. These planned tasks can range from mere planned allocations of resources to geographical information for site-specific field operations.

5.3 Preselection and assignment of working sets

Any device in the mobile system can only be identified uniquely by its working-set master's NAME. At the FMIS, the preselection of devices depends on the planned task. It can be necessary to assign a type of device or function, a specific device or even the device of a particular manufacturer. The DeviceAllocation XML elements may include planned assignments about the working sets to be used. This information ranges between specific and indistinct.

The XML attribute WorkingSetMasterNAMEValue contains the eight-byte NAME of a control function as it is defined in ISO 11783-5. Not all parts of a NAME need be specified and only certain elements of a NAME need to be defined to determine a device on the mobile network. Those parts of WorkingSetMasterNAMEValue containing information to be used on the mobile system to select a proper device are masked by a bitset structure stored in the XML attribute, WorkingSetMasterNAMEMask. All different combinations of elements of the NAME structure can be marked as valid for device selection (logical AND). On the FMIS, these masks could be coded as symbols. Once the preselection information is set in a task on the FMIS, it is not overwritten on the mobile system, because the working set that is used during task processing on the mobile system is stored as XML attribute DeviceIdRef.

5.4 Task-controller interface driver

After generating the interface files, the task-controller driver of the manufacturer of the task controller is activated on the farm computer. This driver is responsible for the data transfer to the task controller, which is part of the MICS using its proprietary data format or the ISO 11783-10 XML format and data carrier, such as any type of memory card or radio link. The translation of the data from the data transfer files in messages on the ISO 11783 network, as well as the kind of transfer between the mobile system and the FMIS, is not subject to a standardized procedure. To make implement-specific set point data, this driver can add and use device description data supplied by the manufacturers.

5.5 Task-controller user interface

A task controller can provide a means for the user to interact with the task controller. User interaction can be through a VT (virtual terminal) or other interface. Operator interfaces can range from very simple to elaborate, depending on the designer's intent. For example, a simple task controller that only supports single tasks that run automatically might not require any user interaction. More advanced task controllers can offer additional operator interface capabilities, such as

— select a task from a list,

— start/stop/resume a task,

— modify a task,

— create a task, and

— add new coding data.

Through the user interface, an operator can react to special circumstances or events in order to execute tasks in a reasonable way. The operator can also be informed about the status and results of the tasks and their components. For example, the operator could print a work confirmation for a farmer.

6 Task-controller requirements

6.1 Task selection and execution

The task controller can provide a mechanism for the selection and shall provide a mechanism for the execution of a task contained in the data transfer file. Selection of an individual task can be done either by the operator through an operator interface or automatically by the task controller. The means of task selection is not specified by this part of ISO 11783. The task-controller designer is free to decide how task selection is implemented. The method provided for starting and stopping a task is not subject to standardization. This functionality is also to be determined by the task-controller designer.

On a MICS, a task shall always be selected. If no task was selected by the user, then the task controller shall prompt the operator for a task selection or select a task automatically.

The status of a task is defined in Table 1.

Table 1 — Task status

Initial	Task is prepared at FMIS but not yet processed on MICS.
Running	Task is currently being processed on MICS. Only one task can be active per task controller at the same time.
Paused	Task was previously running, is not currently running, and is not yet completed.
Completed[a]	Task is finished. This state can only be set by the operator and cannot be set automatically by the MICS.

[a] The *completed* task state is optional. Some task controllers may not support this state.

6.2 Logging of time and position

There can be a need to assign some optional information, such as date, time and GPS position data, to events that occur during task processing. This could be an event reflecting the interconnection of XML elements, such as the allocation of GPS position-related comments and/or flags or the assignment or exchange of a worker, for example. Other events are based on logging of received process data variable values from task-related working sets, which shall be supplied with information of date, time and position.

The XML elements AllocationStamp and Time enable the allocation of local time and date values to several XML elements where needed. These XML elements can be of the type planned or effective, for specifying whether an event was planned or accomplished. Additionally, at a task level within the Time XML element only, more detailed time types, such as preparation time, ineffective time, repair time or clearing time, are defined. All time and date values shall be local time and date values inside this XML element.

Optionally, the AllocationStamp and Time XML elements can include a position XML element that contains GPS-related information. To be able to allocate multiple process data values, the Time XML element can include several DataLogValue XML elements. The number of these DataLogValues stored inside the task shall be limited to totals or single instance values. For large amounts of DataLogValues, the use of the TimeLog XML element and binary log file is defined.

All process data-related logging data can be stored in a binary format as a separate file. The reference to the binary file is to be set by XML element TimeLog. More TimeLog XML elements can exist per task, referring to external files with unique names inside the name space of the set of tasks belonging to the data transfer file. The uniqueness of file name prefixes shall be guaranteed by the task controller. Each TimeLog file definition results in two separate files — one to contain the binary data and the other to contain the XML coded header structure of the binary data set. The header structure defines the maximum data per binary record and enables the correct interpretation of the binary data. The filename extension of the binary file shall be ".bin"; the filename extension of the XML header structure file shall be ".xml".

6.3 Logging parameters from parameter groups

In addition to the ProcessDataVariable values, the values or parameters from other parameter groups can be logged by a task controller. The XML elements DataLogTrigger and DataLogValue contain attributes to specify from which parameter group a value is to be logged. These attributes are optional. When these attributes are specified, the DataLogDDI attribute in DataLogTrigger or DataLogValue shall be set to the ParameterGroupNumberValue (DDI = $0xDFFE_{16}$). Each parameter group may contain multiple values, and therefore both a parameter group number and a start and stop bit to obtain a single value from the data field of the CAN data frame shall be specified when the task controller logs data from parameter groups other than the ProcessDataVariable. The size of the value is at a maximum 32 bits, and is stored as the DataLogValue attribute in the XML element DataLogValue.

When the task controller logs data from parameter groups, a reference to a DeviceElement is required. If this log data originates from parameter groups other than the ProcessDataVariable, a device with a control function NAME with a filled-in WorkingSetMasterNAME XML attribute and a DeviceElement referring to this device shall be defined. These device and DeviceElement XML elements may be generated by the task controller or can be supplied by the FMIS.

6.4 Connection management

Upon power-up, a specific sequence of events shall occur, in order to ensure proper initialization of the task controller and working sets, as described in 6.4.1 and 6.4.2 and depicted in Figure 2.

6.4.1 Task-controller initialization

The task controller shall complete the following initialization.

a) The task controller shall complete the address claim procedure in accordance with ISO 11783-5 and shall also send a request for an address claimed to the global destination address (255).

b) The task controller shall wait for 6 s after completing the address claim procedure.

c) The task controller shall begin transmission of the task-controller status message.

d) The task controller shall allow working sets to initialize and to load their device description objects.

6.4.2 Working-set initialization with the task controller

The working set shall complete the following initialization.

a) The working set shall complete the address claim procedure in accordance with ISO 11783-5.

b) The working-set master shall wait for 6 s after completing the address claim procedure.

c) The working-set master shall wait until the task controller begins transmission of the task-controller status message.

d) The working-set master shall identify itself and its members to the task controller, using the working-set master and working-set member messages given in ISO 11783-7.

   The working-set master may send these messages for other purposes (e.g. virtual terminal initialization).

e) The working-set master shall begin transmission of the working-set task message.

f) The working-set master may query the task controller, as necessary, to determine its capabilities.

g) The working-set master may request the language and format messages from a virtual terminal.

h) The working-set master shall query the task controller to determine if its device description object pool already exists.

i) The working-set master shall either

   1) activate the existing device description at the task controller, or

   2) commence and complete a transfer of the device description object pool to the task controller and activate the device description at the task controller, accomplished using the messages defined in Annex B and either the transport protocol (see ISO 11783-3) or extended transport protocol (see ISO 11783-6), depending on the size of the object pool.

6.5 Connection management

The task controller shall transmit a cyclic task controller status message at an interval of 2 s. The task controller also sends a task controller status message immediately whenever the task status changes or the value in any of the other task controller status message bytes changes, although at least 200 ms shall elapse between task controller status messages (the maximum transmit rate for task controller status message is 5 Hz). This message includes an indication of the current task status and is sent to the destination-specific global address (GA). If the working-set master does not receive this message for at least 6 s, it assumes a possible uncontrolled shutdown of the task controller and stops sending the working set task message. The working set may re-establish connection to the task controller by restarting the initialization procedure.

All working-set masters that maintain a connection with a task controller shall indicate their presence by transmitting to the task controller a cyclic working-set task message at an interval of 2 s. The working-set master shall wait at least 6 s after finishing the address claim procedure before transmitting the working-set task message. This timeout enables the task controller to detect a restart of a working set. If the task controller does not receive this message for at least 6 s, it assumes a possible uncontrolled shutdown of the working set.

The task controller status message and the working-set task message are defined in Annex B.

When a working set is restarted, or starts and connects to the task controller during an active task, the task controller shall accept the upload and activation of the working sets device descriptor and issue the measurement commands applicable to this working set.

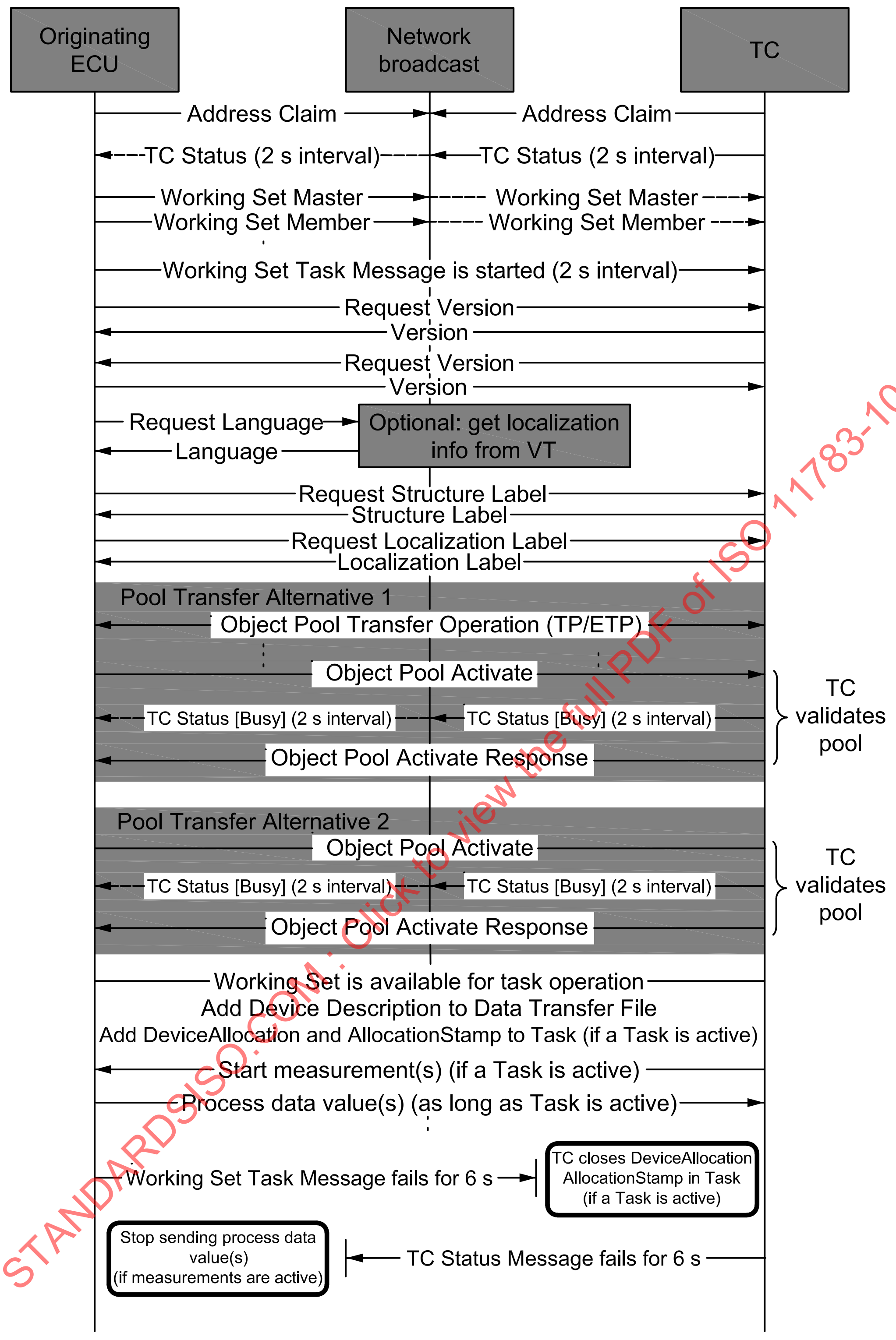

Originating ECU
Network broadcast
TC
Address Claim
Address Claim
TC Status (2 s interval)
TC Status (2 s interval)
Working Set Master
Working Set Member
Working Set Master
Working Set Member
Working Set Task Message is started (2 s interval)
Request Version
Version
Request Version
Version
Request Language
Language
Optional: get localization info from VT
Request Structure Label
Structure Label
Request Localization Label
Localization Label
Pool Transfer Alternative 1
Object Pool Transfer Operation (TP/ETP)
Object Pool Activate
TC Status [Busy] (2 s interval)
TC Status [Busy] (2 s interval)
Object Pool Activate Response
TC validates pool
Pool Transfer Alternative 2
Object Pool Activate
TC Status [Busy] (2 s interval)
TC Status [Busy] (2 s interval)
Object Pool Activate Response
TC validates pool
Working Set is available for task operation
Add Device Description to Data Transfer File
Add DeviceAllocation and AllocationStamp to Task (if a Task is active)
Start measurement(s) (if a Task is active)
Process data value(s) (as long as Task is active)
Working Set Task Message fails for 6 s
TC closes DeviceAllocation AllocationStamp in Task (if a Task is active)
Stop sending process data value(s) (if measurements are active)
TC Status Message fails for 6 s

Figure 2 — Initialization and shutdown of connections

6.6 Data exchange on the network

The task controller converts data from the data transfer file into process data messages to control the devices. These process data messages contain commands and values sent to the participating working-set control functions. The task controller performs calculations to schedule the process data messages to be transmitted to the required addresses on the ISO 11783 network. An example of these calculations are site-specific applications such as when the task controller looks up the position of an element of the working set in the application grid, combines it with the operation delays of this working set, and sends the relevant data to this working set. In the opposite direction, the task controller processes the process data messages sent by the participating working-set members and converts these process data variables to task data, in order to return these to the data transfer file.

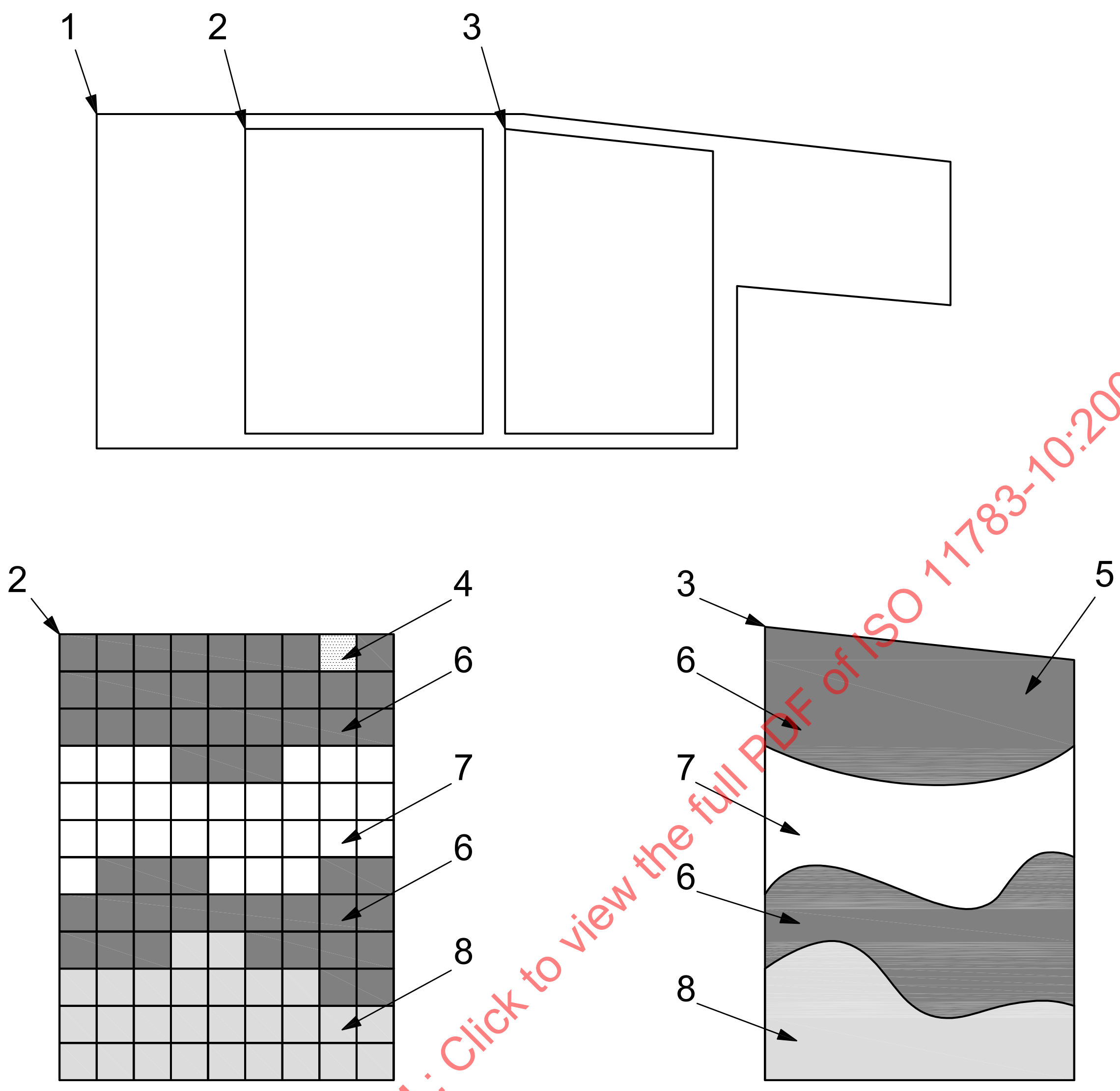
All task-controller-specific data exchange on the ISO 11783 network is based on process data messages.

The task controller can generate process data messages containing process data variables that are not specified in a task data file. The process data messages control operation and data logging of the participating working-set control functions. The task controller shall only send or request process data variables that are supported by the working-set control functions. Examples of this type of control are the use of sensor systems and the use of recorded spatial operation of working set control functions to control working set control functions.

6.6.1 Site-specific application

Site-specific applications require the task controller to schedule sending of process data messages according to the actual location. For matching of the actual location with ProcessDataVariable XML element definitions, the geometry for site-specific process data has to be specified in the task data. The geometry definition is either a gridcell or a polygon, and shall be labelled with a unique identification. Gridcells and polygons refer to a TreatmentZone to which the site-specific process data variable values are related. When the relevant DeviceElement enters a new TreatmentZone, the new set point values associated with that TreatmentZone are sent over the ISO 11783 network to the appropriate working-set master.

Of the geometry definitions, gridcells have constant length and width dimensions. The gridcell location is relative to the origin of the grid to which a gridcell belongs. The structure and identification of gridcells are specified in the XML elements Grid and Gridcell. Polygons can be used to define irregularly shaped TreatmentZones. The XML elements Polygon, Linestring and Point are used to define this type of TreatmentZone. Figure 3 presents a comparison of both types of TreatmentZone definition and Figure 4 the grid definition in more detail.

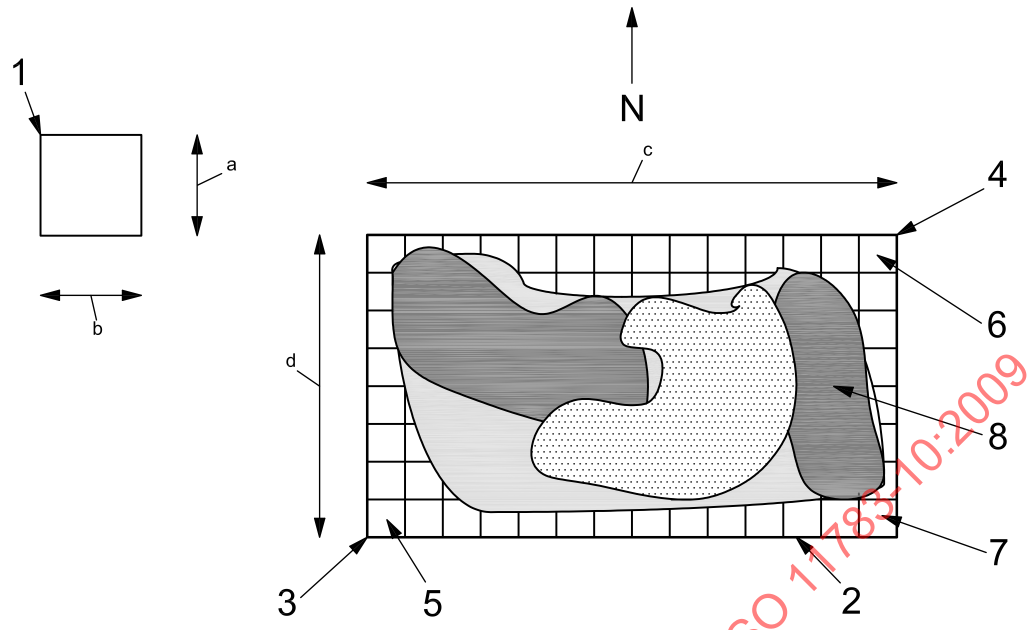


Key

1	frame partfield
2	partfield A
3	partfield B
4	gridcell
5	polygon
6	treatment zone 1
7	treatment zone 2
8	treatment zone 3

Figure 3 — Frame partfield, partfield, gridcell, polygon, treatment zone relationship

Inside the XML data definitions, a Grid shall always be filled completely with Gridcells. This means that there is no information like index, column and row defined for a gridcell. The Grid itself is defined by the GridMaximumColumns and the GridMaximumRows attributes, the GridMinimumNorthPosition and GridMinimumEastPosition, and the size of each Gridcell. The Gridcell ordering inside the hierarchical XML structure always starts with the GridMinimumNorthPosition/GridMinimumEastPosition, in ascending order of the columns (moving east), then proceeding in ascending order (moving north) on each row, beginning each time at the GridMinimumEastPosition column for each row. Figure 4 shows an example of a Grid and its Gridcells.



Key

1 gridcell
2 frame partfield
3 origin (GridMinimumNorthPosition, GridMinimumEastPosition)
4 FrameNorthMax, FrameEastMax
5 first grid cell (row 0 and column 0)
6 last grid cell (row 7 and column 13)
7 fourteenth grid cell (row 0 and column 13)
8 treatment zone

[a] GridCellNorthSize.
[b] GridCellEastSize.
[c] GridColumns.
[d] GridRows.

Figure 4 — Definition of frame of grid for partfield

When using overlaying polygons, the task controller shall always use the exterior polygon definition for a given location. Each interior boundary describes a hole in the exterior polygon.

Gridcell definitions data can optionally be stored in a task-specific manner in binary format as separate files. The reference to the separate file is to be set inside the XML element Grid. There can only be a single Grid XML element per task, referring to both a binary and an XML file of a unique name prefix for all tasks of the data transfer file. The uniqueness of grid file names shall be guaranteed by the FMIS. Each grid file definition results in two separate files — one to contain the binary data and the other the XML coded header structure of the binary data set. The header structure defines the maximum data per binary record and enables the correct interpretation of the binary data. The binary file suffix is to be set to “.bin”; the suffix of the XML header structure file is to be set to “.xml”.

6.6.2 Data logging

The task controller can also be used to log data for transfer back to the FMIS. For this purpose, process data variable values can be requested on the ISO 11783 network. Data log triggers can be specified by the FMIS in the data transfer file to the task controller. The task controller is responsible for converting these data log triggers into process data measurement commands, and for logging the replied process data variable values. The process data measurement commands are sent by the task controller after starting and resuming a task. The actual values for the requested process data variables are provided by the involved working-set members, as long as the task is active. The sending of actual values is stopped and measurements from the task controller are cancelled when the task is paused. When a working set sends more data than requested by the task controller for data logging, these additional data shall be ignored by the task controller. The description of the XML element DataLogTrigger in Annex D provides more detail on the use of data log triggers.

A data log trigger with a data log method of type total shall be used to request the task controller to store totals. Each total shall be stored once in the Time XML element in a task. In addition to this, total values can also be stored more frequently in the TimeLog related data log files. When a task is resumed, the task controller shall send the totals stored in the Time XML element to the working set to continue counting from these total values. The total values may be requested by the task controller from the working set to obtain the latest total values when the task is stopped. Alternatively, when process data variables that represent total values are transferred as process data messages to the task controller cyclically, such a query might not be necessary. Device process data objects with a data log trigger method of type total shall be settable.

Definition of total behaviour regarding Start/Resume/Pause task events:

a) Task started: the working set resets its totals to zero.

b) Task paused: the task controller collects totals from all working sets, and all totals retain their values.

c) Task resumed: this is to be handled similarly to the Start event for working sets. After the task has been restarted, the task controller sets all totals to the previously stored total values, sends out these values as process data variable values to all appropriate working-set masters and the working sets count from these values onwards. (The task controller is responsible for keeping track of previously collected totals.)

The task controller status message indicating the change of state of the totals' active bit from inactive to active shall be sent before the total values are set. The task controller status message indicating the change of state of the totals' active bit from active to inactive shall be sent before final total values are requested.

Each process data variable of a certain device might be asked for, and can be logged, by the task controller. The amount and types of process data variables to be logged are specified by the XML element DataLogTrigger. The DataLogTrigger XML element can specify precisely which process data variables are required from certain device elements. Alternatively, for a default data logging mechanism, a process data variable called RequestDefaultProcessData (DDI = $DFFF_{16}$) is specified. By using the identifier (DDI) of this process data variable in the DataLogTrigger XML element, the task controller is commanded to request the specified devices to send all their default process data variable values from their default measurement method. The RequestDefaultProcessData DDI may only be requested from device element 0 of a device. The task controller uses a request value command with DDI = $DFFF_{16}$ to request the sending of the default process data variables values. The set of process data variable values that belong to the default set of a device are specified in the device description. When the device description specifies certain DeviceProcessData objects to be part of the default set, a DeviceProcessData object with the RequestDefaultProcessData DDI shall be included in the device description of that device. The ProcessDataTriggerMethod attribute of this DeviceProcessData object shall be set to $1F_{16}$.

EXAMPLE 1 Device description containing the RequestDefaultProcessData DDI and a set of default data:

```
<DVC A="DVC1" B="Tiller" C="1.02*" D="A00484000B2CAF13" F="32A0FE34A56F00" G="FF000000006E65">
    <DET A="DET1" B="1" C="1" E="0" F="0">
        <DOR A="2"/>
        <DOR A="3"/>
        <DOR A="4"/>
        <DOR A="5"/>
        <DOR A="6"/>
    </DET>
    <DPD A="2" B="74" C="2" D="16" E="Area" F="9"/>
    <DPD A="3" B="77" C="2" D="16" E="Time ON" F="7"/>
    <DPD A="4" B="8D" C="1" D="3" E="Work ON/OFF"/>
    <DPD A="5" B="43" C="1" D="3" E="Width" F="8"/>
    <DPD A="6" B="DFFF" C="0" D="31"/>
    <DVP A="7" B="0" C="2.777778E-04" D="2" E="hr"/>
    <DVP A="8" B="0" C="1.000000E-03" D="1" E="m"/>
    <DVP A="9" B="0" C="1.000000E-04" D="2" E="ha"/>
</DVC>
```

In this example, the process data DDIs $8D_{16}$ and 43_{16} are part of the default set of data. This device supports the request default process data mechanism which is indicated by the DPD with object ID 6.

EXAMPLE 2 RequestDefaultProcessData DataLogTrigger in a task in a data transfer file:

```
<TSK A="TSK1" E="PFD1" G="3">
    <DLT A="DFFF" B="31"/>
</TSK>
```

In this example, the task specifies the requesting of default data from connected working sets that support the request default process data mechanism.

During data logging, there can be different frequencies of transmission of DDIs and also different time delays of replies to value request commands. The task controller logs process data variable values at certain time intervals, and therefore the following rules apply.

a) Time and position data shall be logged at a maximum once per Time or TimeLog instance.

b) Each process data variable value shall be logged at a maximum once per Time or TimeLog instance. If data logging can only be performed at a low frequency because of task-controller capabilities, the value logged depends on the task-controller design.

c) If the task controller is capable of data logging at the rate given by DDI with the highest update frequency, then a new data log record shall be started each time the next DDI with highest update frequency is received. All other process data values that are received between two values of the highest update frequency DDI are logged in the current record.

d) A received process data value shall be logged a maximum of once. If no new value is available for the next record then no value shall be logged for that DDI.

Following these rules, logged data values in the same record can be separated by, at a maximum, the time difference that exists between two records. If several DDIs need to be grouped together, the process data variable “Log Count” (DDI = 0093_{16}) shall be used with the highest repetition rate to time tag the data logging of associated process data variable values.

EXAMPLE 3 Process data variable DDI “A” has the highest update frequency, process data variable DDIs “B” and “C” are added to current DataLog:

Working set sends value of DDI A_1: task controller stores value of A_1 in DataLog 1

Working set sends value of DDI B_1: task controller adds value of B_1 to DataLog 1

Working set sends value of DDI A_2: task controller closes DataLog 1; stores value of A_2 in DataLog 2

Working set sends value of DDI C_1: task controller adds value of C_1 to DataLog 2

Working set sends value of DDI A_3: task controller closes DataLog 2; stores value of A_3 in DataLog 3

Working set sends value of DDI A_4: task controller closes DataLog 3; stores value of A_4 in DataLog 4

EXAMPLE 4 Working set desires to group process data variable DDIs: “A”, “B”, “C” and “D”.

Working set sends value of DDI (LogCount) LC_1: task controller stores value of DDI LC_1 in DataLog 1

Working set sends value of DDI A_1: task controller adds value of A_1 to DataLog 1

Working set sends value of DDI B_1: task controller adds value of B_1 to DataLog 1

Working set sends value of DDI C_1: task controller adds value of C_1 to DataLog 1

Working set sends value of DDI D_1: task controller adds value of D_1 to DataLog 1

Working set sends value of DDI LC_2: task controller stores value of DDI LC_2 in DataLog 2

Working set 1 sends value of DDI A_2: task controller adds value of A_2 to DataLog 2

Working set 1 sends value of DDI D_2: task controller adds value of D_2 to DataLog 2

Working set 1 sends value of DDI B_2: task controller adds value of B_2 to DataLog 2

Working set 1 sends value of DDI C_2: task controller adds value of C_2 to DataLog 2

Working set sends value of DDI LC_3: task controller stores value of DDI LC_3 in DataLog 3

Working set 1 sends value of DDI D_3: task controller adds value of D_3 to DataLog 3

Working set 1 sends value of DDI B_3: task controller adds value of B_3 to DataLog 3

Working set 1 sends value of DDI C_3: task controller adds value of C_3 to DataLog 3

Working set 1 sends value of DDI A_3: task controller adds value of A_3 to DataLog 3

The FMIS can determine, based on the LogCounts, that DDIs A, B, C and D all belong to the same group. There will always be a new DataLog started before the group is transmitted.

7 Data transfer

7.1 General

The communication FMIS to MICS to FMIS is based on data transfer files. Each XML data transfer file is formatted according to XML definitions version 1.0. The XML files contain text only and are coded in UTF-8 (see Reference [1]). Optionally, binary-coded data files for gridcell definitions or logged process data can be part of the data transfer files set. All files shall be contained in the same directory.

Both coding data and task data are stored in the same set of XML files upon transmission from FMIS to the MICS. During processing of tasks by the task controller, these files are likely to be modified and, when tasks are finished, can be transferred back to the FMIS.

7.2 Extensible Markup Language

XML is a language for the description of structured documents and data and represents a technological basis for data exchange. XML is a subset of SGML (standardized general markup language).

XML is a hierarchical set of XML elements. XML elements can contain one or more XML attributes. Inside the data transfer XML files, no text is allowed inside XML elements.

An XML element consists at least of an opening label, a number of attributes and a closing label. The following line is an example of an XML element definition called *Worker*:

<Worker WorkerId="WKR0002" WorkerDesignator="Miller">

</Worker>

Opening and closing labels shall be of the same string, except that the closing label shall be preceded by a "/". The case is significant in all uses of XML element labels, i.e. <worker> is not the same as <Worker>.

XML attributes carry information contained in XML elements. The syntax of an XML attribute is *attribute-name*="*description*".

The XML attribute label follows the same rules as an XML element label. The case is significant. The use of the equals sign is always required. The text string description shall be enclosed in quotes.

XML elements that do not contain child elements, but contain attributes, may be represented in a shortened form:

<Worker WorkerId="WKR0001" WorkerDesignator="Smith" />

If an XML file follows the rules of the XML syntax itself, it is said to be *well formed*. The elements of an XML document can be formally specified using a DTD. If a DTD exists for an XML document, XML parsers with the appropriate extensions can test the XML document against the DTD. A document that passes such a consistency and validation test is said to be *valid*.

XML data transfer files are always text files. Due to the hierarchical structure, XML files can only contain a single XML root element. The elements list of the root element specifies which XML elements might be used as primary elements. Primary elements consist of coding data elements and entities which are not related to a single task. The root element of a data transfer file is named as ISO11783_TaskData. Data transfer files shall always be well formed and valid, otherwise the contents of the file cannot be processed.

7.3 Extensible schema definition

XML provides an application-independent way of sharing and transferring data. With a DTD, it is possible to verify that received data from the outside world is valid. Also, a DTD might be used to verify its own created data.

The purpose of a DTD is to define the legal building blocks of an XML document. It defines the document structure with a list of legal XML elements. A DTD can be declared inline in an XML document, or as an external reference. An object-oriented language modelling approach is to use an XSD as DTD.

An XML schema

— defines elements that can appear in a document,

— defines attributes that can appear in a document,

— defines which elements are child elements,

— defines the order of child elements,

— defines the number of child elements,

— defines whether an element is empty or can include text,

— defines data types for elements and attributes, and

— defines default and fixed values for elements and attributes.

XML schemas are formatted in XML and well formed. A well-formed XML document is a document that conforms to the XML syntax rules:

a) it shall begin with the XML declaration;

b) it shall have one unique root element;

c) all start tags shall match end tags;

d) XML tags are case-sensitive;

e) all elements shall be closed;

f) all elements shall be properly nested;

g) all attribute values shall be quoted.

Schemas support data types and namespaces. With the support for data types it is possible to describe permissible document content, validate the correctness of data, work with data from a database, define data facets (restrictions on data) and data patterns (data formats), and convert data between different data types.

7.4 XML schema definition

The validity of data transfer files is defined by the schema ISO11783_TaskFile. The schema is based on http://www.w3.org/2001/XMLSchema definitions and data types. Unique identifiers of XML elements and references to unique identifiers across XML elements are defined as data types ID and IDREF respectively. To create a unique namespace all over the data transfer file, each XML element identifier shall be defined in its specific XML element namespace. Table 2 defines the valid namespaces of XML element identifiers.

The XML schema definition is to be applied for all data transfer files in both data transfer directions. All XML elements are considered as entities. Each entity shall have a unique identifier (if there is an identifier defined as the XML attribute for this element) over the whole data transfer file. Each identifier starts with the appropriate namespace letters followed by, at a maximum, an 11-digit decimal number without leading zeroes. Unique identifiers are at least 4 bytes long and shall be provided by both the FMIS and the MICS. The value of –2147483648 for identifiers is reserved and shall not be used inside XML elements.

On the MICS, no coding data XML element shall be changed or deleted. Non-coding data XML elements may be edited and changed. If the operator of the task controller creates new entities of XML elements, then the task controller is responsible for creating unique identifiers for all newly created entities of XML elements. To differentiate between those entities provided by the FMIS and those created on the mobile system, each newly created identifier shall have the appropriate namespace letters followed by a negative decimal number, e.g. “WKR–1”.

Table 2 — XML element types and acronyms

XML element designator	Coding data	XML element name
AllocationStamp		ASP
CodedComment	X	CCT
CodedCommentGroup	X	CCG
CodedCommentListValue	X	CCL
ColourLegend	X	CLD
ColourRange	X	CRG
CommentAllocation		CAN
Connection		CNN
CropType	X	CTP
CropVariety	X	CVT
CulturalPractice	X	CPC
Customer	X	CTR
DataLogTrigger		DLT
DataLogValue		DLV
Device	X	DVC
DeviceAllocation		DAN
DeviceElement	X	DET
DeviceObjectReference	X	DOR
DeviceProcessData	X	DPD
DeviceProperty	X	DPT
DeviceValuePresentation	X	DVP
Farm	X	FRM
Grid		GRD
Linestring		LSG
OperationTechnique	X	OTQ
OperationTechniqueReference	X	OTR
OperTechPractice		OTP
Partfield	X	PFD
Point		PNT
Polygon		PLN
Position		PTN
ProcessDataVariable		PDV
Product	X	PDT
ProductAllocation		PAN
ProductGroup	X	PGP
Task		TSK
Time		TIM
TimeLog		TLG
TreatmentZone		TZN
ValuePresentation	X	VPN
Worker	X	WKR
WorkerAllocation		WAN
ExternalFileContents		XFC
ExternalFileReference		XFR

Any XML attribute of type IDREF may only contain a single identifier reference.

Manufacturer proprietary XML elements and XML attributes in the data transfer shall be preceded by a string consisting of a character “P”, the decimal manufacturer code and an underscore character “_”.

EXAMPLE 1 A proprietary XML element designated “MyProprietaryElement” in the data transfer file of a manufacturer with manufacturer code 500 receives the XML element name:

“P500_MyProprietaryElement”.

EXAMPLE 2 A proprietary XML attribute designated “MyProprietaryAttribute” from the same manufacturer receives the XML attribute name:

“P500_MyProprietaryAttribute”.

The contents of the manufacturer proprietary XML elements and XML attributes shall be ignored when the data transfer file is processed by a MICS or FMIS of another manufacturer. On the MICS, when the data transfer file originating from the FMIS contains proprietary XML elements or XML attributes, these XML tags and their values shall be preserved in the data transfer file originating from the MICS back to the FMIS.

7.5 XML data transfer files

The name of the main XML coded data transfer file containing the root XML element ISO11783_TaskData shall be set to “TASKDATA.XML”. The medium used to transfer the data between FMIS and MICS is proprietary (see Figure 1). When, for instance, a removable storage device is used to transfer the data transfer file, the recommended location of this file is in a directory named “TASKDATA”, located in the root directory of the transfer medium. Both this directory name and the data transfer file name are case-sensitive and all the characters are in upper case.

The main data transfer file contains coding data and a number of tasks. Figure 5 illustrates the structure of XML elements in the data transfer file. There can be references to other XML files containing coding data or tasks inside the main data transfer file. References to XML files are accomplished by use of the XML element XFR. XML file references cannot be nested. This means that only the main XML file can include XML elements XFR. There can be a single XML element in the referenced external XML file. The elements shall be embedded in an XML root element named XFC. A referenced XML file shall only contain top-level XML elements of the same type — for example, it may include definitions of customers but not of customers and of partfields together. Top-level XML elements may be defined both in the main and in the referenced XML files.

Each XML coded data transfer file shall start with an XML identification section and shall be well formed:

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

In the root data transfer file, all XML elements are always embedded in the global root structure:

```
<ISO11783_TaskData VersionMajor="..." VersionMinor="..." TaskControllerManufacturer="..."
    TaskControllerVersion="..." ManagementSoftwareManufacturer="..." ManagementSoftwareVersion="..."
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="… " DataTransferOrigin="...">

</ISO11783_TaskData>
```

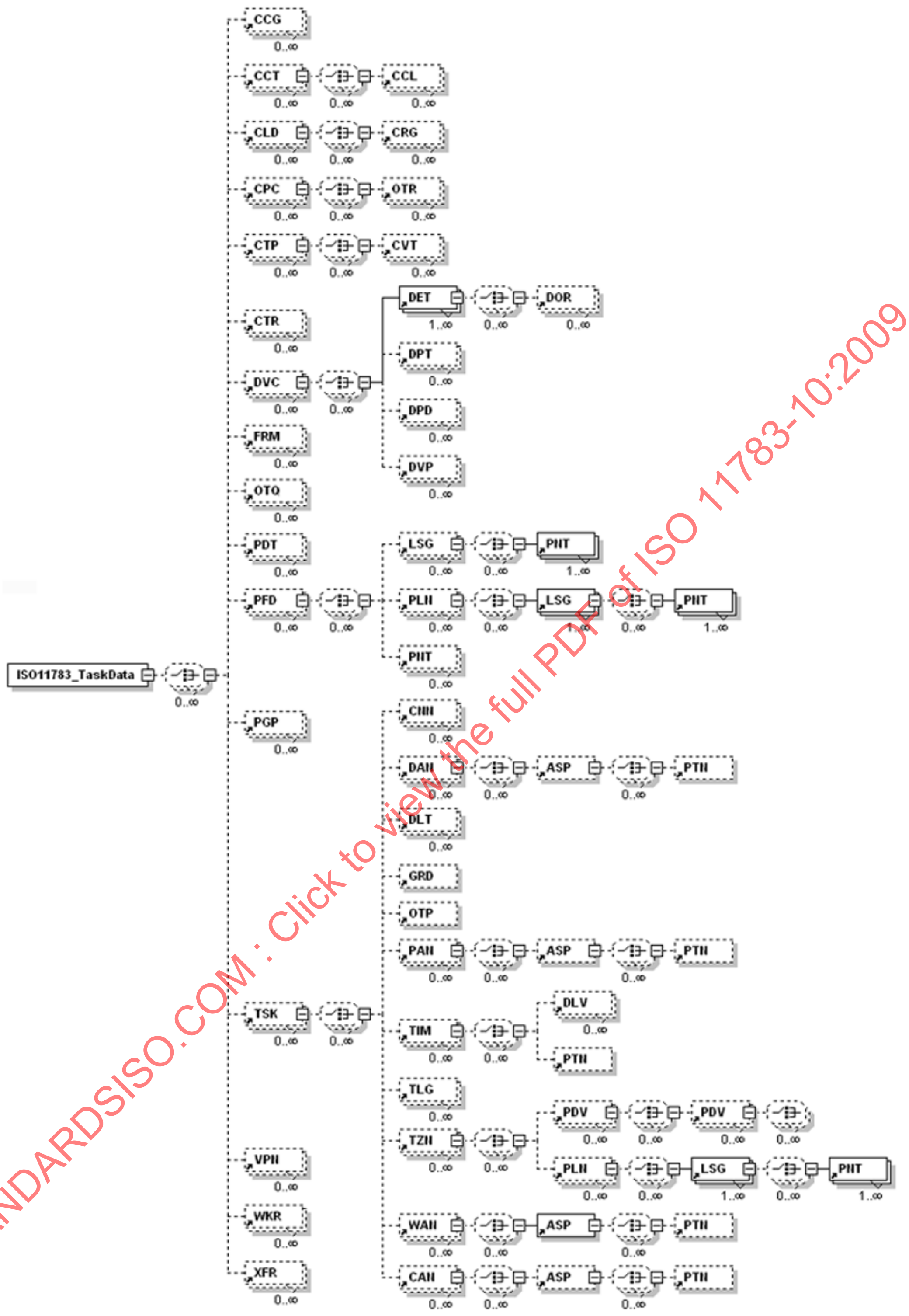


Figure 5 — Schematic structure of a data transfer file

Figure 6 shows the usage of referenced XML files.

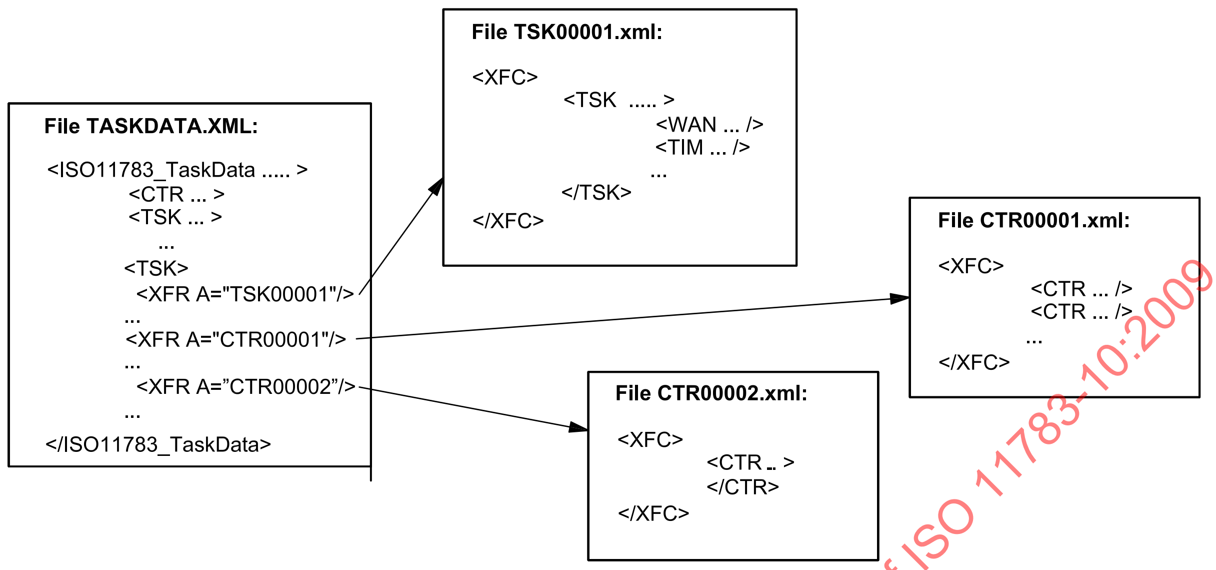


Figure 6 — XFR/XFC XML file references

The direction indication attribute in the XML header section (FMIS➔MICS or MICS➔FMIS) is to be handled and updated properly by both FMIS and MICS.

For an update of coding data, the MICS is allowed to add new instances of XML elements only. This means that modification of FMIS coding data can only be done by the FMIS.

The task controller or task controller configuration program shall be prepared to receive either a single XML data transfer file or set of XML data transfer files from the management information system. The management information system shall be prepared to receive either a single data transfer file or a set of XML data transfer files from the MICS, and both can refer to binary files.

7.6 Binary data transfer files

7.6.1 General

There are two types of binary file allowed to be used as part of the data transfer file set. One is to contain gridcell values. The second is to contain binary-coded log data. Both binary files belong to a task.

7.6.2 Grid binary file structure

The gridcells of a Grid can only be defined in a binary file. Two types of grid are supported: the first type of grid contains TreatmentZone codes; the second type of grid contains ProcessDataVariable values.

The first type is used when a limited number of TreatmentZones is defined and the grid functions as a lookup table to the TreatmentZones. In this case, each gridcell in the binary grid file contains a TreatmentZoneCode of the TreatmentZone to which that gridcell belongs. This grid type has at a maximum one value per gridcell, which is an unsigned 8-bit integer coded TreatmentZoneCode.

EXAMPLE 1 Grid that contains TreatmentZoneCodes:

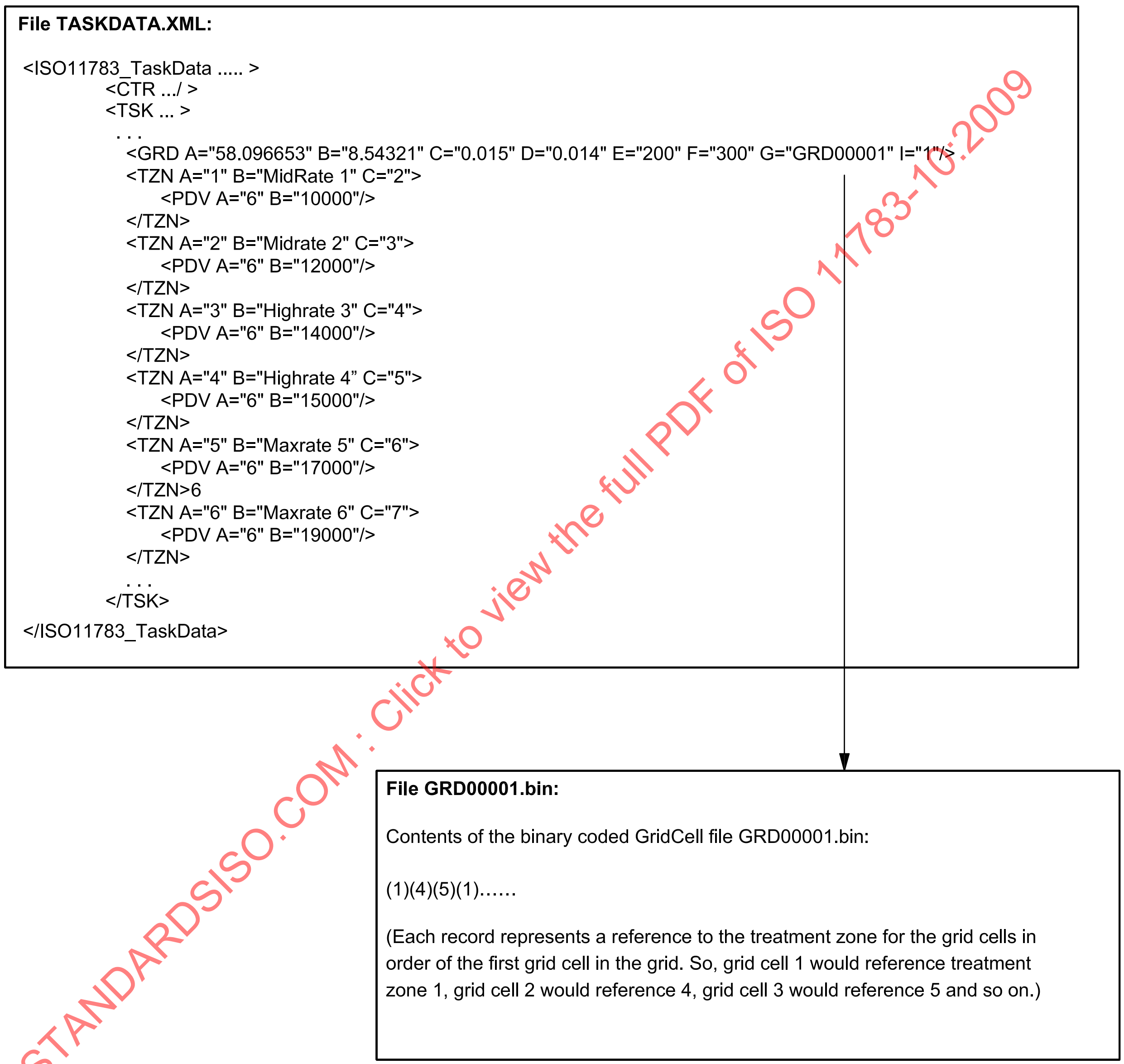
The task data XML file contains a grid definition with an empty TreatmentZoneIdRef attribute:

```
<GRD A="58.096653" B="8.54321" C="1.5E-3" D="1.4E-3" E="200" F="300" G="GRD00001" I="1"/>
```

The binary file contains: (TreatmentZoneCodes)

(1)(4)(3)(6)…

Each value inside brackets is a record and represents the binary-coded treatment zone code. The brackets are for better reading purposes in this example and are not part of the binary format. The relation between the binary file and the grid type 1 specification is shown in Figure 7.



File TASKDATA.XML:

```
<ISO11783_TaskData ..... >
        <CTR .../ >
        <TSK ... >
         . . .
          <GRD A="58.096653" B="8.54321" C="0.015" D="0.014" E="200" F="300" G="GRD00001" I="1"/>
          <TZN A="1" B="MidRate 1" C="2">
             <PDV A="6" B="10000"/>
          </TZN>
          <TZN A="2" B="Midrate 2" C="3">
             <PDV A="6" B="12000"/>
          </TZN>
          <TZN A="3" B="Highrate 3" C="4">
             <PDV A="6" B="14000"/>
          </TZN>
          <TZN A="4" B="Highrate 4" C="5">
             <PDV A="6" B="15000"/>
          </TZN>
          <TZN A="5" B="Maxrate 5" C="6">
             <PDV A="6" B="17000"/>
          </TZN>6
          <TZN A="6" B="Maxrate 6" C="7">
             <PDV A="6" B="19000"/>
          </TZN>
          . . .
        </TSK>
</ISO11783_TaskData>
```

File GRD00001.bin:

Contents of the binary coded GridCell file GRD00001.bin:

(1)(4)(5)(1)……

(Each record represents a reference to the treatment zone for the grid cells in order of the first grid cell in the grid. So, grid cell 1 would reference treatment zone 1, grid cell 2 would reference 4, grid cell 3 would reference 5 and so on.)

Figure 7 — Grid type 1 binary file example

The second type of grid is used when the ProcessDataValue attribute of the ProcessDataVariable is not classified into a limited number of TreatmentZones. In this case, one TreatmentZone, possibly with multiple ProcessDataVariables, is defined as a template for the binary grid file. There are no values of the XML attributes “ProcessDataValue” in this template definition in the XML file specified and, instead, these values are coded in the binary grid file as signed 32-bit integers. For byte ordering of the 32-bit integer data type, the same rules apply as defined in ISO 11783-6.

EXAMPLE 2 Grid that contains ProcessDataVariable ProcessDataValues

The task data XML file contains a Grid definition with a TreatmentZoneCode attribute:

```
<GRD A="58.096653" B="8.54321" C="1.5E-3" D="1.4E-3" E="200" F="300" G="GRD00001" I="2" J="6"/>
```

and a TreatmentZone prototype without ProcessDataValue attributes:

```
<TZN A="6" B="Precision Farming" C="2">

<PDV A="0001" B="0"/>

<PDV A="0006" B="0"/>

<PDV A="001A" B="0"/>

</TZN>
```

The binary file contains: (ProcessDataValue records [])

```
[(11100)(15000)(190)]

[(14000)(20000)(200)]

[(19000)(25000)(200)]……
```

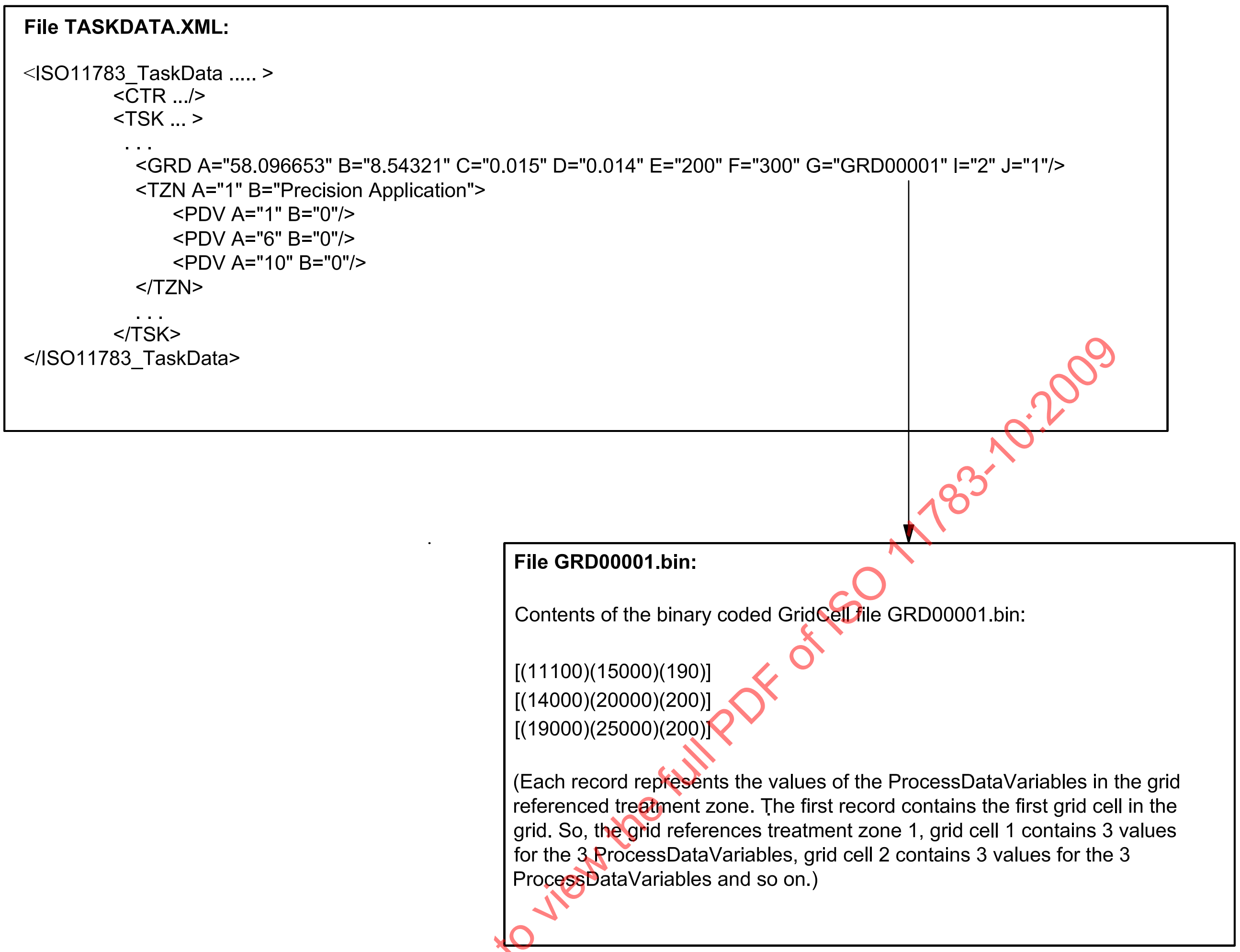
where the first record represents the first gridcell with the following ProcessDataValues:

```
<PDV A="0001" B="11100"/>

<PDV A="0006" B="15000"/>

<PDV A="001A" B="190"/>
```

Each value inside the square brackets in the binary file example is a record and represents the ProcessDataValues for one gridcell. The rounded brackets delimit the individual ProcessDataValue values when more than one ProcessDataVariable is present in the referenced TreatmentZone. The brackets are for better reading purposes in this example and are not part of the binary format. Figure 8 shows the binary file related to the grid type 2 specification.



File TASKDATA.XML:

```
<ISO11783_TaskData ..... >
          <CTR .../>
          <TSK ... >
           . . .
            <GRD A="58.096653" B="8.54321" C="0.015" D="0.014" E="200" F="300" G="GRD00001" I="2" J="1"/>
            <TZN A="1" B="Precision Application">
                <PDV A="1" B="0"/>
                <PDV A="6" B="0"/>
                <PDV A="10" B="0"/>
            </TZN>
            . . .
          </TSK>
</ISO11783_TaskData>
```

File GRD00001.bin:

Contents of the binary coded GridCell file GRD00001.bin:

```
[(11100)(15000)(190)]
[(14000)(20000)(200)]
[(19000)(25000)(200)]
```

(Each record represents the values of the ProcessDataVariables in the grid referenced treatment zone. The first record contains the first grid cell in the grid. So, the grid references treatment zone 1, grid cell 1 contains 3 values for the 3 ProcessDataVariables, grid cell 2 contains 3 values for the 3 ProcessDataVariables and so on.)

Figure 8 — Grid type 2 binary file example

7.6.3 Log data binary file structure

The TimeLog XML element is used in the task data file to refer to a binary log file. In the task data file, a TimeLog XML element without referenced Time, Position or DataLogValues shall be used. This TimeLog XML element refers to two files: a TimeLog XML header file and a binary log file. Inside the TimeLog XML header file, the XML elements shall include attributes without any values to define the record structure of the binary log file. All attributes with non-empty value definitions in the TimeLog XML header file contain fixed values which are valid for all binary-coded records in the binary log file. Only the values of the empty value attributes of the TimeLog XML header file are written in the binary log file. Inside the binary records, the IDREF value shall be of data type signed 32-bit integer; the namespace letters DET are used only inside the XML attribute values. An example of an XML-coded TimeLog file that specifies the structure of the binary TimeLog file is given by:

```
<TIM A="" D="4">

 <PTN A="" B="" D=""/>

 <DLV A="0815" B="" C="DET1"/>

 <DLV A="4711" B="" C="DET2"/>

 <DLV A="4522" B="" C="DET3"/>

</TIM>
```

TimeLog binary data formats are shown in Table 3.

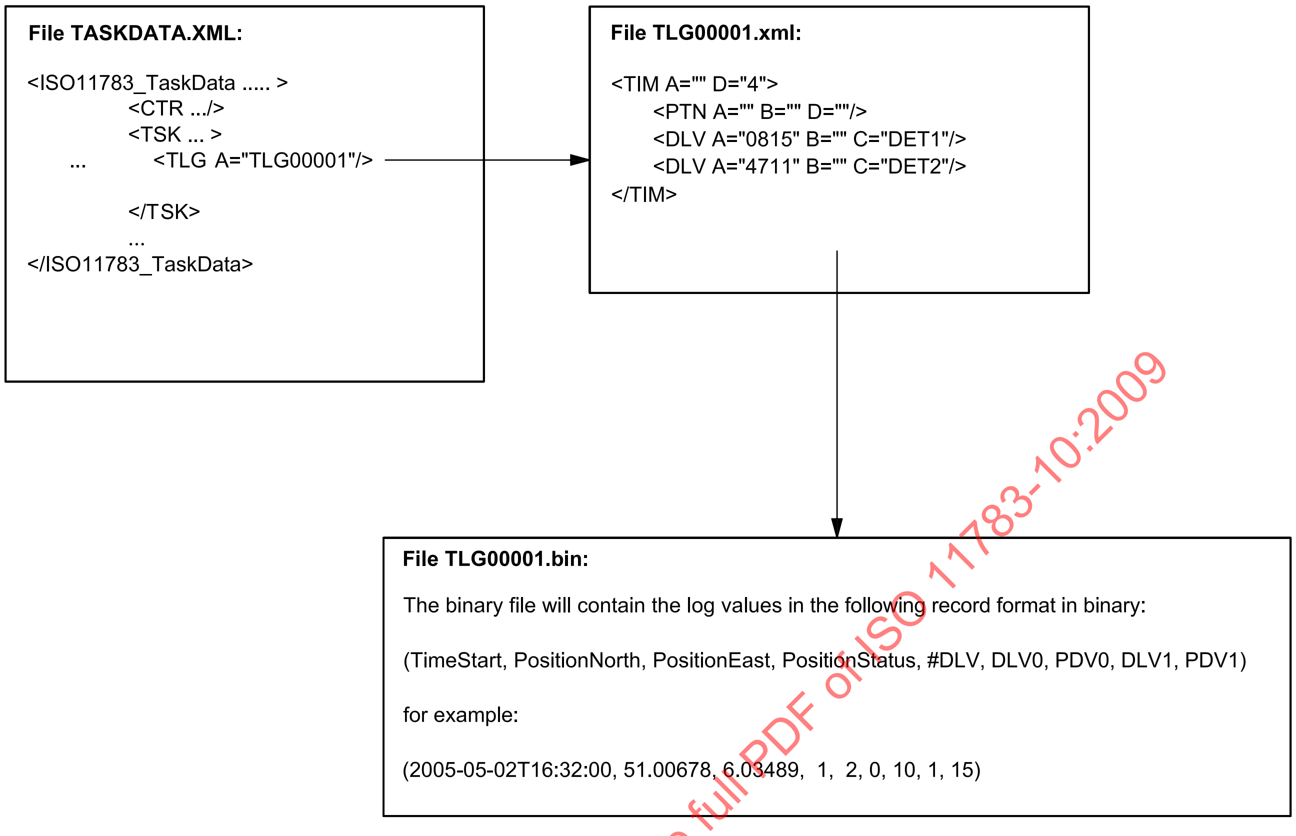
Table 3 — Binary file record value definitions

Value	XML reference	Binary data type	Definition
TimeStart: time of day	TIM, A	Unsigned 32-bit integer	Milliseconds since midnight
TimeStart: date	TIM, A	Unsigned 16-bit integer	Days since 1980-01-01
PositionNorth	PTN, A	32-bit integer	10^{-7} degrees WGS-84
PositionEast	PTN, B	32-bit integer	10^{-7} degrees WGS-84
PositionUp	PTN, C	32-bit integer	Millimetres relative to WGS-84 ellipsoid
PositionStatus	PTN, D	Byte	Position status. Definition references NMEA2000 MethodGNSS parameter. 0 = no GPS fix 1 = GNSS fix 2 = DGNSS fix 3 = Precise GNSS, no deliberate degradation (such as SA), and higher resolution code (P-code) and 2 frequencies are used to correct atmospheric delays 4 = RTK Fixed Integer 5 = RTK Float 6 = Est(DR)mode 7 = Manual Input 8 = Simulate mode 9-13 = Reserved 14 = Error 15 = PositionStatus value not available
PDOP	PTN, E	Unsigned 16-bit integer	10^{-1} PDOP quality information
HDOP	PTN, F	Unsigned 16-bit integer	10^{-1} HDOP quality information
NumberOfSatellites	PTN, G	Byte	Number of used satellites
GpsUtcTime	PTN, H	Unsigned 32-bit integer	UTC milliseconds since midnight
GpsUtcDate	PTN, I	Unsigned 16-bit integer	UTC days since 1980-01-01
#DLV		Byte	Number of PDV to follow
DLVn		Byte	Ordering number of PDV to follow, starting with 0 for first DataLogValue definition
ProcessDataValue		32-bit integer	According to DDI

All attributes that refer to any values are considered to be fixed values throughout all records of the binary file. All DLV elements are indexed and referred to in the binary records according to their definitions order. The number of DataLogValues actually stored is defined as #DLV. An example of the binary log data file is shown in Figure 9, in which there are a total of three DLVs. To allow a dynamic set of DLVs in the binary record, each DLV is identified by its ordering number, DLVn, of definition in the XML file. The example of an XML-coded TimeLog file given above, just before Table 3, specifies that all following records consist of the following set of values:

(TimeStart,PositionNorth,PositionEast,PositionStatus,#DLV,DLV0,PDV0,DLV1,PDV1,DLV2,PDV2)

This means that a time entry can have a maximum of 255 PDVs.



File TASKDATA.XML:

```
<ISO11783_TaskData ..... >
          <CTR .../>
          <TSK ... >
    ...      <TLG A="TLG00001"/>

          </TSK>
          ...
</ISO11783_TaskData>
```

File TLG00001.xml:

```
<TIM A="" D="4">
    <PTN A="" B="" D=""/>
    <DLV A="0815" B="" C="DET1"/>
    <DLV A="4711" B="" C="DET2"/>
</TIM>
```

File TLG00001.bin:

The binary file will contain the log values in the following record format in binary:

(TimeStart, PositionNorth, PositionEast, PositionStatus, #DLV, DLV0, PDV0, DLV1, PDV1)

for example:

(2005-05-02T16:32:00, 51.00678, 6.03489, 1, 2, 0, 10, 1, 15)

Figure 9 — Log data binary file example

7.7 Device description data

The purpose of device description data is to specify device-related properties on both the MICS and the FMIS. Device description data can be supplied by the device manufacturer as a set of XML definitions and can be integrated into the FMIS through a data import. Once the device description data is known at the FMIS, the description data can be used for task planning and included in the data transfer file from FMIS to MICS.

If in a separate file, the device description data shall begin with an XML identification section. Its contents shall follow the definitions of the data transfer file schema for devices. The used identifiers of XML elements shall be unique and will be transferred into the unique identifier namespace of the farm-management system by the FMIS. The device description data file has to contain all information about a device that is necessary when using it for a task. For example, a device description for a sprayer contains the geometry of the sections or nozzles, the number of tanks and their volumes, and supported process data variables. This information shall be specified by DeviceElement, DeviceProperty, and DeviceProcessData XML elements in the file.

The first element of DeviceElements represents the device itself and gets ElementNumber = 0. For this DeviceElement, the ParentIdRef refers to the XML element device. See the XSD for the description of allowed values. For all other (sub-)device elements on a device, the ParentIdRef can refer to a DeviceElement to describe a hierarchical structure, or to a device for device elements that do not have a sub-position in a hierarchy but belong directly to the device.

The device description data can originate from devices on the MICS. Device descriptions are transmitted to the task controller on the ISO 11783 network using the appropriate process data commands specified in Annex B. Thus, the task controller shall be able to receive the device description to determine whether a task can be executed as specified. The task controller shall also store the device description for transfer back to the FMIS. For this transfer of the device description data over the ISO 11783 network, a set of objects is defined. The objects containing the device description data are described in Annex A.

The procedure for transfer of the device description data over the network is the following.

a) The working-set master shall determine if the task controller has available memory by transmitting a request object-pool transfer message. The task controller acknowledges this message with a request object-pool transfer response message. The working-set master shall check the status code returned. If enough memory is available, the working-set master can proceed.

b) The working-set master uses the transport protocol (see ISO 11783-3) or extended transport protocol (see ISO 11783-6) to transmit the updated or new object(s) to the task controller. Normal handshaking, error checking and retransmission, in accordance with ISO 11783-3, shall be implemented.

c) Upon completion, the working-set master shall transmit an object-pool activate message to the task controller, to indicate that the update is now complete and ready for use.

d) The task controller shall respond with the object-pool activate response message.

If the working-set master needs to modify an object such that

— the length of the object will change,

— references to other objects will change, and/or

— multiple attributes of objects will change,

then the working-set master can update its pool at runtime. New objects can also be added to the object pool with this procedure. This is accomplished by using the same messages and procedures used to transfer the pool at initialization. That is to say, if the working-set master needs to modify a value presentation object due to a language or unit of measure change detected via the language command, it shall do so by transmitting an updated value of presentation objects to the task controller. This is accomplished by updating the device description object pool at runtime.

The primary intention for device description object-pool update at runtime is to allow working-set masters to update their device description objects when the user changes language and/or units of measure settings. This is to allow a task controller with a user interface to update any displayed information to appear in a language and units of measure format that the user prefers. No restrictions are made on when device description object-pool updates can take place; however, working-set master designers shall avoid, as much as possible, performing device description updates during those times when a task is active.

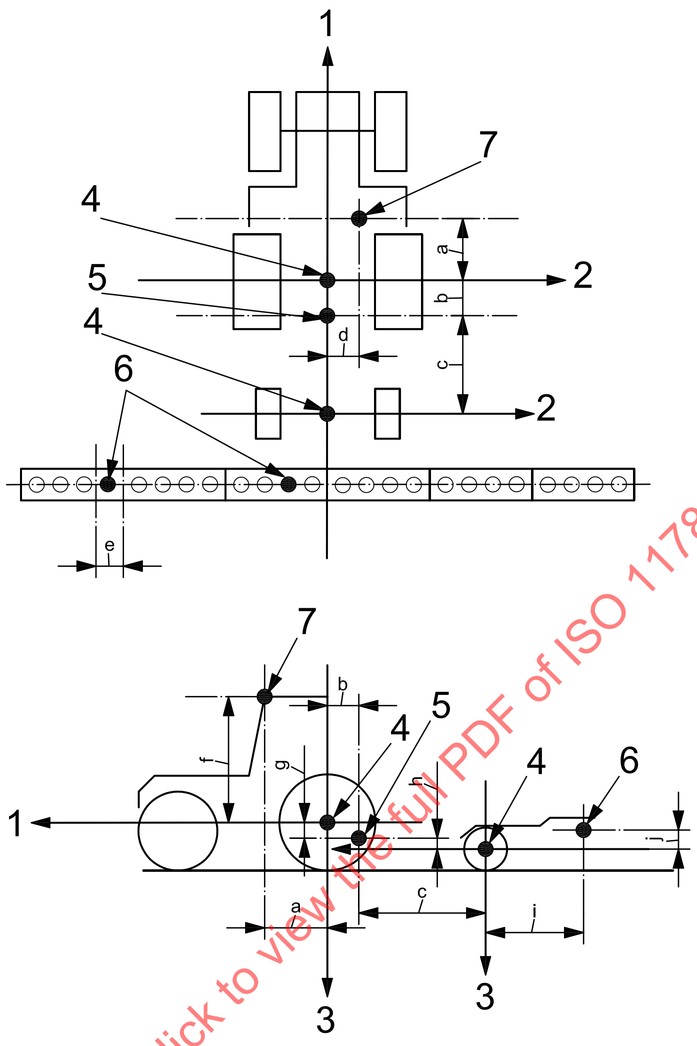
Only those objects that need to be changed shall be transmitted during the update; all other objects will remain in task-controller memory.

In the case of errors in the update to the device description object pool, the task controller shall indicate the errors with the object-pool activate response message. The task controller shall delete the entire device description object pool from volatile memory (including the object pool as it existed prior to the update), and may inform the operator, by an alarm-type method, of the suspension of the working-set master and the reason for the deletion.

For the geometry description of a device, the following information is needed: numbering and specification of DeviceElements; supported ProcessDataVariables; specification of DeviceProperties.

NOTE In order to obtain a general description, no difference is made between an implement, a sensor system or a tractor. The device description is usable both for implements and sensor systems with a GPS receiver and for a tractor with fixed DeviceElements.

Figure 10 shows the reference points in a connected system. All reference points are described by DeviceElements and the relative distances can be specified by process data variable values, defined in the data dictionary specified in ISO 11783-11.



Key

1 device X axis (positive direction)
2 device Y axis (positive direction)
3 device Z axis (positive direction)
4 DeviceReferencePoint (DRP)
5 ConnectorReferencePoint (CRP)
6 DeviceElementReferencePoint (ERP)
7 NavigationReferencePoint (NRP)

a	NRP_X.
b	CRP_X (Tractor).
c	CRP_X (Implement).
d	NRP_Y.
e	ElementWorkingWidth.
f	NRP_Z.
g	CRP_Z (Tractor).
h	CRP_Z (Implement).
i	ERP_X.
j	ERP_Z.

Figure 10 — Device geometry definitions

Each device has a coordinate system. The centre of the device coordinate system is defined as a DeviceReferencePoint with coordinates (0,0,0). The location of a DeviceElement inside a device is relative to the DRP and the coordinate system is a right-hand rule coordinate system with the following axis definitions.

— The X-axis is specified as positive in the normal driving direction.

— The Y-axis is specified as positive to the right side of the device relative to the normal driving direction.

— The Z-axis is specified as positive downward toward the ground plane.

For a tractor, the DRP is the centre of the rear axle. For a wheeled implement, the DRP is the centre of the front axle. In other cases, the DRP can be chosen freely, e.g. DRP=CRP or DRP=ERP. If there are angles on the device geometry, the device control function is responsible for recalculating the new DRP/CRP and sending this as dynamic data to the task controller.

The NavigationReferencePoint and ConnectorReferencePoints refer to the location of the DeviceElements that are of DeviceElementType “Navigation” and “Connection”. In case a NavigationReferencePoint is not reported by a tractor but is required by a task controller, the task controller can provide alternative means to specify the offsets of the navigation system's reference point.

DevicePropertyObjects shall be defined for a device and referenced by DeviceElementObjects to provide the values for DeviceElement offsets, time delays, capacities, etc.

The ConnectorReferencePoint specifies the position of one or more mounting facilities, e.g. rear and front three-point hitch or drawbar. For the three-point hitch, the CRP is the centre of the lower link points. If there are dynamic changes (e.g. by a steering axle of the device), the device has to calculate the new CRP and send it to the task controller.

The ERP is the centre of one (or of a group of) device element(s), i.e. the centre of a spraying boom.

Annex A
(normative)

Device description objects

A.1 General

Each device, whether machine or sensor system, is defined by the XML element “Device”, at least one XML element “DeviceElement” and the optional device-element-related XML elements “DeviceProcessData”, “DeviceProperty” and “ValuePresentation”. By defining these device descriptions of XML elements as objects, similar to the way in which objects are used inside an object-pool definition of ISO 11783-6, all device-related data can be transferred to the task controller by means of the ISO 11783 transport protocol and/or extended transport protocol. This annex describes the representation of the device description XML elements as binary objects. Note that several attributes in this representation are coded as UTF-8 strings. These strings do not have a preceding byte-order mark (BOM).

All object IDs shall be unique inside the whole device description object pool.

When no object is referenced, the NULL Object ID shall be used. The NULL Object ID has value 65535 ($FFFF_{16}$).

Data transfer of device description objects enables the task controller to get the actual description of a connected device. The received object definitions of a certain device are converted into XML elements and stored in XML data transfer files. If the object definitions are not part of the current task data file (as received from FMIS) then the new device shall be added to the task data file. If the same device exists in the current task data file then the software and description version label between both data sets shall be checked. It is up to the mobile system either to update the definitions inside the task data file by the newly received device description data, or store the received description objects as new device XML elements.

A.2 Definition of DeviceObject

DeviceObject is the object definition of the XML element device. Each device may have only a single DeviceObject in its device description object pool. See Table A.1.

Table A.1 — DeviceObject definition

Attribute Name	**Type**	**Size** bytes	**Range or value**	**Record byte**	**Description**
Table ID	String	3	DVC	1-3	XML element namespace for device.
Object ID	Integer	2	0	4-5	Unique object identifier inside this object pool. DeviceObject ID = 0.
Number of designator bytes (N)	Byte	1	0 .. 32	6	Length of following designator UTF-8 string.
Device designator	UTF-8 string, no BOM	0 .. 32		7	Descriptive text to identify this device. This text can be displayed to the operator.
Number of software version bytes (M)	Byte	1	0 .. 32	7+N	Length of following software version UTF-8 string.
Device software version	UTF-8 string, no BOM	0 .. 32		8+N	Software version indicating text.
WorkingSet MasterNAME	Double integer	8		8+N+M	NAME of working-set master. The NAME structure is defined in ISO 11783-5
Number of DeviceSerial-Numberbytes (O)	Byte	1	0 .. 32	16+N+M	Length of following DeviceSerialNumber UTF-8 string.
DeviceSerial Number	UTF-8 string, no BOM	0 .. 32		17+N+M	Device and manufacturer-specific serial number.
Device structure label	Byte array	7	0..254 for each byte	17+N+M+O	Label given by device to identify the device description structure. This label allows the device to identify the current version of the device description object pool present in a task controller and to determine whether an update is needed. The device structure label byte 1 is transmitted closest to the DeviceSerialNumber parameter.
Device localization label	Byte array	7	0..254 for each of bytes 1 to 6, 255 for byte 7	24+N+M+O	Label given by device to identify the device description localization. Bytes 1 to 6 are defined by the language command PGN (see ISO 11783-7). Byte 7 is reserved and set to FF_{16}. Byte 1 is transmitted closest to the device structure label parameter and is the least significant byte of the DeviceLocalizationLabel in the XML element device.

A.3 Definition of DeviceElementObject

DeviceElementObject is the object definition of the XML element DeviceElement. The attribute DeviceElementType specifies the type of this particular element definition. The type “device” represents the complete device itself and therefore can exist only once per object pool. See Table A.2.

Referable child objects:

DeviceProcessDataObject

DevicePropertyObject

Table A.2 — DeviceElementObject definition

Attribute Name	**Type**	**Size** bytes	**Range or value**	**Record byte**	**Description**
Table ID	String	3	DET	1-3	XML element namespace for DeviceElement.
Object ID	Integer	2	1 .. 65534	4-5	Unique object identifier.
DeviceElementType	Byte	1	1 .. 7	6	1 = device 2 = function 3 = bin 4 = section 5 = unit 6 = connector 7 = navigation reference See detailed description below this table.
Number of designator bytes (N)	Byte	1	0 .. 32	7	Length of following designator UTF-8 string.
DeviceElement Designator	UTF-8 string, no BOM	0 .. 32		8	Descriptive text to identify this device element.
DeviceElement Number	Integer	2	0 .. 4095	8+N	Element number for process data variable addressing according to the definitions in process data variable definitions in Annex B.
Parent ObjectId	Integer	2	0 .. 65534	10+N	Object ID of parent DeviceElementObject or DeviceObject in order to establish a hierarchical order of DeviceElements.
Number of objects to follow	Integer	2		12+N	Number of following object references.
REPEAT: ObjectId	Integer	2		14+N+object	List of references to DeviceProcessDataObjects or DevicePropertyObjects.

A.3.1 Device element type — Device

The device description object pool shall have one device element of type device with device element number = 0, which represents the complete device and makes it addressable for the task controller.

A.3.2 Device element type — Function

This device element type can be used as a generic device element to define individually accessible components of a device like valves or sensors.

A.3.3 Device element type — Bin

This is, for instance, the tank of a sprayer or the bin of a seeder.

A.3.4 Device element type — Section

This is, for instance, the section of a spray boom, seed toolbar or planter toolbar. A section may provide device geometry definitions (x, y, z) and a working width next to supported process data elements as device process variable values or device property values.

A.3.5 Device element type — Unit

This device element type is, for example, used for spray boom nozzles, seeder openers or planter row units. It is intended as a layer below the device element type section in the hierarchical device element structure.

A.3.6 Device element type — Connector

This device element type specifies the mounting/connection position of the device. More than one connector can be defined for one device (e.g. a tractor may provide front-end mounting and rear-end mounting connection locations). A connector element shall provide its device geometry definitions (x, y, z) relative to the device reference point as device process data values or as device property values, even when the device reference point is the same as the location of the connector (x = y = z = 0).

A.3.7 Device element type — Navigation reference

This device element type defines the navigation reference position for navigation devices such as GPS receivers. Such elements have to reference their position in the x-, y-, and z-direction as device process data values or device property values.

A.4 Definition of DeviceProcessDataObject

The DeviceProcessDataObject is the object definition of the XML element DeviceProcessData. Each object contains a single process data variable definition. See Table A.3.

Referable child object: DeviceValuePresentationObject.

Table A.3 — DeviceProcessDataObject definition

Attribute Name	Type	Size bytes	Range or value	Record byte	Description
Table ID	String	3	DPD	1-3	XML element namespace for DeviceProcessData.
Object ID	Integer	2	1 .. 65534	4-5	Unique object identifier.
Process data DDI	Integer	2	0 .. 65535	6-7	Identifier of process data variable (DDI) according to definitions in Annex B and ISO 11783-11.
Process data properties	Byte	1	0 .. 3	8	A bitset combined of: bit 0 = 1 = member of default set. bit 1 = 1 = setable.
Process data available trigger methods	Byte	1	0 .. 31	9	A bitset combined of: Bit 0 = 1 = time interval. Bit 1 = 1 = distance interval. Bit 2 = 1 = threshold limits. Bit 3 = 1 = on change. Bit 4 = 1 = total. See detailed description in A.4.1 to A.4.5.
Number of designator bytes (N)	Byte	1	0 .. 32	10	Length of following designator UTF-8 string.
Process data designator	UTF-8 string, no BOM	0 .. 32		11	Descriptive text for this device process data.
Device value presentation object ID	Integer	2	1 .. 65535	11+N	Object identifier of DeviceValuePresentationObject. Use NULL object ID when no DeviceValuePresentationObject is referenced.

A.4.1 Device process data trigger method — Time interval

The device can provide these device process data based on a time interval.

A.4.2 Device process data trigger method — Distance interval

The device can provide these device process data based on a distance interval.

A.4.3 Device process data trigger method — Threshold limits

The device can provide these device process data based on a surpassing of the value threshold.

A.4.4 Device process data trigger method — On change

The device can provide these device process data when its value changes.

A.4.5 Device process data trigger method — Total

These device process data are a total. See 6.6.2 for a description of total functionality.

A.5 Definition of DevicePropertyObject

DevicePropertyObject is the object definition of the XML element DeviceProperty. Each object contains a single DeviceElementProperty definition. See Table A.4.

Referable child object: DeviceValuePresentationObject.

Table A.4 — DevicePropertyObject definition

Attribute name	**Type**	**Size** bytes	**Range or value**	**Record byte**	**Description**
Table ID	String	3	DPT	1-3	XML element namespace for DeviceProperty.
Object ID	Integer	2	1 .. 65534	4-5	Unique object identifier.
Property DDI	Integer	2	0 .. 65535	6-7	Identifier of property (DDI) according to definitions in Annex B and ISO 11783-11.
Property value	Signed integer	4	-2^{31} .. $(2^{31}-1)$	8-11	Value of property.
Number of designator bytes (N)	Byte	1	0 .. 32	12	Length of following designator UTF-8 string.
Property designator	UTF-8 string, no BOM	0 .. 32		13	Descriptive text for this device property.
Device value presentation object ID	Integer	2	1 .. 65535	13+N	Object identifier of DeviceValuePresentationObject. Use NULL object ID when no DeviceValuePresentationObject is referenced.

A.6 Definition of DeviceValuePresentationObject

DeviceValuePresentationObject is the object definition of the XML element DeviceValuePresentation. This object contains the presentation information to display the value of a DeviceProcessData or DeviceProperty object. The device can update these objects when the language and/or units of measure are changed by the operator. See Table A.5.

Referable child objects: none.

Table A.5 — DeviceValuePresentationObject definition

Attribute name	Type	Size bytes	Range or value	Record byte	Description
Table ID	String	3	DVP	1-3	XML element namespace for DeviceValuePresentation.
Object ID	Integer	2	1 .. 65534	4-5	Unique object identifier.
Offset	Signed integer	4	-2^{31} .. $(2^{31}-1)$	6-9	Offset to be applied to the value for presentation.
Scale	Float	4	0.000000001 … 100000000.0	10-13	Scale to be applied to the value for presentation.
Number of decimals	Byte	1	0 .. 7	14	Specify number of decimals to display after the decimal point.
Number of UnitDesignator bytes (N)	Byte	1	0 .. 32	15	Length of following unit designator UTF-8 string.
UnitDesignator	UTF-8 string, no BOM	0 .. 32		16	Unit designator for this value presentation.

A.7 Object hierarchy

Figure A.1 illustrates the hierarchy of device-related objects. A device descriptor object pool can contain only a single device object and can have multiple DeviceElement, DeviceProcessData, DeviceProperty and DeviceValuePresentation objects. The DeviceProcessData and DeviceProperty objects can be referenced from multiple DeviceElement objects. The DeviceValuePresentation object can be referenced from multiple DeviceProcessData or DeviceProperty objects.

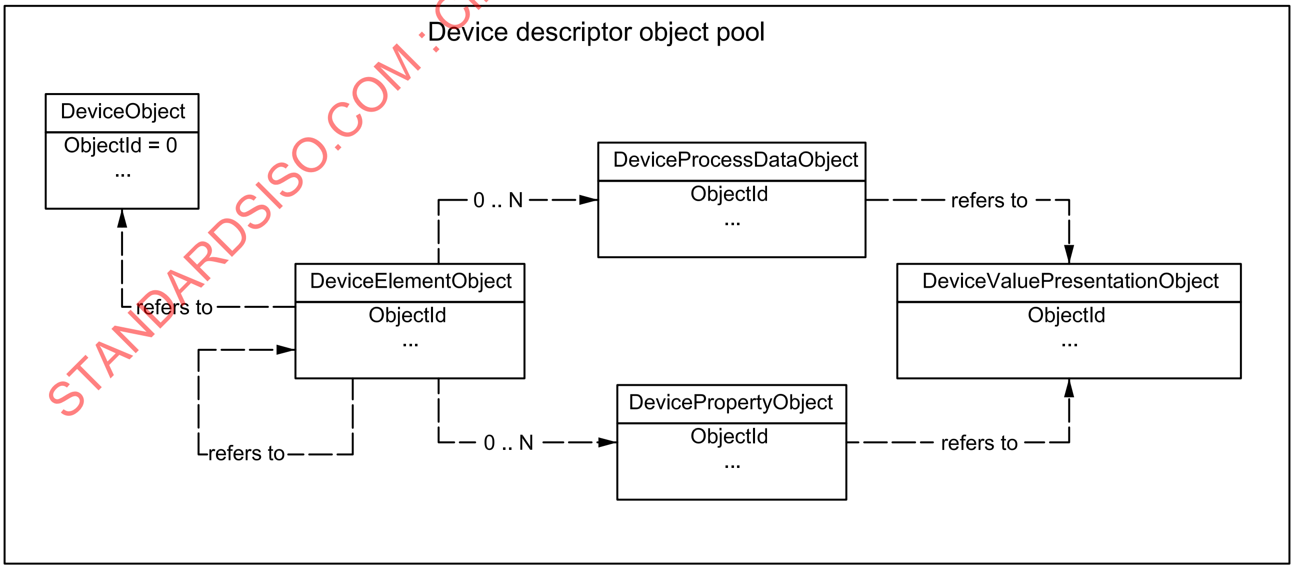


Figure A.1 — Object hierarchy

Annex B
(normative)

Message definitions

B.1 General

B.1.1 Process Data message

The Process Data message is used for the transmission of device description data, measured data and/or set point commands to one or more controllers. The first nibble of the first byte of the message identifies the command or the action that the controller(s) are required to perform. The remainder of the process data message for commands 0_{16} and 1_{16} is different from the remainder of the process data message of the commands 2_{16} until 8_{16} and D_{16} until F_{16}. Table B.1 lists these commands and specifies the definition or location in this annex.

B.1.2 Command parameter

Data length:	4 bits
Data range:	0 to 15
SPN:	5199

B.2 Process Data message parameter group

The Process Data message parameter group is defined as:

Data page:	0
Extended data page:	0
PDU format:	203
PDU specific:	Destination address
Default priority:	3
Parameter group number:	51968 ($00CB00_{16}$)
Data length:	Variable, minimum of 8 bytes.

The message may be sent to a GA, requiring that all controllers parse and determine the disposition of the message. The message should only be processed when the controller has determined that it has been addressed to its location.

The task controller can consist of multiple control functions; in that case, the task controller shall use the working-set master and member messages as defined in ISO 11783-1 and ISO 11783-7 to announce the task controller working set.

Table B.1 — Process Data commands

Command value	Meaning
0_{16}	Subcommand for determining the technical capabilities of a task controller or working-set master. These technical data messages are defined in B.4.
1_{16}	Subcommand for the transfer and management of device descriptions. These device description messages are defined in B.5.
2_{16}	Request value command. The value of the data entity specified by the data dictionary identifier is requested. The layout of this message is defined in B.3.
3_{16}	Value command: the process data value is the value of the data entity specified by the data dictionary identifier. This command is used both to answer a request value command and to set the value of a process data entity. The layout of this message is defined in B.3.
4_{16}	Measurement command: the process data value is the time interval for sending the data element specified by the data dictionary identifier. The working set has to send the value of this data element to the task controller cyclic with this time interval. The unit of the time interval is milliseconds. The layout of this message is defined in B.3.
5_{16}	Measurement command: the process data value is the distance interval for sending the data element specified by the data dictionary identifier. The working set has to send the value of this data element to the task controller cyclic with this distance interval. The unit of the distance interval is millimetres. The layout of this message is defined in B.3.
6_{16}	Measurement command: the process data value is the minimum within threshold for sending the data element specified by the data dictionary identifier. The working set has to send the value of this data element to the task controller when the value is higher than the threshold value. The unit of the threshold is specified by the data dictionary identifier definition. The layout of this message is defined in B.3.
7_{16}	Measurement command: the process data value is the maximum within threshold for sending the data element specified by the data dictionary identifier. The working set has to send the value of this data element to the task controller when the value is lower than the threshold value. The unit of the threshold is specified by the data dictionary identifier definition. The layout of this message is defined in B.3.
8_{16}	Measurement command: the process data value is the change threshold for sending the data entity specified by the data dictionary identifier. The working set has to send the value of this data element to the task controller when the value change is higher than the change threshold since last transmission. The unit of the threshold is specified by the data dictionary identifier definition. The layout of this message is defined in B.3.
9_{16} to C_{16}	Reserved.
D_{16}	Message is a process data negative acknowledge (PDNACK). This message is specified in B.6.
E_{16}	Message is a task-controller status message. This message is specified in B.7.1
F_{16}	Working-set task message — sent by the working-set master — is defined in B.7.2.

B.3 Value and measurement commands

B.3.1 Parameter fields

The second nibble of the first byte and the second byte identify the controllable element that must perform the commanded action. The third and fourth bytes contain the data dictionary identifier that identifies the data entity of which the value is specified in the remainder of the message. The last four bytes in the Process Data message contain the value of the data to be used to perform the specified action.

The byte ordering of the process variable parameter fields is:

Byte 1:	Bits 4–1	Command, possible values 2_{16} until 8_{16} (see Table B.1)
Byte 1:	Bits 8–5	Element number (LSNibble) (see B.3.2)
Byte 2:	Bits 8–1	Element number (MSB) (see B.3.2)
Byte 3:		DD identifier (LSB) (see B.3.3)
Byte 4:		DD identifier (MSB) (see B.3.3)
Bytes 5–8:		Process variable value (see B.3.4)

B.3.2 Element number

The element number is a 12-bit field comprises Byte 1, bits 4–7 and Byte 2. It indicates the specific controllable element that must act on the command. The set of numbers is defined in the device description. The syntax of {element number, parent number} is used to describe the configuration.

Data length:	12 bits
Data range:	0 to 4095
SPN:	5200

EXAMPLE A sprayer of three boom sections with six nozzles on the first boom section, eight nozzles on the second boom section and six nozzles on the third boom section {element number, parent number}:

{0,Null}

{1,0}, {2,0}, {3,0}

{4,1}, {5,1}, {6,1}, {7,1}, {8,1}, {9,1}

{10,2}, {11,2}, {12,2}, {13,2}, {14,2}, {15,2}, {16,2}, {17,2}

{18,3}, {19,3}, {20,3}, {21,3}, {22,3}, {23,3}

The element number would be 0 to address the implement sprayer. The element number would be 2 to address the second boom section. To address the second nozzle on the third boom, the element number would be 19.

B.3.3 Data dictionary identifier

This two-byte parameter is the identifier of the data dictionary entity that defines the data contained in the process variable value parameter. The data dictionary entities are listed in ISO 11783-11.

Data length:	2 bytes
Data range:	0 to 65255
SPN:	5201

B.3.4 Process variable value

This four-byte parameter contains the data value for the Process Data message. This value is defined as signed long integer data type.

Data length:	4 bytes
Resolution:	per data dictionary entity definition
Data range:	per data dictionary entity definition
SPN:	5202

B.4 Technical data messages

B.4.1 General

The technical data messages are used to request the characteristics of the task controller and participating working sets.

B.4.2 Request Version message

The Request Version message allows the task controller and the working set to determine the version of the implementation.

Transmission repetition rate:			On request
Data length:			8 bytes
Parameter group number:			Process data, destination-specific
Byte 1:	Bits 3–0	0000	Command technical data
	Bits 7–4	0000	Parameter Request Version
Bytes 2–8:			Reserved, transmit as FF_{16}

B.4.3 Version message

The Version message is sent in response to the request version message and contains the version information of the task controller or working-set implementation.

Transmission repetition rate:			In response to Request Version message
Data length:			8 bytes
Parameter group number:			Process data, destination-specific
Byte 1:	Bits 3–0	0000	Command technical data
	Bits 7–4	0001	Parameter Version
Byte 2:	Version number		The version of ISO 11783-10 that this control function meets
		0	The version of the DIS (draft International Standard)
		1	The version of the FDIS.1 (final draft International Standard, first edition)
		2	The version of the FDIS.2 and the first edition published as an International Standard, and so on
Bytes 3–8:			Reserved, transmit as FF_{16}

B.5 Device description messages

B.5.1 General

The device description messages are used to transfer the device description from working set to task controller and to maintain the device description object pool.

B.5.2 Request Structure Label message

The Request Structure Label message allows the working set to determine the version of the latest device description structure present at the task controller. If a structure label is not present, a structure label message with all structure label bytes set to value = FF_{16} shall be transmitted by the task controller to the sender of the Request Structure Label message.

Transmission repetition rate:			On request
Data length:			8 bytes
Parameter group number:			Process data, destination-specific
Byte 1:	Bits 3–0	0001	Device description
	Bits 7–4	0000	Request Structure Label
Bytes 2–8:			Reserved, transmit as FF_{16}

B.5.3 Structure Label message

The Structure Label message is sent by the task controller to inform the working set about the latest version of the device description structure present at the task controller.

Transmission repetition rate:			In response to Request Structure Label message
Data length:			8 bytes
Parameter group number:			Process data, destination-specific
Byte 1:	Bits 3–0	0001	Device description
	Bits 7–4	0001	Structure label
Bytes 2–8:			7-byte device description structure label. Byte 2 is data array byte 1 and Byte 8 is data array byte 7. Transmit all bytes with value = FF_{16} when structure label is not present.

B.5.4 Request Localization Label message

The device description Request Localization Label request message allows the working set to determine the localization version of the latest device description available at the task controller. If a localization label is not present, a localization label message with all localization label bytes set to value = FF_{16} shall be transmitted by the task controller to the sender of the Request Localization Label message.

Transmission repetition rate:			On request
Data length:			8 bytes
Parameter group number:			Process data, destination-specific
Byte 1:	Bits 3–0	0001	Device description
	Bits 7–4	0010	Request Localization Label
Bytes 2–8:			Reserved, transmit as FF_{16}

B.5.5 Localization Label message

Transmission repetition rate:			In response to Request Localization Label message
Data length:			8 bytes
Parameter group number:			Process data, destination-specific
Byte 1:	Bits 3–0	0001	Device description
	Bits 7–4	0011	Localization Label
Bytes 2–8:			7 byte device description localization label. Bytes 1 to 6 are defined by the language command PGN (see ISO 11783-7). Byte 7 is reserved and set to FF_{16}. Transmit all bytes with value = FF_{16} when localization label is not present.

B.5.6 Request Object-pool Transfer message

The Request Object-pool Transfer message allows the working set to determine whether it is allowed to transfer (part of) the device description object pool to the task controller.

Transmission repetition rate:			On request
Data length:			8 bytes
Parameter group number:			Process data, destination-specific
Byte 1:	Bits 3–0	0001	Device description
	Bits 7–4	0100	Request Object-pool Transfer
Bytes 2–5:			Requested transfer data size in bytes
Bytes 6–8:			Reserved, transmit as FF_{16}

B.5.7 Request Object-pool Transfer Response message

Transmission repetition rate:			In response to Request Object-pool Transfer message
Data length:			8 bytes
Parameter group number:			Process data, destination-specific
Byte 1:	Bits 3–0	0001	Device description
	Bits 7–4	0101	Request Object-pool Transfer Response
Byte 2:	Status	0	There may be enough memory available. However, because there is overhead associated with object storage it is impossible to predict whether there is enough memory available.
		1	There is not enough memory available. Do not transmit object pool.
Bytes 3–8:			Reserved, transmit as FF_{16}

B.5.8 Object-pool Transfer message

The Object-pool Transfer message enables the working set to transfer (part of) the device description object pool to the task controller. The transfer of the device description object pool can be split up over multiple object-pool transfer messages. When the object-pool transfer is split up over multiple Object-pool Transfer messages, each single object-pool transfer shall contain complete object descriptions.

Transmission repetition rate:			On request
Data length:			Variable
Parameter group number:			Process data, destination-specific
Byte 1:	Bits 3–0	0001	Device description
	Bits 7–4	0110	Object-pool Transfer
Bytes 2–n:			Object-pool records

B.5.9 Object-pool Transfer Response message

Transmission repetition rate:			In response to Object-pool Transfer message
Data length:			8 bytes
Parameter group number:			Process data, destination-specific
Byte 1:	Bits 3–0	0001	Device description
	Bits 7–4	0111	Object-pool Transfer Response
Byte 2:	Error code	0	No error, transfer OK
		1	Task controller ran out of memory during transfer
		2	Any other error
Bytes 3–6:			Received data size in bytes
Bytes 7–8:			Reserved, transmit as FF_{16}

B.5.10 Object-pool Activate message

This message is sent by a working-set master to indicate that the object pool is complete and ready for use. It is sent after the transfer of the object pool, after any object is redefined or added to the pool during operation or when comparison of the requested labels indicates that the version in the device is the same as the version available in the task controller.

Transmission repetition rate:			On request
Data length:			8 bytes
Parameter group number:			Process data, destination-specific
Byte 1:	Bits 3–0	0001	Device description
	Bits 7–4	1000	Object-pool Activate
Bytes 2–8:			Reserved, transmit as FF_{16}

B.5.11 Object-pool Activate Response message

This message is sent by the task controller to a working-set master to acknowledge the device description Object-pool Activate message. When the task controller replies with an error of any type, the task controller shall delete the object pool from the volatile memory. The task controller can optionally inform the operator of the reason for deletion. Communication with the working-set master can continue in the case of a device description already being present in the data transfer file from the FMIS. The task controller can also proceed with requesting process data from the working-set master based on what is in the data transfer file, either via the existing device description or from a planned task with DataLogTrigger elements defined.

Device descriptions that contain errors shall not be written to the data transfer file for transfer from the MICS to the FMIS.

Transmission repetition rate:			In response to Object-pool Activate message
Data length:			8 bytes
Parameter group number:			Process data, destination-specific
Byte 1:	Bits 3–0	0001	Device description
	Bits 7–4	1001	Object-pool Activate Response
Byte 2			Error codes (0 = no errors)
		Bit 0 = 1 =	There are errors in the object pool, refer to Bytes 3 to 7 for additional error information
		Bit 1 = 1 =	Task controller ran out of memory during activation
		Bit 2 = 1 =	Any other error
		Bits 3–7 =	Reserved, transmitted as 0 (zero)
Bytes 3, 4			Parent object ID of faulty object, transmit as NULL object ID if there are no object-pool errors
Bytes 5, 6			Object ID of faulty object, transmit as NULL object ID if there are no object-pool errors
Byte 7			Object-pool error codes (0 = no errors)
		Bit 0 = 1 =	Method or attribute not supported by the task controller
		Bit 1 = 1 =	Unknown object reference (missing object)
		Bit 2 = 1 =	Any other error
		Bit 3 = 1 =	Object pool was deleted from volatile memory
		Bits 4–7 =	Reserved, transmitted as 0 (zero)
Byte 8			Reserved, transmit as FF_{16}

B.5.12 Object-pool Delete message

This is a message to delete the device description object pool for the working set that sends this message. The Object-pool Delete message enables a working set to delete the entire device description object pool before sending an updated or changed device description object pool with the object-pool transfer message.

Only device description object pools that have been defined by the working set may be deleted. Device description object pools received from the FMIS are not affected.

Transmission repetition rate:			On request
Data length:			8 bytes
Parameter group number:			Process data, destination-specific
Byte 1:	Bits 3–0	0001	Device description
	Bits 7–4	1010	Object-pool Delete
Bytes 2–8:			Reserved, transmit as FF_{16}

B.5.13 Object-pool Delete Response message

Transmission repetition rate:			In response to Object-pool Delete message
Data length:			8 bytes
Parameter group number:			Process data, destination-specific
Byte 1:	Bits 3–0	0001	Device description
	Bits 7–4	1011	Object-pool Delete Response
Byte 2:			Error code: 0 = success; 1 = error
Bytes 3–8:			Reserved, transmit as FF_{16}

B.5.14 Change Designator message

This message is to update the designator of an object.

Transmission repetition rate:			On request
Data length:			Variable
Parameter group number:			Process data, destination-specific
Byte 1:	Bits 3–0	0001	Device description
	Bits 7–4	1100	Change Designator
Bytes 2, 3:			Object ID
Bytes 4:			Designator length, number of bytes
Bytes 5–n:			UTF-8 encoded designator, no BOM

B.5.15 Change Designator Response message

Transmission repetition rate:			In response to Change Designator message
Data length:			8 bytes
Parameter group number:			Process data, destination-specific
Byte 1:	Bits 3–0	0001	Device description
	Bits 7–4	1101	Change Designator Response
Bytes 2, 3:			Object ID
Bytes 4:			Error code: 0 = success; 1 = error
Bytes 5–8:			Reserved, transmit as FF_{16}

B.6 Process Data Negative Acknowledge (PDNACK) message

This message is sent by the working-set master or by the task controller to reject commands and process data. The reasons are given in the least significant byte of the process data value. When the process data error code reported is not associated with a specific element number or a specific DDI, the values of the element number or DDI shall be set to “not available”. The “not available” value of the element number is FFF_{16} and the “not available” value of the DDI is $FFFF_{16}$.

Transmission repetition rate:			On request
Data length:			8 bytes
Parameter group number:			Process data, destination-specific
Byte 1:	Bits 0–3	1101	Process Data Negative Acknowledge (PDNACK)
	Bits 4–7		Element number (LSNibble), send as F_{16} when not applicable to the error code
Byte 2:			Element number (MSB), send as FF_{16} when not applicable to the error code
Bytes 3–4:			DDI, send as $FFFF_{16}$ when not applicable to the error code
Byte 5:			Process Data Error Codes (0 = no errors)
		Bit 0 = 1 =	Process Data Command not supported
		Bit 1 = 1 =	Invalid element number
		Bit 2 = 1 =	DDI not supported by element
		Bit 3 = 1 =	Trigger method not supported
		Bit 4 = 1 =	Process data not setable
		Bit 5 = 1 =	Invalid or unsupported interval or threshold
		Bits 6, 7 = 0 =	Reserved
Bytes 6–8:			Reserved, transmit as FF_{16}

B.7 Status messages

The status messages allow the working set to determine the health of the task controller and to monitor the progress of tasks in the task controller. They also allow the task controller to monitor the health of working sets and supported commands, and processing of data by the working sets.

B.7.1 Task-controller Status message

This message shall be sent by the task controller to indicate the current task status.

Transmission repetition rate:			2 s and on change of any byte in this message. At least 200 ms shall pass between status messages.
Data length:			8 bytes
Parameter group number:			Process data, global destination
Byte 1:	Bits 0–3	1110	Task-controller status
	Bits 4–7	1111	Element number, set to not available
Byte 2:		FF_{16}	Element number, set to not available
Bytes 3–4:		$FFFF_{16}$	DDI, set to not available
Byte 5:			Actual task controller status
	Bit 0 = 1 =		Task controller totals active, a task is started or resumed
	On 0 to 1 transition:		Internal control function totals are reset to zero and counting starts
	On 1 to 0 transition:		Internal control function totals are stopped and may be queried by task controller through a series of Request Value commands
	Bit 1 = 1 =		Task controller is busy saving data to non-volatile memory
	Bit 2 = 1 =		Task controller is busy reading data from non-volatile memory
	Bit 3 = 1 =		Task controller is busy executing a B.3 command (device description messages)
	Bits 4–6 = 0 =		Reserved
	Bit 7 = 1 =		Task controller is out of memory
Byte 6:			Source address of working-set master for which B.3 command is being executed (valid only if Byte 5, Bit 3 is set, or else transmit value = 0)
Byte 7:			B.3 command (Byte 1) that is being executed (valid only if Byte 5, Bit 3 is set, or else transmit value = 0)
Byte 8:	0		Reserved

B.7.2 Working-set Task message

This message shall be sent by all involved working-set masters to the task controller to indicate the current working-set task status.

Transmission repetition rate:			2 s and on each task status change. At least 200 ms shall pass between working-set task messages.
Data length:			8 bytes
Parameter group number:			Process data, destination-specific
Byte 1:	Bits 0–3	1111	Working-set task
	Bits 4–7	1111	Element number, set to not available
Byte 2:		FF_{16}	Element number, set to not available
Bytes 3–4:		$FFFF_{16}$	DDI, set to not available
Bytes 5–8:	Bit 0		Actual task controller status: totals active (as received in task controller status message, Byte 5, Bit 0)
	Bits 1–31	0	Reserved

Annex C
(normative)

XML elements relationship diagram

C.1 XML element relationship

Figure C.1 shows the entity relationship diagram for all entities. The contents of an XML file shall be according to the relationships as specified in the diagram. The entities shown in Figure C.1 do not always contain foreign key identifiers, since this can be determined by the sequence of definition in the XML file.

The ERD specifies, for example, that the *Task* entity has a one-to-zero-or-more relation to *Worker*, the task can have no workers assigned, but also one or more. When one or more workers are assigned, this means that the *Worker* definition shall follow the *TaskHeaderData* definition, in order to get its relation to the task.

The line ends at each relation shall be read as "opposite line end" and determine multiplicity from the originating entity towards the other entity. For example, the relation between *Worker* and *WorkerAllocation* is read as: *Worker* can occur in zero or more *WorkerAllocations* (0 + crow foot) and a *WorkerAllocation* entity is bound to exactly 1 worker (single bar).

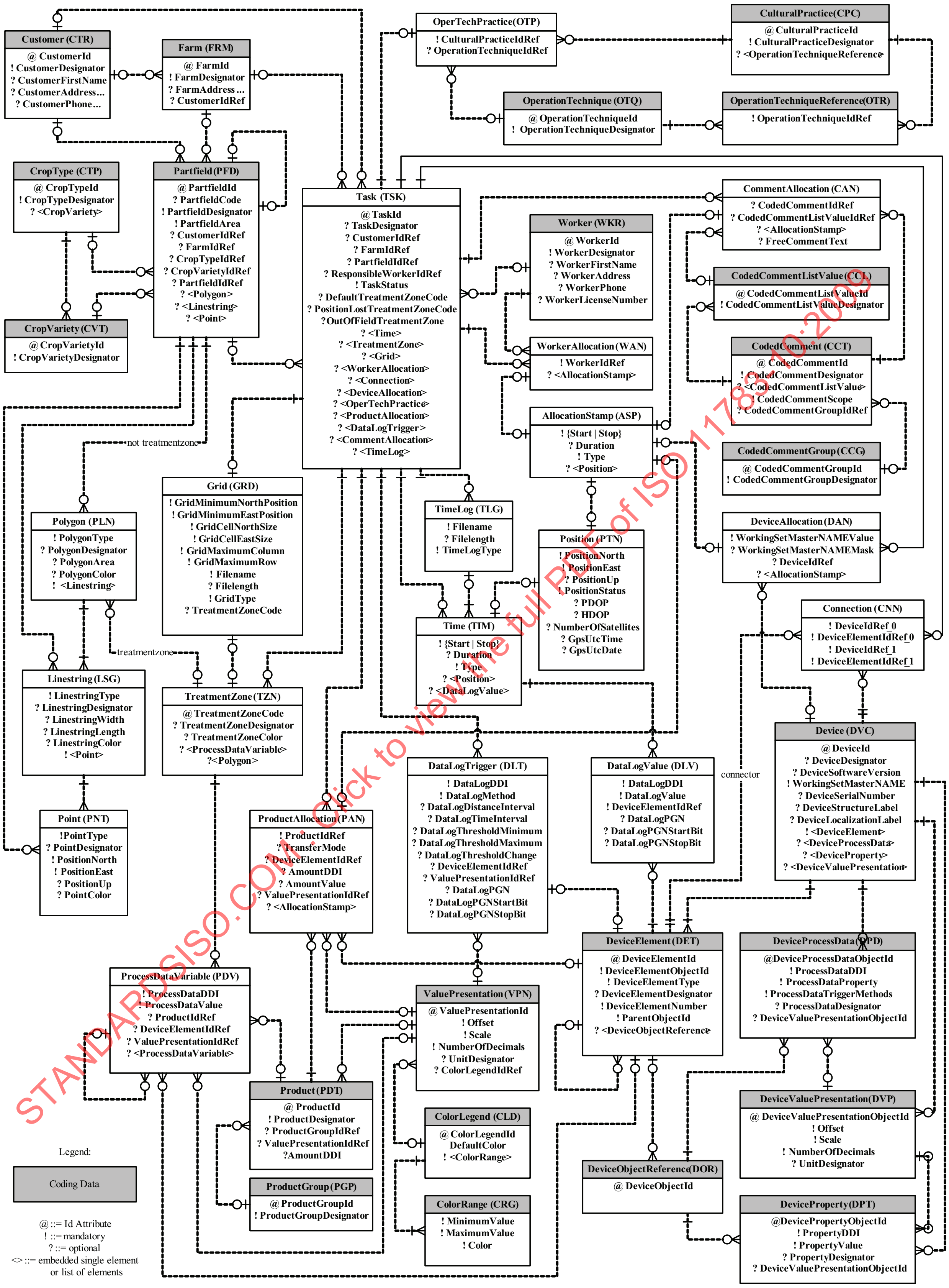


Figure C.1 — XML elements relationship diagram

Annex D
(normative)

XML elements and attributes

D.1 XML elements

The XML attributes of an XML element are described in this annex in tables with six columns:

— **Attribute**, name of the XML attribute.

— **XML**, XML attribute tag.

— **Use**, “r” denotes required, “o” denotes optional.

— **Type**, XML type specification of this attribute.

— **Length/range**, number of characters or value range of this attribute.

— **Comment**, additional explanation or restrictions on format of this attribute.

NOTE Child XML elements are also listed in the XML attributes.

D.2 AllocationStamp — ASP

Type:

Task data

Description:

The AllocationStamp XML element specifies a recording of an allocation event. Optionally, a position can be recorded at an AllocationStamp specification. All AllocationStamp XML elements that were provided by the FMIS shall be of the type *planned* and all AllocationStamp XML elements that were provided by the MICS shall be of the type *effective*.

All AllocationStamp XML elements shall have a Start or Stop attribute value defined. Duration shall always contain a positive value. All time values are to be set in local time.

Included by XML elements:

— *CommentAllocation*

— *DeviceAllocation*

— *ProductAllocation*

— *WorkerAllocation*

Includes XML element:

— *Position*

Attributes:

Attribute	XML	Use	Type	Length/range	Comment
Start	A	o	xs:datetime	Max. 19	Time of start. Format: yyyy-mm-ddThh:mm:ss
Stop	B	o	xs:datetime	Max. 19	Time of end. Format: yyyy-mm-ddThh:mm:ss
Duration	C	o	xs:unsignedLong	0 .. (2^{32}–2)	Time between start and stop in number of seconds.
Type	D	r	xs:NMTOKEN	1, 4	Type of the AllocationStamp, possible values: 1 = Planned 4 = Effective
Position		o	xs:element		Includes a single XML element Position

EXAMPLE

```
<ASP A="2003-08-20T08:10:00" D="4">
    <PTN A="54.588945" B="9.989209" D="3"/>
</ASP>
<ASP A="2003-08-20T08:10:00" C="3512" D="4"/>
```

D.3 CodedComment — CCT

Type:

Coding data

Description:

The CodedComment XML element describes predefined comments that can be used to annotate tasks. Comments can be grouped in CodedCommentGroups by establishing a reference to that group in CodedCommentGroupIdRef. A coded comment can include a list of possible values (e.g. low, medium, high). These values are described in CodedCommentValues. One of these values can be referenced when a comment is assigned to a task.

Referenced by XML element:

— *CommentAllocation*

Includes XML element:

— *CodedCommentListValue*

Attributes:

Attribute	XML	Use	Type	Length/range	Comment
CodedCommentId	A	r	xs:ID	min. 4 .. max. 14	Unique identifier of CodedComment Format: (CCT|CCT-)([0-9])+ Records generated on MICS have negative IDs
CodedCommentDesignator	B	r	xs:string	max. 32	Designator of CodedComment
CodedCommentScope	C	r	xs:NMTOKEN	1 .. 3	Selection of one attribute: 1 = point 2 = global 3 = continuous
CodedCommentGroupIdRef	D	o	xs:IDREF	min. 4 .. max. 14	Reference to ID of CodedCommentGroup Format: (CCG|CCG-)([0-9])+
CodedCommentListValue		o	xs:element		Includes XML elements CodedCommentListValue

EXAMPLE

```
<CCT A="CCT7" B="Thistles" C="3">
    <CCL A="CCL1" B="10 P./sqm"/>
    <CCL A="CCL2" B="20 P./sqm"/>
    <CCL A="CCL3" B="30 P./sqm"/>
</CCT>
```

D.4 CodedCommentGroup — CCG

Type:

Coding data

Description:

The CodedCommentGroup XML element can be used to combine predefined CodedComments into groups. The purpose is to have a better navigation and selection of CodedComments on the mobile system. Each CodedComment can belong to only one CodedCommentGroup. A CodedCommentGroup can, for instance, be “weeds”, which contains the CodedComments “camomile”, “couch grass” and “thistle”. To set up a list of CodedComments that belong to a particular CodedCommentGroup on the MICS, all CodedComment XML elements shall be examined for a match between XML attributes CodedCommentGroupIdRef and the identifier of that CodedCommentGroup.

Referenced by XML element:

— *CodedComment*

Attributes:

Attribute	XML	Use	Type	Length/range	Comment
CodedCommentGroupId	A	r	xs:ID	min. 4 .. max. 14	Unique identifier of CodedCommentGroup Format: (CCG|CCG-)([0-9])+ Records generated on MICS have negative IDs
CodedCommentGroupDesignator	B	r	xs:string	max. 32	Designator of CodedCommentGroup

EXAMPLE

```
<CCG A="CCG3" B="Weeds" />
<CCT A="CCT7" B="Thistles" C="3" D="CCG3" >
    <CCL A="CCL3" B="30 P./sqm"/>
    <CCL A="CCL2" B="20 P./sqm"/>
    <CCL A="CCL1" B="10 P./sqm"/>
</CCT>
```

D.5 CodedCommentListValue — CCL

Type:

Coding data

Description:

The CodedCommentListValue XML element provides a value to qualify a coded comment. Each CodedCommentListValue always belongs to a single CodedComment only. One of the CodedCommentListValues that belongs to a CodedComment can be referenced in CodedCommentAllocation when that CodedComment is assigned to a task. Examples of CodedCommentListValues are sets like “low”, “medium” and “high” or definitions such as “crop growth stage codes”.

Referenced by XML element:

— *CommentAllocation*

Included by XML element:

— *CodedComment*

Attributes:

Attribute	XML	Use	Type	Length/range	Comment
CodedCommentListValueId	A	r	xs:ID	min. 4 .. max. 14	Unique identifier of CodedCommentListValue Format: (CCL|CCL-)([0-9])+ Records generated on MICS have negative IDs
CodedCommentListValueDesignator	B	r	xs:string	max. 32	Designator of CodedCommentListValue

EXAMPLE

<**CCL** A="CCL0001" B="10 P./sqm"/>

<**CCL** A="CCL0002" B="20 P./sqm"/>

<**CCL** A="CCL0003" B="30 P./sqm"/>

D.6 ColourLegend — CLD

Type:

Coding data

Description:

The ColourLegend XML element describes a colour legend that can be used to display values in different colours on a grid map. A ColourLegend XML element includes ColourRange XML elements which specify the colour to be used when presenting the value. The attribute DefaultColour is used to present the value when no ColourRange within which the value falls is specified.

Referenced by XML element:

— *ValuePresentation*

Includes XML element:

— *ColourRange*

Attributes:

Attribute	XML	Use	Type	Length/range	Comment
ColourLegendId	A	r	xs:ID	min. 4 .. max. 14	Unique identifier of ColourLegend Format: (CLD|CLD-)([0-9])+ Records generated on MICS have negative IDs
DefaultColour	B	o	xs:unsignedByte	0 .. 254	Default colour of the ColourLegend Format: palette like ISO 11783-6
ColourRange		r	xs:element		Includes a list of XML elements ColourRange

EXAMPLE

```
<CLD A="CLD1" B="0">
    <CRG A="0" B="9999" C="1"/>
    <CRG A="10000" B="14999" C="2"/>
    <CRG A="15000" B="19999" C="3"/>
    <CRG A="20000" B="29999" C="4"/>
    <CRG A="30000" B="99999" C="5"/>
</CLD>
```

D.7 ColourRange — CRG

Type:

Coding data

Description:

The ColourRange XML element specifies the colour to use to display values within a certain value range on a grid map. A ColourRange XML element is included in a ColourLegend XML element to specify a range of colours to be used to present different ranges of values.

Included by XML element:

— *ColourLegend*

Attributes:

Attribute	XML	Use	Type	Length/range	Comment
MinimumValue	A	r	xs:long	$(–2^{31}+1)$ .. $(2^{31}–1)$	ColourRange minimum value. The value is included in the range.
MaximumValue	B	r	xs:long	$(–2^{31}+1)$ .. $(2^{31}–1)$	ColourRange maximum value. The value is included in the range.
Colour	C	r	xs:unsignedByte	0 .. 254	ColourRange colour Format: palette like ISO 11783-6

EXAMPLE

```
<CLD A="CLD1">
    <CRG A="0" B="9999" C="1"/>
    <CRG A="10000" B="14999" C="2"/>
    <CRG A="15000" B="19999" C="3"/>
    <CRG A="20000" B="29999" C="4"/>
    <CRG A="30000" B="99999" C="5"/>
</CLD>
```

D.8 CommentAllocation — CAN

Type:

Task data

Description:

The CommentAllocation XML element allocates a CodedComment or a free comment text to a task. The allocation of comments is specified at a certain position and time by inclusion of XML element AllocationStamp. Free comment can be added in XML attribute FreeCommentText. In case of assigning a CodedComment and a CodedCommentListValue to a task, the CodedCommentListValue shall be out of the list of values of the assigned CodedComment. A CommentAllocation can assign either a CodedComment or FreeComment exclusively.

Includes XML element:

— *AllocationStamp*

Included by XML element:

— *Task*

References XML elements:

— *CodedComment*

— *CodedCommentListValue*

Attributes:

Attribute	XML	Use	Type	Length/range	Comment
CodedCommentIdRef	A	o	xs:IDREF	min. 4 .. max. 14	Reference to XML element CodedComment Format: (CCT|CCT-)([0-9])+
CodedCommentListValueIdRef	B	o	xs:IDREF	min. 4 .. max. 14	Reference to XML element CodedCommentListValue Format: (CCL|CCL-)([0-9])+
FreeCommentText	C	o	xs:string	max. 32	Operator-definable free comment text
AllocationStamp		o	xs:element		Includes a single XML element AllocationStamp to specify the time and the position of the comment.

EXAMPLE

```
<CAN C="bad driving conditions">
    <ASP A="2003-08-20T08:00:20" D="4">
        <PTN A="51.23456" B="13.23456" D="3"/>
    </ASP>
</CAN>
<CAN A="CCT5" B="CCL1">
    <ASP A="2003-08-20T08:00:20" D="4">
        <PTN A="51.23456" B="13.23456" D="3"/>
    </ASP>
</CAN>
```

D.9 Connection — CNN

Type:

Task data

Description:

The purpose of the Connection XML element is to specify how two devices are connected to each other within a single task. A connection specification consists of references to the two DeviceElements of type “connector” of the two devices which are connected. The connection specification enables the task controller to determine the position of DeviceElements of one device relative to the position of, for instance, the NavigationReferencePoint of another device. The referenced DeviceElement in DeviceElementIdRef_0 shall be part of the device referenced in DeviceIdRef_0 and shall be of the type “connector”. The referenced DeviceElement in DeviceElementIdRef_1 shall be part of the device referenced in DeviceIdRef_1 and shall also be of the type “connector”.

Included by XML element:

— *Task*

References XML elements:

— *Device*

— *DeviceElement*

Attributes:

Attribute	XML	Use	Type	Length/range	Comment
DeviceIdRef_0	A	r	xs:IDREF	min. 4 .. max. 14	Reference to XML element Device Format: (DVC|DVC-)([0-9])+
DeviceElementIdRef_0	B	r	xs:IDREF	min. 4 .. max. 14	Reference to XML element DeviceElement Format: (DET|DET-)([0-9])+
DeviceIdRef_1	C	r	xs:IDREF	min. 4 .. max. 14	Reference to XML element Device Format: (DVC|DVC-)([0-9])+
DeviceElementIdRef_1	D	r	xs:IDREF	min. 4 .. max. 14	Reference to XML element DeviceElement Format: (DET|DET-)([0-9])+

EXAMPLE

```
<CNN A="DVC2" B="DET2" C="DVC1" D="DET1"/>
```

D.10 CropType — CTP

Type:

Coding data

Description:

The CropType XML element describes a crop that can be cultivated on a partfield. A CropType XML element can include several CropVariety XML elements.

Referenced by XML elements:

— *Partfield*

Includes XML elements:

— *CropVariety*

Attributes:

Attribute	XML	Use	Type	Length/range	Comment
CropTypeId	A	r	xs:ID	min. 4 .. max. 14	Unique identifier of CropType Format: (CTP|CTP-)([0-9])+ Records generated on MICS have negative IDs
CropTypeDesignator	B	r	xs:string	max. 32	Name of the crop
CropVariety		o	xs:element		Includes a list of XML elements CropVariety

EXAMPLE

```
<CTP A="CTP1" B="wheat">
    <CVT A="CVT1" B="Ritmo B"/>
    <CVT A="CVT2" B="Dekan B"/>
    <CVT A="CVT3" B="Ares C"/>
</CTP>
<CTP A="CTP2" B="barley"/>
<CTP A="CTP3" B="oats"/>
```

D.11 CropVariety — CVT

Type:

Coding data

Description:

The CropVariety XML element describes varieties of a crop specified by CropType that can be cultivated on a partfield. Each CropVariety definition belongs to a single CropType definition.

Included by XML elements:

— *CropType*

Referenced by XML elements:

— *Partfield*

Attributes:

Attribute	XML	Use	Type	Length	Comment
CropVarietyId	A	r	xs:ID	min. 4 .. max. 14	Unique identifier of CropVariety Format: (CVT|CVT-)([0-9])+ Records generated on MICS have negative IDs
CropVarietyDesignator	B	r	xs:string	max. 32	Name of the CropVariety

EXAMPLE

<**CVT** A="CVT1" B="Ritmo B"/>

<**CVT** A="CVT2" B="Dekan B"/>

<**CVT** A="CVT3" B="Ares C"/>

D.12 CulturalPractice — CPC

Type:

Coding data

Description:

The CulturalPractice XML element describes cultural practices that can be allocated to a task. Examples of CulturalPractice definitions are “ploughing” or “seeding”. A CulturalPractice can refer to a list of references to several OperationTechniques (e.g. cultural practice: “fertilization” -> operation techniques: “liquid fertilization”, “organic fertilization”, “gaseous fertilization”).

Referenced by XML elements:

— *OperTechPractice*

Includes XML elements:

— *OperationTechniqueReference*

Attributes:

Attribute	XML	Use	Type	Length	Comment
CulturalPracticeId	A	r	xs:ID	min. 4 .. max. 14	Unique identifier of CulturalPractice Format: (CPC|CPC-)([0-9])+ Records generated on MICS have negative IDs
CulturalPracticeDesignator	B	r	xs:string	max. 32	Designator of the CulturalPractice
OperationTechniqueReference		o	xs:element		Includes a list of XML element OperationTechniqueReference

EXAMPLE

```
<CPC A="CPC1" B="fertilization">
    <OTR A="OTQ1"/>
</CPC>
<CPC A="CPC2" B="seeding"/>
<CPC A="CPC3" B="harvest"/>
```

D.13 Customer — CTR

Type:

Coding data

Description:

The Customer XML element describes a customer. A customer can be referenced in a task, farm and partfield. The relationship between a customer and farms and partfields can be multiple. To determine which farms or partfields belong to a specific customer, the CustomerIdRef values of all farms or partfields, respectively, shall be examined for a match with a particular CustomerId value.

Referenced by XML elements:

— *Task*

— *Farm*

— *Partfield*

Attributes:

Attribute	XML	Use	Type	Length/range	Comment
CustomerId	A	r	xs:ID	min. 4 .. max. 14	Unique identifier of customer Format: (CTR|CTR-)([0-9])+ Records generated on MICS have negative IDs
CustomerDesignator	B	r	xs:string	max. 32	Customer designator/surname
CustomerFirstName	C	o	xs:string	max. 32	Customer's first name
CustomerStreet	D	o	xs:string	max. 32	Street
CustomerPOBox	E	o	xs:string	max. 32	PO box
CustomerPostalCode	F	o	xs:string	max. 10	Postal code
CustomerCity	G	o	xs:string	max. 32	City
CustomerState	H	o	xs:string	max. 32	State or county
CustomerCountry	I	o	xs:string	max. 32	Country
CustomerPhone	J	o	xs:string	max. 20	Telephone number
CustomerMobile	K	o	xs:string	max. 20	Mobile phone number
CustomerFax	L	o	xs:string	max. 20	Fax number
CustomerEMail	M	o	xs:string	max. 64	E-mail

EXAMPLE

<**CTR** A="CTR1" B="Smith" C="John" G="Munich"/>

D.14 DataLogTrigger — DLT

Type:

Task data

Description:

The DataLogTrigger XML element is included in the task and contains the information about which ProcessDataVariables values shall be logged as DataLogValues during task processing. The reference to the DeviceElement can be added on the mobile system, as soon as a device is allocated to the task. When the reference to the DeviceElement is already given at FMIS then a specific device was planned for the task. The attributes of the DataLogTrigger define the behaviour of the task controller, regarding how to collect and store the ProcessDataVariable values.

The data log methods “time interval”, “distance interval” and “on change” can be used in any combination. The logging is triggered by the event that occurs first, and all active methods of these three data log methods are then restarted. Additionally, the “threshold limits” method can be added. With this addition, the logging is triggered when the logging value enters the value range specified by the threshold limit definitions and is enabled as long as the value is within the value range specified by the threshold limit definitions.

If the DataLogThresholdMinimum is smaller than the DataLogThresholdMaximum, data logging is enabled when the value is between the DataLogThresholdMinimum and the DataLogThresholdMaximum. If the DataLogThresholdMinimum is larger than the DataLogThresholdMaximum, data logging is enabled when the value is larger than the DataLogThresholdMinimum or smaller than the DataLogThresholdMaximum.

The data log method “total” is independent of the other data log methods and can be used with any combination described above. DataLogValues of the type “total” shall be stored once per task in the data transfer file.

A maximum of 10 messages per process data variable per second can be transmitted by a device.

Values from parameter groups can be logged by specification of the attributes DataLogPGN, DataLogPGNStartBit and DataLogPGNStopBit. When these attributes are specified, the DataLogDDI attribute shall be set to 0xDFFE (PGN log value).

Included by XML elements:

— *Task*

References XML elements:

— *DeviceElement*

Attributes:

Attribute	XML	Use	Type	Length/range	Comment
DataLogDDI	A	r	xs:hexBinary	0x0000 .. 0xFFFF	Unique number, which identifies the ProcessDataVariable (as specified in ISO 11783-11)
DataLogMethod	B	r	xs:unsignedByte	1 .. 31	Selection of the log method: 1 = time interval 2 = distance interval 4 = threshold limits 8 = on change 16 = total
DataLogDistanceInterval	C	o	xs:long	0 .. 1000000	Distance interval for data log in mm, 0 stops measurement.
DataLogTimeInterval	D	o	xs:long	0 .. 60000	Time interval for data log in ms, 0 stops measurement, 10 ms is minimum time interval.
DataLogThresholdMinimum	E	o	xs:long	$(-2^{31}+1)$ .. $(2^{31}-1)$	Minimum threshold to activate the data log. Threshold limit value is included in the value range to log. $(2^{31}-1)$ stops measurement.
DataLogThresholdMaximum	F	o	xs:long	$(-2^{31}+1)$ .. $(2^{31}-1)$	Maximum threshold to activate the data log. Threshold limit value is included in the value range to log. $(-2^{31}+1)$ stops measurement.
DataLogThresholdChange	G	o	xs:long	$(-2^{31}+1)$ .. $(2^{31}-1)$	Change threshold to activate the data log, 0 stops measurement, 1 logs each change.
DeviceElementIdRef	H	o	xs:IDREF	min. 4 .. max. 14	Reference to XML element DeviceElement Format (DET|DET-)([0-9])+
ValuePresentationIdRef	I	o	xs:IDREF	min. 4 .. max. 14	Reference to XML element ValuePresentation Format (VPN|VPN-)([0-9])+
DataLogPGN	J	o	xs:unsignedLong	0 .. 2^{18}	Parameter Group to log a value from.
DataLogPGNStartBit	K	o	xs:unsignedByte	0 .. 63	First bit of the value to log from a parameter group. Bit 0 is the least significant bit of Byte 1 in the data field of a data frame. The start bit is included in the value and becomes the least significant bit.
DataLogPGNStopBit	L	o	xs:unsignedByte	0 .. 63	Stop bit of the value to log from a parameter group. The stop bit is included in the value and becomes the most significant bit.

EXAMPLE

<**DLT** A="1122" B="1" D="1000" H="DET2"/>

D.15 DataLogValue — DLV

Type:

Task data

Description:

The DataLogValue specifies a single value of a single ProcessDataVariable, specified by its DDI and supplied by a single DeviceElement. The XML attribute DeviceElementIdRef references the appropriate DeviceElement. The position and time of a DataLogValue are specified through the XML element Time. Time is included in the task and by this relation all DataLogValues belong to a task. DataLogValues are part of the data logging functionality of the task controller.

When the values are logged from a parameter group, the attributes DataLogPGN, DataLogPGNStartBit and DataLogPGNStopBit are used and the DataLogDDI attribute shall be set to 0xDFFE (PGN log value).

Included by XML elements:

— *Time*

References XML elements:

— *DeviceElement*

Attributes:

Attribute	XML	Use	Type	Length/range	Comment
ProcessDataDDI	A	r	xs:hexBinary	0x0000 .. 0xFFFF	Unique number, which defines the ProcessDataVariable (as specified in Annex B and ISO 11783-11)
ProcessDataValue	B	r	xs:long	-2^{31} .. $(2^{31}-1)$	Value
DeviceElementIdRef	C	r	xs:IDREF	min. 4 .. max. 14	Reference to DeviceElement Format: (DET|DET-)([0-9])+
DataLogPGN	D	o	xs:unsignedLong	0 .. 2^{18}	Parameter Group to log a value from.
DataLogPGNStartBit	E	o	xs:unsignedByte	0 .. 63	First bit of the value to log from a parameter group. Bit 0 is the least significant bit of Byte 1 in the data field of a data frame. The start bit is included in the value and becomes the least significant bit.
DataLogPGNStopBit	F	o	xs:unsignedByte	0 .. 63	Stop bit of the value to log from a parameter group. The stop bit is included in the value and becomes the most significant bit.

EXAMPLE

<**DLV** A="0815" B="10" C="DET1"/>

<**DLV** A="4711" B="15" C="DET2"/>

<**DLV** A="4522" B="20" C="DET3"/>

D.16 Device — DVC

Type:

Coding data

Description:

The Device XML element describes a complete device, like a machine or sensor system. Each device shall have at least one DeviceElement.

Includes XML elements:

— *DeviceElement*

— *DeviceProcessData*

— *DeviceProperty*

— *DeviceValuePresentation*

Referenced by XML elements:

— *Connection*

— *DeviceAllocation*

Attributes:

Attribute	XML	Use	Type	Length/range	Comment
DeviceId	A	r	xs:ID	min. 4 .. max. 14	Unique identifier of device Format: (DVC|DVC-)([0-9])+
DeviceDesignator	B	o	xs:string	max. 32	Device designator
DeviceSoftwareVersion	C	o	xs:string	max. 32	Software version of device
WorkingSetMasterNAME	D	r	xs:hexBinary	0x0000000000000000 .. 0xFFFFFFFFFFFFFFFF	NAME of working-set master (see ISO 11783-5)
DeviceSerialNumber	E	o	xs:string	max. 32	Serial number of device
DeviceStructureLabel	F	r	xs:hexBinary	0x00 .. 0xFE per byte	Label of device description structure Array Byte 1 is most significant and Array Byte 7 is least significant.
DeviceLocalizationLabel	G	r	xs:hexBinary	Bytes 1 to 6: 0x00 .. 0xFE per byte, Byte 7 = 0xFF	Label of device description localization. Bytes 1 to 6 are defined by the language command PGN (see ISO 11783-7). Byte 7 is reserved and set to FF_{16}. Language command PGN Byte 1 is least significant and language command Byte 7 is most significant.
DeviceElement		r	xs:element		Includes a list of XML elements DeviceElement
DeviceProcessData		o	xs:element		Includes a list of XML elements DeviceProcessData
DeviceProperty		o	xs:element		Includes a list of XML elements DeviceProperty
DeviceValuePresentation		o	xs:element		Includes a list of XML elements DeviceValuePresentation

EXAMPLE

```
<DVC A="DVC1" B="sprayer 4711" C="1.0" D="0102030405060708" F="050504A" G="FF000000006C6E">
    <DET A="DET1" B="1" C="1" D="all elements" E="0" F="0">
        <DOR A="3"/>
        <DOR A="4"/>
        <DOR A="8"/>
        <DOR A="9"/>
        <DOR A="10"/>
        <DOR A="11"/>
    </DET>
    <DET A="DET2" B="2" C="3" D="main tank" E="1" F="1">
        <DOR A="5"/>
        <DOR A="6"/>
        <DOR A="7"/>
        <DOR A="8"/>
        <DOR A="9"/>
        <DOR A="10"/>
        <DOR A="12"/>
    </DET>
    <DPD A="3" B="1234" C="3" D="1"/>
    <DPD A="4" B="8765" C="1" D="1"/>
    <DPD A="5" B="1111" C="3" D="1"/>
    <DPD A="6" B="1112" C="3" D="1"/>
    <DPD A="7" B="1133" C="1" D="2"/>
    <DPT A="8" B="4301" C="0"/>
    <DPT A="9" B="4302" C="0"/>
    <DPT A="10" B="4303" C="0"/>
    <DPT A="11" B="4305" C="2700"/>
    <DPT A="12" B="4304" C="4500"/>
</DVC>
```

D.17 DeviceAllocation — DAN

Type:

Task data

Description:

The DeviceAllocation XML element includes information on which device(s) the planned task was created for and which devices were actually used during task processing. For a planned task the DeviceAllocation describes the WorkingSetMasterNAMEValue and optionally a NAME mask to enable a range of NAME values which can specify devices that are also allowed for the task processing. During task processing the task controller modifies the WorkingSetMasterNAMEValue attribute to the NAME of the working-set master that was actually used to perform the task.

Includes XML elements:

— *AllocationStamp*

Included by XML elements:

— *Task*

References XML elements:

— *Device*

Attributes:

Attribute	XML	Use	Type	Length/range	Comment
WorkingSetMasterNAMEValue	A	r	xs:hexBinary	0x000000 .. 0xFFFFFFFFFFFFFFFF	NAME of working-set master (see ISO 11783-5) from the device the task is planned for or was processed with.
WorkingSetMasterNAMEMask	B	o	xs:hexBinary	0x000000 .. 0xFFFFFFFFFFFFFFFF	Bit-Mask, which is to be used for a logical AND operation to WorkingSetMasterNAMEValue, to allow more then one specific device for the task. bit=1 =>relevant bit of the WorkingSetMasterNAMEValue bit=0 =>bit of the WorkingSetMasterNAMEValue is not relevant.
DeviceIdRef	C	o	xs:IDREF	min. 4 .. max. 14	Reference to XML element Device Format: (DVC|DVC-)([0-9])+
AllocationStamp		o	xs:element		Includes a single XML element AllocationStamp

EXAMPLE

```
<DAN B="000000FF00000000" A="1234567812345678" C="DVC2">
    <ASP A="2003-08-20T08:00:00" B="2003-08-20T17:00:00" D="4"/>
</DAN>
<DAN B="000000FF00000000" A="8765432187654321" C="DVC1">
    <ASP A="2003-08-20T08:00:00" B="2003-08-20T17:00:00" D="4"/>
</DAN>
```

D.18 DeviceElement — DET

Type:

Coding data

Description:

The DeviceElement XML element describes the functional or physical elements of a device. To establish a hierarchical structure of element groups, a DeviceElement shall refer to another DeviceElement or to the device itself. The DeviceElementType attribute is defined in A.3, "Definition of DeviceElementObject". The ParentObjectId attribute is used to refer either to the DeviceObject (object ID = 0) or to a parent DeviceElementObject to establish a hierarchical ordering of DeviceElements.

Included by XML elements:

— *Device*

Includes XML elements:

— *DeviceObjectReference*

References XML elements:

— *DeviceElement*

— *Device*

Referenced by XML elements:

— *Connection*

— *DataLogTrigger*

— *DataLogValue*

— *ProcessDataVariable*

— *ProductAllocation*

Attributes:

Attribute	XML	Use	Type	Length/range	Comment
DeviceElementId	A	r	xs:ID	min. 4 .. max. 14	Unique identifier of DeviceElement Format: (DET|DET-)([0-9])+ Records generated on MICS have negative IDs
DeviceElementObjectId	B	r	xs:unsignedShort	1 .. 65534	Object ID unique inside a device description
DeviceElementType	C	r	xs:NMTOKEN	1 .. 7	Selection of type: 1 = device 2 = function 3 = bin 4 = section 5 = unit 6 = connector 7 = navigation
DeviceElementDesignator	D	o	xs:string	max. 32	Designator of element
DeviceElementNumber	E	r	xs:unsignedLong	0 .. 4095	Unique number of the element, refer to Annex B: ProcessDataVariable element numbering
ParentObjectId	F	r	xs:unsignedShort	0 .. 65534	Object ID of parent DeviceElement or Device
DeviceObjectReference		o	xs:element		Includes a list of XML elements DeviceObjectReference

EXAMPLE

```
<DET A="DET1" B="1" C="1" D="all elements" E="0" F="0">
    <DOR A="3"/>
    <DOR A="4"/>
    <DOR A="8"/>
    <DOR A="9""/>
    <DOR A="10"/>
    <DOR A="11"/>
</DET>
```

D.19 DeviceObjectReference — DOR

Type:

Coding data

Description:

The DeviceObjectReference XML element describes a reference to a DeviceProcessData object or a DeviceProperty object.

This XML element is part of the device description.

References XML elements:

— *DeviceProcessData*

— *DeviceProperty*

Included by XML elements:

— *DeviceElement*

Attributes:

Attribute	XML	Use	Type	Length/range	Comment
DeviceObjectId	A	r	xs:unsignedShort	1 .. 65534	Object ID of DeviceProcessData or DeviceProperty

EXAMPLE

```
<DOR A="3" />
```

D.20 DeviceProcessData — DPD

Type:

Coding data

Description:

The DeviceProcessData XML element describes ProcessDataVariable DDIs, supported by the DeviceElement that references this XML element. The attributes specify the available trigger methods for a ProcessDataVariable DDI.

This XML element is part of the device description.

References XML element:

— *DeviceValuePresentation*

Referenced by XML element:

— *DeviceObjectReference*

Included by XML elements:

— *Device*

Attributes:

Attribute	XML	Use	Type	Length/range	Comment
DeviceProcessDataObjectId	A	r	xs:unsignedShort	1 .. 65534	Unique number inside a single device
DeviceProcessDataDDI	B	r	xs:hexBinary	0x0000 .. 0xFFFF	Unique number which specifies the process data variable (defined in ISO 11783-11)
DeviceProcessDataProperty	C	r	xs:NMTOKEN	0 .. 3	Bit combination to specify the ProcessDataVariable property: 1 = belongs to default set 2 = setable
DeviceProcessDataTriggerMethods	D	r	xs:integer	0 .. 31	Bit combination to specify supported trigger methods: 1 = time interval 2 = distance interval 4 = threshold limits 8 = on change 16 = total
DeviceProcessDataDesignator	E	o	xs:string	max. 32	Designator of DeviceProcessData
DeviceValuePresentationObjectId	F	o	xs:unsignedShort	1 .. 65534	Object ID of DeviceValuePresentation

EXAMPLE

<**DPD** A="1" B="1234" C="3" D="1"/>

D.21 DeviceProperty — DPT

Type:

Coding data

Description:

The DeviceProperty XML element describes a property of a DeviceElement by means of a reference and a value for a DDI. This XML element is part of the device description.

References XML element:

— *DeviceValuePresentation*

Referenced by XML element:

— *DeviceObjectReference*

Included by XML elements:

— *Device*

Attributes:

Attribute	XML	Use	Type	Length/range	Comment
DevicePropertyObjectId	A	r	xs:unsignedShort	1 .. 65534	Unique number inside a single device
DevicePropertyDDI	B	r	xs:hexBinary	0x0000 .. 0xFFFF	Unique number which defines the property (specified in ISO 11783-11)
DevicePropertyValue	C	r	xs:long	-2^{32} .. $(2^{32}-1)$	Property value
DevicePropertyDesignator	D	o	xs:string	max. 32	Optional designator for property
DeviceValuePresentationObjectId	E	o	xs:unsignedShort	1 .. 65534	Object ID of DeviceValuePresentation

EXAMPLE

```
<DPT A="8" B="1235" C="-65233"/>
```

D.22 DeviceValuePresentation — DVP

Type:

Coding data

Description:

The DeviceValuePresentation XML element is used to specify the presentation of the data dictionary entity-defined integer values that are used within a single device. The presentation shall be according to the following formula:

Presented value = (integer value + Offset) * Scale

Presented values are always rounded to the number of decimals specified in the NumberOfDecimals attribute.

Referenced by XML elements:

— *DeviceProcessData*

— *DeviceProperty*

Included by XML elements:

— *Device*

Attributes:

Attribute	XML	Use	Type	Length/range	Comment
DeviceValuePresentationObjectId	A	r	xs:unsignedShort	1 .. 65534	Unique number inside a single device
Offset	B	r	xs:long	-2^{31} .. $(2^{31}-1)$	Offset to be applied to the value for presentation
Scale	C	r	xs:decimal	0.000000001 .. 100000000.0	Scale to be applied to the value for presentation
NumberOfDecimals	D	r	xs:unsignedByte	0 .. 7	Number of decimals to be presented after the decimal separator
UnitDesignator	E	o	xs:string	max. 32	Optional unit designator string

EXAMPLE

<**DVP** A="1" B="0" C="1.0" D="0" E="kg"/>

<**DVP** A="2" B="32" C="1.8" D="1" E="°F"/>

D.23 Farm — FRM

Type:

Coding data

Description:

The Farm XML element contains all required information to describe a farm. The relationships between a customer and farms and partfields can be multiple. To determine which farms or partfields belong to a specific customer, the CustomerIdRef values of all farms or partfields, respectively, shall be examined for a match with a particular CustomerId value.

References XML element:

— *Customer*

Referenced by XML elements:

— *Partfield*

— *Task*

Attributes:

Attribute	XML	Use	Type	Length/range	Comment
FarmId	A	r	xs:ID	min. 4 .. max. 14	Unique identifier of Farm Format: (FRM|FRM-)([0-9])+ Records generated on MICS have negative IDs
FarmDesignator	B	r	xs:string	max. 32	Farm designator/name
FarmStreet	C	o	xs:string	max. 32	Street
FarmPOBox	D	o	xs:string	max. 32	PO Box
FarmPostalCode	E	o	xs:string	max. 10	Postal code
FarmCity	F	o	xs:string	max. 32	City
FarmState	G	o	xs:string	max. 32	State or county
FarmCountry	H	o	xs:string	max. 32	Country
CustomerIdRef	I	o	xs:IDREF	min. 4 .. max. 14	Reference to XML element customer Format: (CTR|CTR-)([0-9])+

EXAMPLE

```
<FRM A="FRM1" B="bonanza ranch" />
```

D.24 Grid — GRD

Type:

Task data

Description:

The Grid XML element describes the dimension and position of a set of gridcells. There shall be a definition of the minimum north/east position, the size of each gridcell and the number of gridcells in the north/east direction. There can only be a single grid specified per task. The grid is always related to a partfield but the definition of grid is always task-specific. The grid cells of a grid contain a reference to a TreatmentZone or a process data variable value. The grid shall be specified as a complete array of gridcells in ascending order, as gridcells do not contain any ordering information.

The gridcells shall be defined in a binary format in a separate file external to the XML data transfer file. There can only be a single binary file per grid and per task. The gridcell files shall exist in the same folder as the data transfer file. The name of the gridcell files shall be unique over all grids referred to by tasks of a data transfer file.

References XML element:

— *TreatmentZone*

Included by XML elements:

— *Task*

Attributes:

Attribute	XML	Use	Type	Length/range	Comment
GridMinimumNorthPosition	A	r	xs:decimal	–90.0 .. 90.0	Minimum north position of the grid Format: WGS84
GridMinimumEastPosition	B	r	xs:decimal	–180.0 .. 180.0	Minimum east position of the grid Format: WGS84
GridCellNorthSize	C	r	xs:double	0.0 .. 1.0	North direction gridcell size Format: WGS84
GridCellEastSize	D	r	xs:double	0.0 .. 1.0	East direction gridcell size Format: WGS84
GridMaximumColumn	E	r	xs:unsignedLong	0 .. (2^{32}–1)	Number of the gridcells in east direction
GridMaximumRow	F	r	xs:unsignedLong	0 .. (2^{32}–1)	Number of the gridcells in north direction
Filename	G	r	xs:ID	8	Unique name of gridcell file Format: GRD[0-9][0-9][0-9][0-9][0-9]
Filelength	H	o	xs:unsignedLong	0 .. (2^{32}–2)	Length of gridcell file in number of bytes
GridType	I	r	xs:NMTOKEN	1 .. 2	Grid type specification: 1 = grid type 1 2 = grid type 2
TreatmentZoneCode	J	o	xs:unsignedByte	0 .. 254	Grid type 2 TreatmentZoneCode

EXAMPLE

Grid type 1 XML element specification:

<**GRD** A="58.096653" B="8.54321" C="0.012" D="0.012" E="200" F="300" G="GRD00001" I="1"/>

See 7.6 for examples of both grid type 1 and grid type 2 specifications.

D.25 ISO11783_TaskData

Type:

Root element

Description:

The XML element ISO11783_TaskData is the main XML element called a root element and contains definitions about the construct of the XML File (Version number…) and the use of primary XML elements.

Includes XML elements:

— *Task*

— *CodedComment*

— *CodedCommentGroup*

— *ColourLegend*

— *CropType*

— *CulturalPractice*

— *Customer*

— *Farm*

— *Device*

— *OperationTechnique*

— *Partfield*

— *Product*

— *ProductGroup*

— *ValuePresentation*

— *Worker*

— *ExternalFileReference*

Attributes:

Attribute	XML	Use	Type	Length/ range	Comment
VersionMajor	VersionMajor	r	xs:NMTOKEN	2	List element release number (major), used to specify the version of ISO 11783-10 that this task data file meets: 0 = The version of the DIS (Draft International Standard) 1 = The version of the FDIS (Final Draft International Standard) 2 = The version of the first edition published as an International Standard, and so on

Attribute	XML	Use	Type	Length/ range	Comment
VersionMinor	VersionMinor	r	xs:NMTOKEN	1	List element version number (minor)
ManagementSoftwareManufacturer	Management Software Manufacturer	r	xs:string	32	Name of management-software manufacturer
ManagementSoftwareVersion	Management Software Version	r	xs:string	32	Version of management software
TaskControllerManufacturer	TaskController Manufacturer	o	xs:string	32	Name of task controller manufacturer
TaskControllerVersion	TaskController Version	o	xs:string	32	Version of task controller software
DataTransferOrigin	DataTransfer Origin	r	xs:NMTOKEN	1	Describes the origin of the XML file: 1 = FMIS 2 = MICS
CodedComment	CCT	o	xs:element		Includes XML element CodedComment
CodedCommentGroup	CCG	o	xs:element		Includes XML element CodedCommentGroup
ColourLegend	CLD	o	xs:element		Includes XML element ColourLegend
CropType	CTP	o	xs:element		Includes XML element CropType
CulturalPractice	CPC	o	xs:element		Includes XML element CulturalPractice
Customer	CTR	o	xs:element		Includes XML element Customer
Farm	FRM	o	xs:element		Includes XML element Farm
Device	DVC	o	xs:element		Includes XML element Device
OperationTechnique	OTQ	o	xs:element		Includes XML element OperationTechnique
Partfield	PFD	o	xs:element		Includes XML element Partfield
Product	PDT	o	xs:element		Includes XML element Product
ProductGroup	PGP	o	xs:element		Includes XML element ProductGroup
Task	TSK	o	xs:element		Includes XML element Task
ValuePresentation	VPN	o	xs:element		Includes XML element ValuePresentation
Worker	WKR	o	xs:element		Includes XML element Worker
ExternalFileReference	XFR	o	xs:element		Includes XML element ExternalFileReference

EXAMPLE

```
<ISO11783_TaskData VersionMajor="1" VersionMinor="0" TaskControllerManufacturer="FarmCtrl" TaskControllerVersion="1.0"
        ManagementSoftwareManufacturer="FarmSystem" ManagementSoftwareVersion="1.0" DataTransferOrigin="1">
    <TSK A="TSK1" F="1" E="WKR1">
    ……
    </TSK>
    ……
</ISO11783_TaskData>
```

D.26 Linestring — LSG

Type:

Task data

Description:

The Linestring XML element describes the position, length and appearance of a line. Linestrings of the type flag can be used to assign a comment to all positions of the linestring. This is to enable the task controller to display farm-side-generated comments stored in the XML attribute LinestringDesignator at certain positions as informational messages to the operator.

Includes XML element:

— *Point*

Included by XML elements:

— *Partfield*

— *Polygon*

Attributes:

Attribute	XML	Use	Type	Length/range	Comment
LinestringType	A	r	xs:NMTOKEN	1 .. 8	Type of the linestring, possible values: 1 = PolygonExterior 2 = PolygonInterior 3 = TramLine 4 = SamplingRoute 5 = GuidancePath 6 = Drainage 7 = Fence 8 = Flag
LinestringDesignator	B	o	xs:string	max. 32	Linestring name or comment
LinestringWidth	C	o	xs:unsignedLong	0 .. (2^{32}–2)	Width of the linestring in mm
LinestringLength	D	o	xs:unsignedLong	0 .. (2^{32}–2)	Length of the linestring in mm
LinestringColour	E	o	xs:unsignedByte	0 .. 254	Colour of the linestring Format: palette like ISO 11783-6
Point		r	xs:element		Includes a list of XML element Point

EXAMPLE

```
<LSG A="1" E="1" B="Line1" D="2000" C="20">
    <PNT A="2" C="58.8754321" D="8.945632" F="1" B="start" E="50"/>
    <PNT A="2" C="58.8789999" D="8.99889099" F="1" B="end" E="50"/>
</LSG>
```

D.27 OperationTechnique — OTQ

Type:

Coding data

Description:

The OperationTechnique XML element describes operation techniques like “drilling”, “spreading”, “gaseous”.

Referenced by XML elements:

— *OperTechPractice*

— *OperationTechniqueReference*

Attributes:

Attribute	XML	Use	Type	Length/range	Comment
OperationTechniqueId	A	r	xs:ID	min. 4 .. max. 14	Unique identifier of OperationTechnique Format: (OTQ|OTQ-)([0-9])+ Records generated on MICS have negative IDs
OperationTechniqueDesignator	B	r	xs:string	max. 32	Designator of OperationTechnique

EXAMPLE

```
<OTQ A="OTQ1" B="drilling"/>
<OTQ A="OTQ2" B="spreading"/>
<OTQ A="OTQ3" B="gaseous"/>
```