

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

Part 10: Task controller and management information system data interchange

ICS 35.240.99; 65.060.01

National foreword

This British Standard is the UK implementation of ISO 11783-10:2009.

The UK participation in its preparation was entrusted to Technical Committee AGE/6, Agricultural tractors and forestry machinery.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 January 2010
© BSI 2010

ISBN 978 0 580 58999 7

Amendments/corrigenda issued since publication

Date	Comments

**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*



PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.



COPYRIGHT PROTECTED DOCUMENT

© ISO 2009

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword	iv
Introduction.....	v
1 Scope	1
2 Normative references	1
3 Terms and definitions	1
4 Abbreviations.....	2
5 General description	2
5.1 Task management	2
5.2 Task management on FMIS computer.....	4
5.3 Preselection and assignment of working sets	4
5.4 Task-controller interface driver	4
5.5 Task-controller user interface	4
6 Task-controller requirements.....	5
6.1 Task selection and execution	5
6.2 Logging of time and position	5
6.3 Logging parameters from parameter groups	6
6.4 Connection management.....	6
6.4.1 Task-controller initialization.....	6
6.4.2 Working-set initialization with the task controller	6
6.5 Connection management.....	7
6.6 Data exchange on the network	9
6.6.1 Site-specific application	9
6.6.2 Data logging	12
7 Data transfer	14
7.1 General	14
7.2 Extensible Markup Language.....	15
7.3 Extensible schema definition	15
7.4 XML schema definition	16
7.5 XML data transfer files	18
7.6 Binary data transfer files	20
7.6.1 General	20
7.6.2 Grid binary file structure.....	20
7.6.3 Log data binary file structure	23
7.7 Device description data	25
Annex A (normative) Device description objects	29
Annex B (normative) Message definitions	36
Annex C (normative) XML elements relationship diagram	48
Annex D (normative) XML elements and attributes	50
Bibliography.....	101

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.

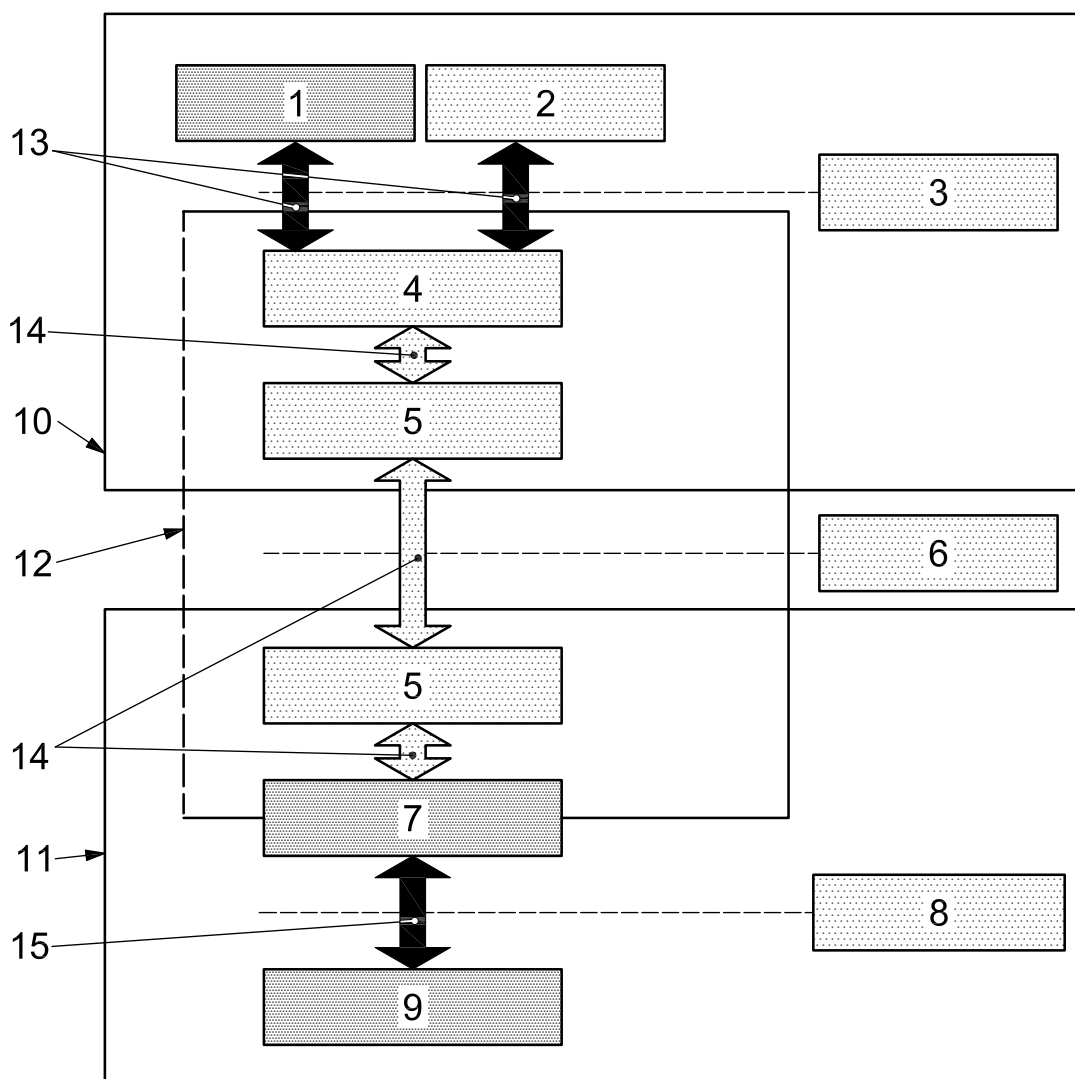
The second purpose is the management of the farm activities carried out in the fields. These activities are described by tasks to distinguish all the work that is planned or has been done by the farmer or by a contractor for one customer in one partfield.

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 <i>completed</i> 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.

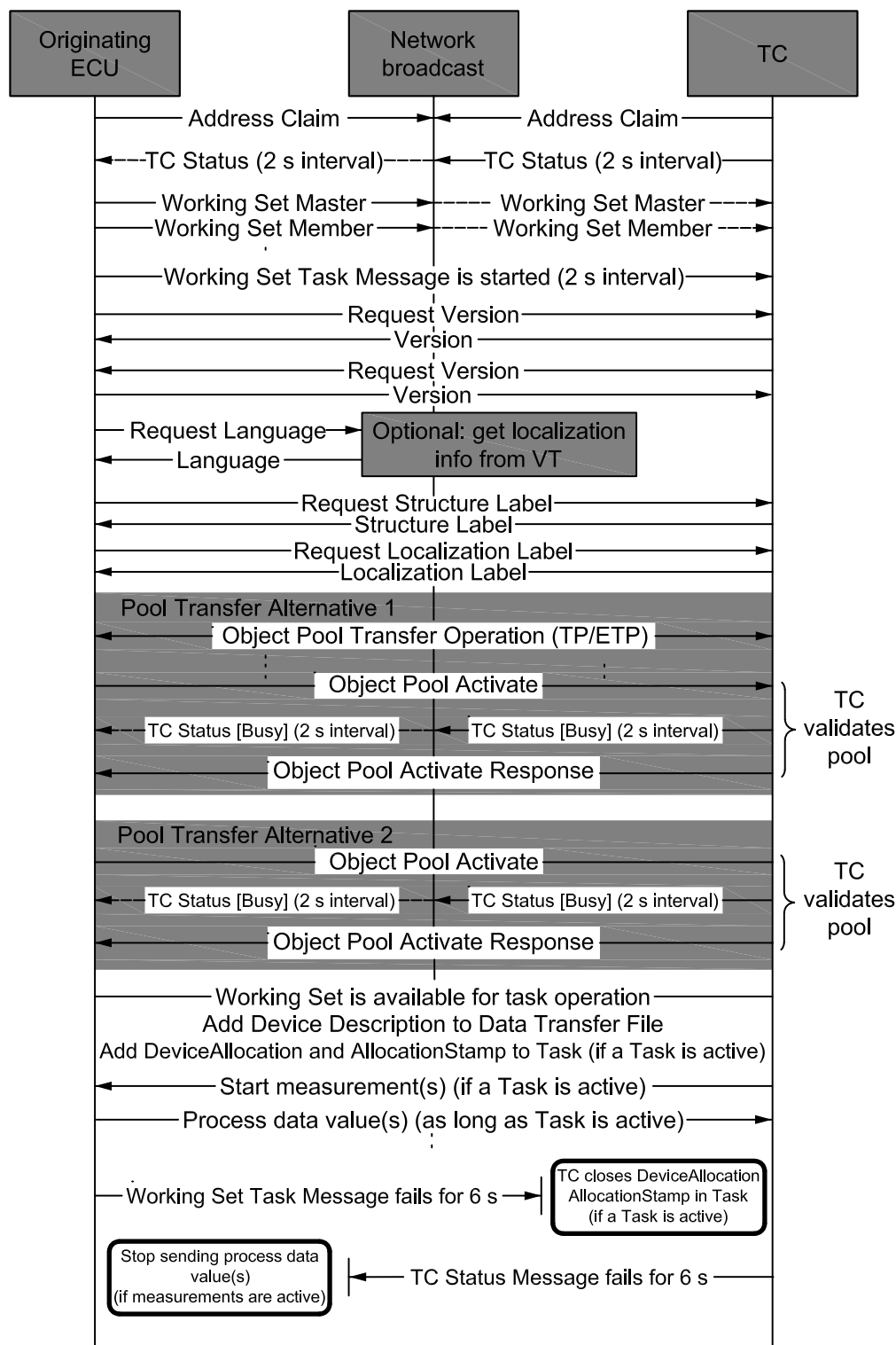


Figure 2 — Initialization and shutdown of connections

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.

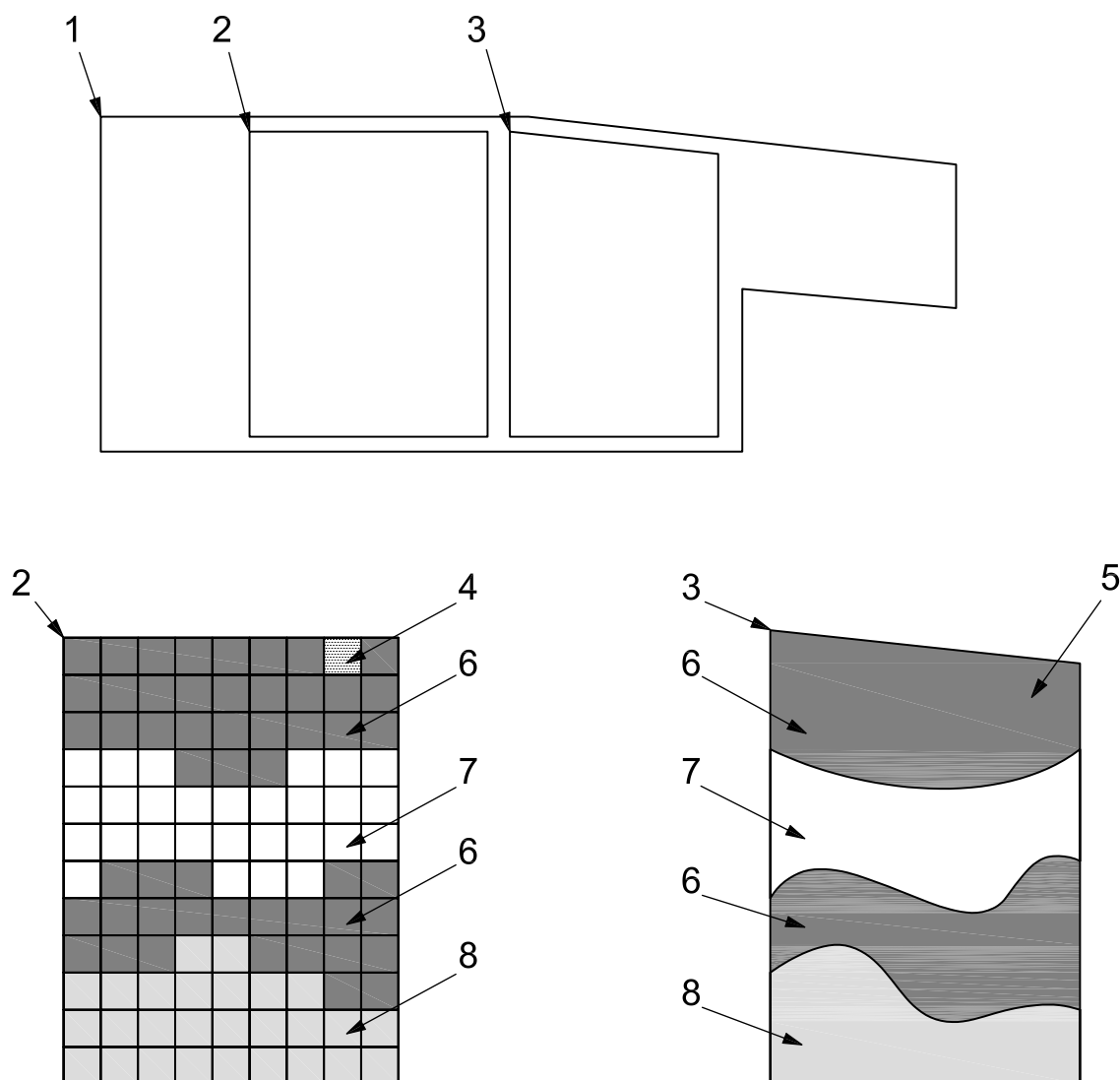
All task-controller-specific data exchange on the ISO 11783 network is based on process data messages.

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.

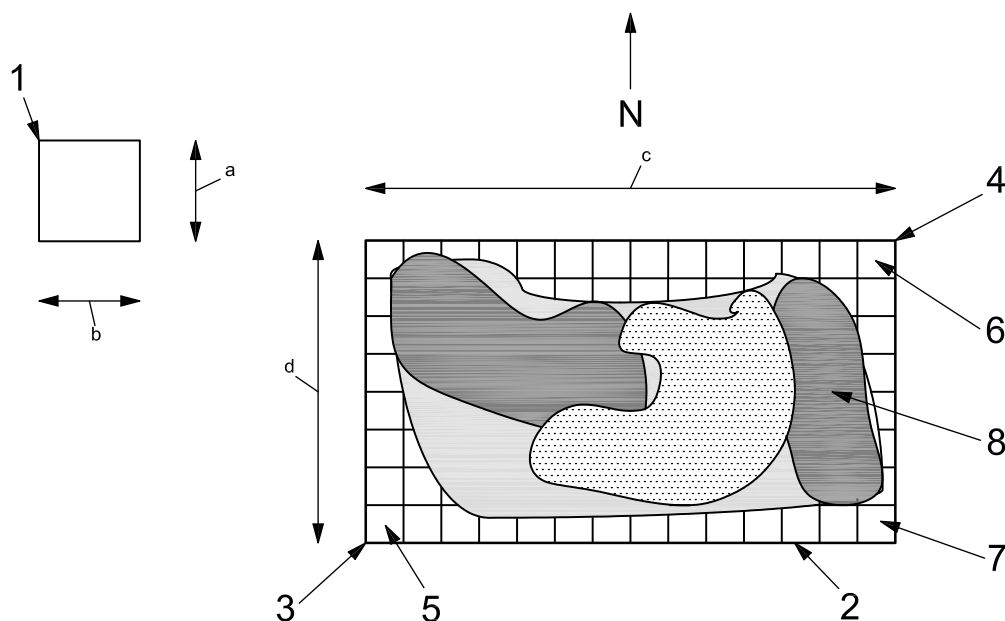


Key

- 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 = DFFF₁₆) 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 = DFFF₁₆ 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 1F₁₆.

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 8D₁₆ and 43₁₆ 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.

- Time and position data shall be logged at a maximum once per Time or TimeLog instance.
- 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.
- 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.
- 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 = 0093₁₆) 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>
```

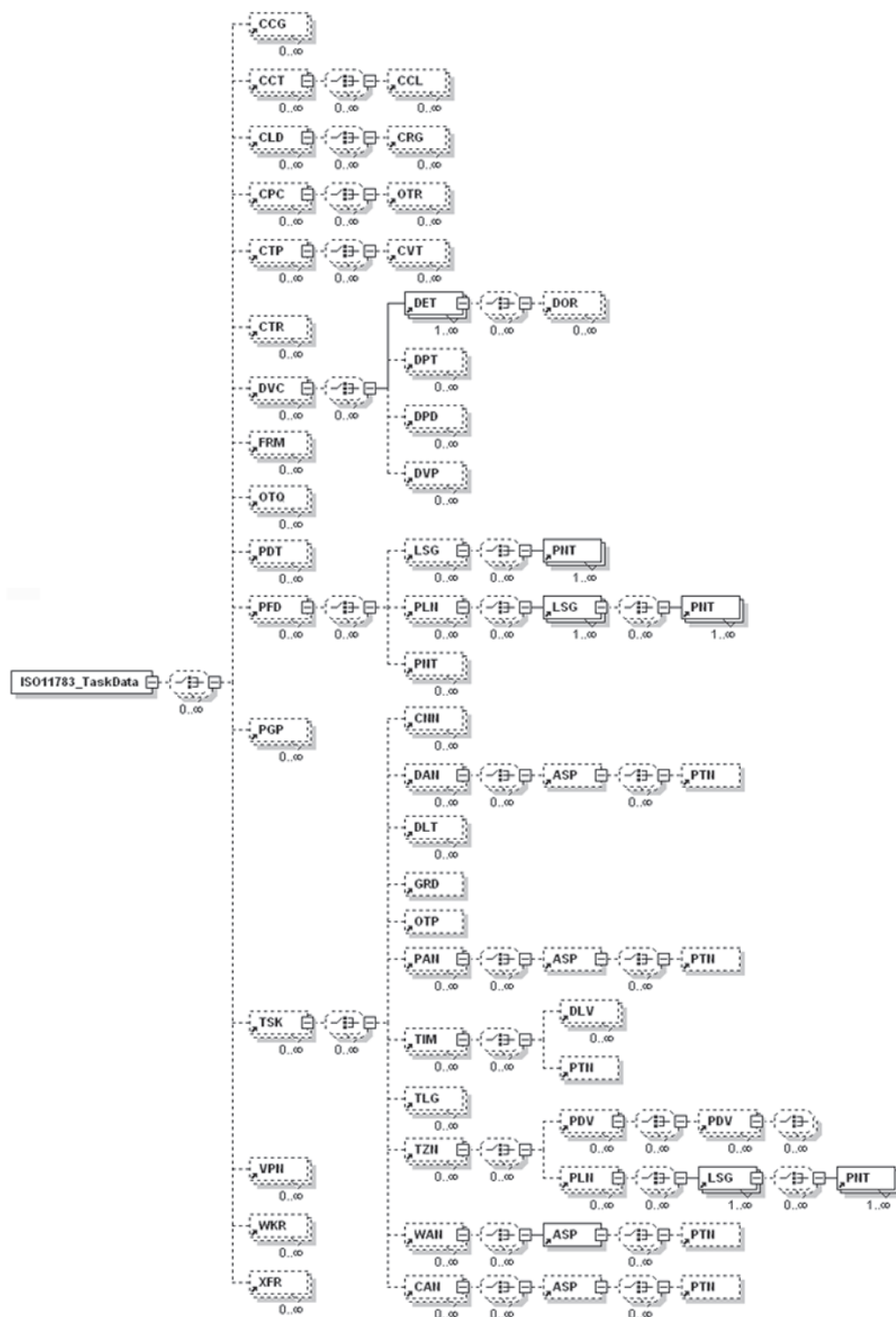


Figure 5 — Schematic structure of a data transfer file

Figure 6 shows the usage of referenced XML files.

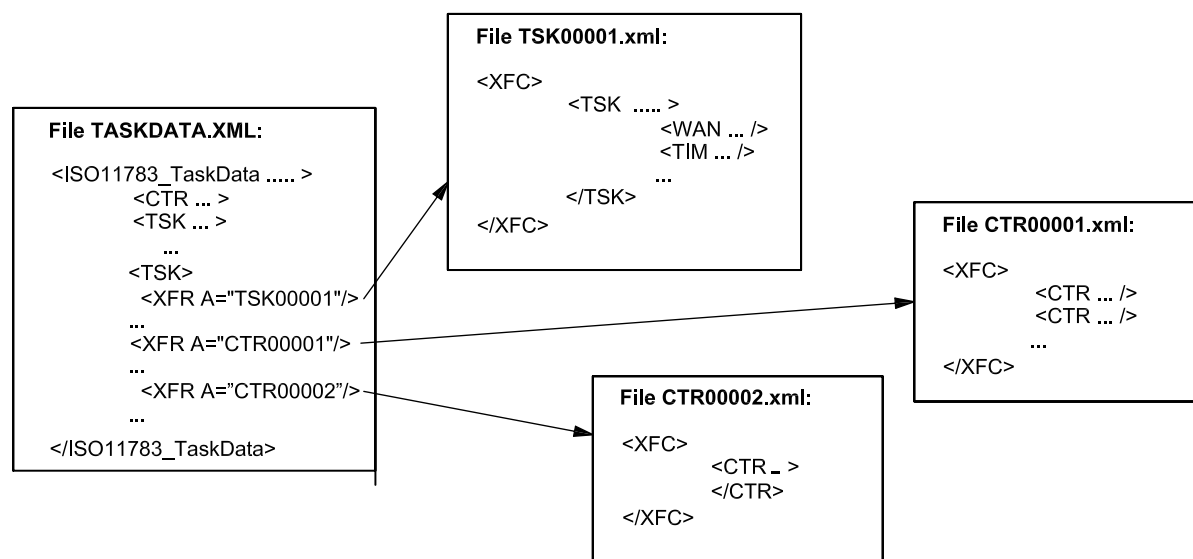


Figure 6 — XFR/XFC XML file references

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 TreatmentZoneldRef 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.

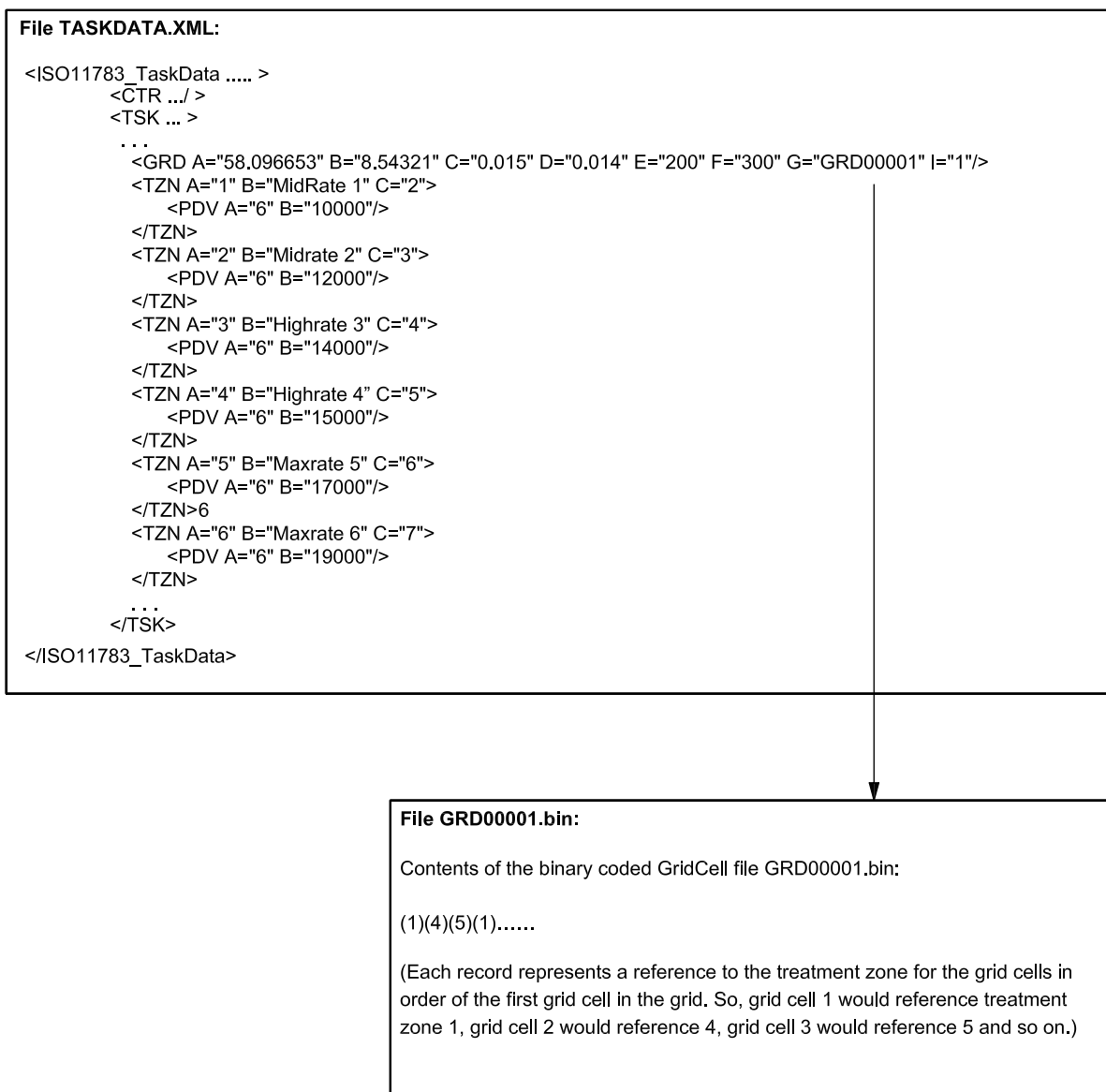


Figure 7 — Grid type 1 binary file example

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.

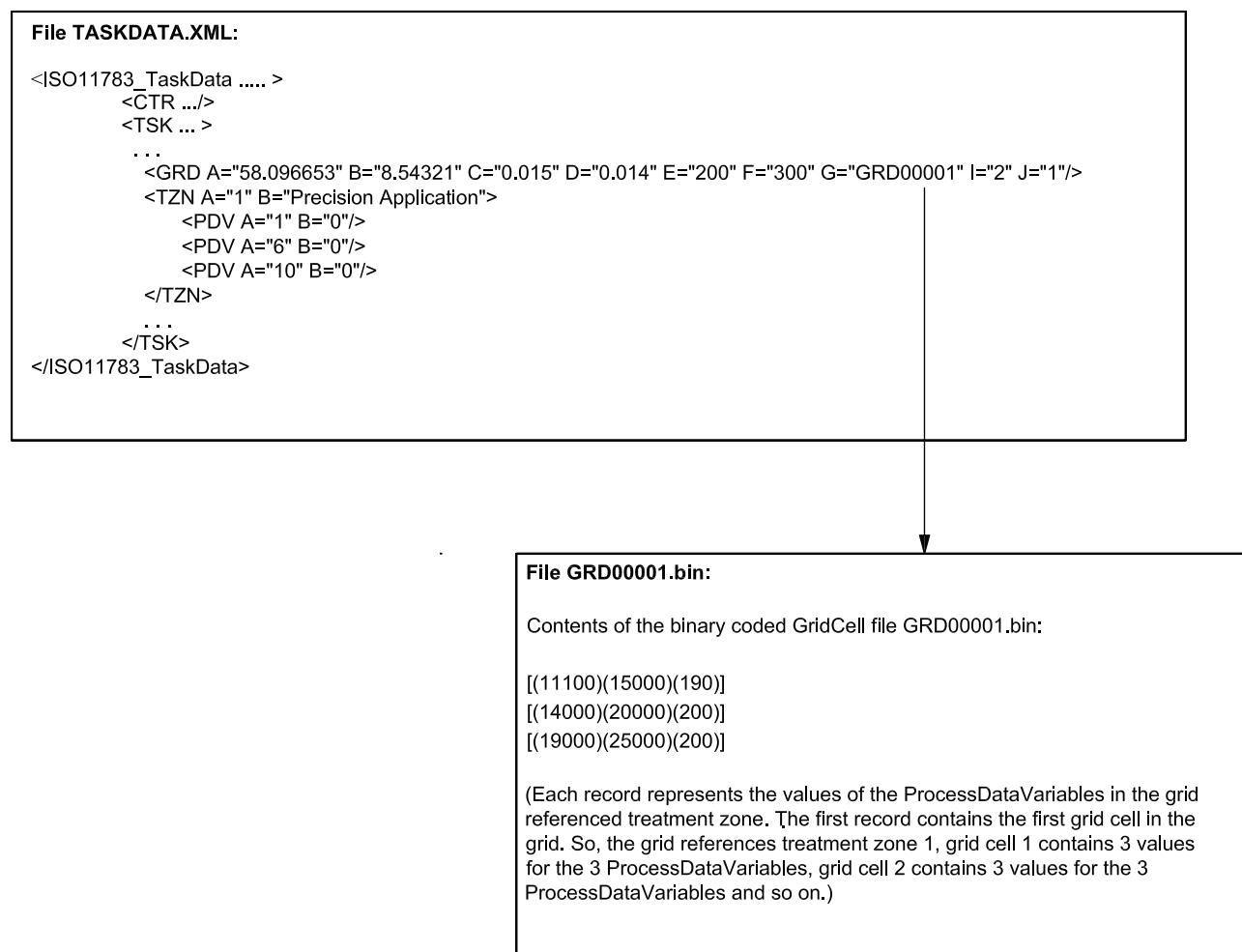


Figure 8 — Grid type 2 binary file example

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.

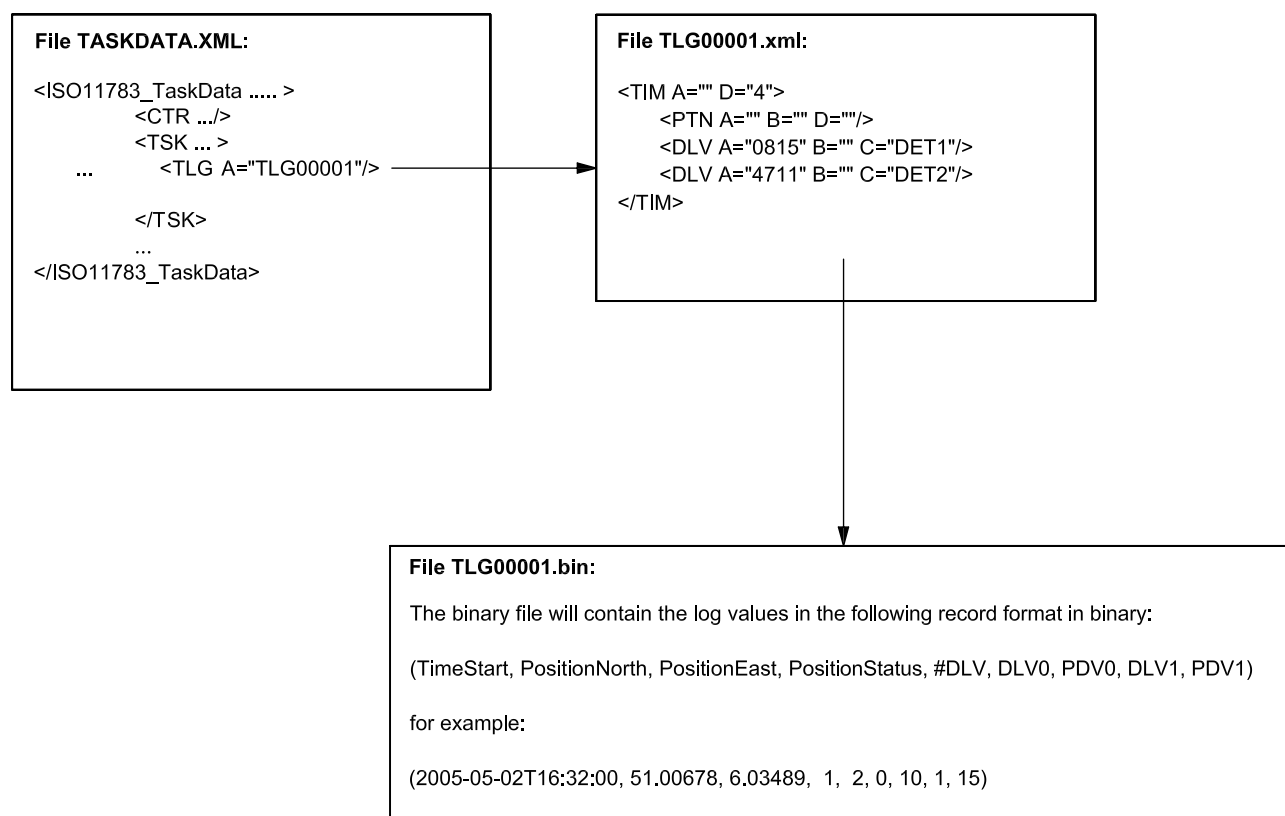


Figure 9 — Log data binary file example

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.